Addressing Efficiency and Reliability Challenges in Natural Language Processing

by

Neeraj Varshney

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2024 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Yezhou Yang
Nakul Gopalan
Pratyay Banerjee

ARIZONA STATE UNIVERSITY

May 2024

ABSTRACT

Recently developed large language models have achieved remarkable success on a wide range of natural language tasks. Furthermore, they have been shown to possess an impressive ability to generate fluent and coherent text. Despite all the notable abilities of these models, there exist several efficiency and reliability related challenges. For example, they are vulnerable to a phenomenon called 'hallucination' in which they generate text that is not factually correct and they also have a large number of parameters which makes their inference slow and computationally expensive. With the objective of taking a step closer towards further enabling the widespread adoption of the Natural Language Processing (NLP) systems, this dissertation studies the following question: how to effectively address the efficiency and reliability related concerns of the NLP systems?

Specifically, to improve the reliability of models, this dissertation first presents an approach that actively detects and mitigates the hallucinations of LLMs using a retrieval augmented methodology. Note that another strategy to mitigate incorrect predictions is abstention from answering when error is likely, i.e., selective prediction. To this end, I present selective prediction approaches and conduct extensive experiments to demonstrate their effectiveness. Building on top of selective prediction, I also present post-abstention strategies that focus on reliably increasing the coverage of a selective prediction system without considerably impacting its accuracy. Furthermore, this dissertation covers multiple aspects of improving the efficiency including 'inference efficiency' (making model inferences in a computationally efficient manner without sacrificing the prediction accuracy), 'data sample efficiency' (efficiently collecting

data instances for training a task-specific system), 'open-domain QA reader efficiency' (leveraging the external knowledge efficiently while answering open-domain questions), and 'evaluation efficiency' (comparing the performance of different models efficiently). In summary, this dissertation highlights several challenges pertinent to the efficiency and reliability involved in the development of NLP systems and provides effective solutions to address them.

ACKNOWLEDGMENTS

I would like to begin by sincerely thanking my dissertation advisor Dr. Chitta Baral for his guidance, mentorship, and support throughout my doctoral journey. Dr. Baral's encouragement and insightful feedback have been instrumental in every step of my research. It has been my privilege to learn from and work under his supervision during my PhD.

I am also thankful to the members of my dissertation committee Dr. Yezhou Yang, Dr. Nakul Gopalan, and Dr. Pratyay Banerjee for their valuable insights and constructive feedback which has helped me immensely in structuring my thesis.

I would also like to express my sincere gratitude to my internship mentors from Tencent AI (Jianshu Chen, Wenlin Yao, Hongming Zhang, and Dong Yu) and Amazon Science (Ankit Chadha, Alessandro Moschitti, Rik koncel-kedziorski, Siddhant Garg, and Ivano Lauriola).

I am highly grateful to Swaroop Mishra under whose guidance and mentorship my research journey began. His consistent support and encouragement have played a very crucial role in my PhD. I would also like to thank my wonderful peers and collaborators: Tejas Gokhale, Kuntal Pal, Man Luo, Pulkit Verma, Bhavdeep Sachdeva, Mihir Parmar, Shailaja Sampat, Himanshu Gupta, Maitreya Patel, Agneet Chatterjee, Pavel Dolin, Agastya Seth, Mutsumi Nakamura, Santosh Mashetty, Mirali Purohit, Eric Robertson, Bing Liu, Nisarg Patel, Ming Shen, Yizhong Wang, Arindam Mitra, Shrinidhi Kumbhar, Daniel Khashabi, Ashwin Kalyan, Ayushi Agarwal, Nemika Tyagi, Aswin RRV, Md Nayem Uddin, Divij Handa, Satyam Raj, Venkatesh Mishra, Ritika Sarkar, and Amir Saeidi.

I am thankful to my friends, Karan, Rahul, Mohit, Mihir, Abhishek, Paritosh, Deepit, Saurabh, Manan, Anmol, Aviral and my colleagues and mentors from Microsoft: Darshat, Amit, Lekshmi, Ayush, Yasir, Neethi, Deepika, Luqman, Shivangi, Juhi, and Shubham for providing me with the confidence to pursue PhD.

I would also like to acknowledge the support of DARPA, ASU HPC, Mayo Clinic, GPSA, Amazon Research Award, and SCAI for financial and computing resources.

Lastly, I would like to acknowledge the support of my family for their unwavering encouragement and understanding. Their belief in me has been a constant source of strength and motivation.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

Table Page

Table Page

Table Page

Table Page

Table Page

Table Page

LIST OF FIGURES

Figure Page

Figure | Page

Figure | Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Overview

Recent years have seen remarkable advancements in the field of Natural Language Processing (NLP) which have led to the development of a number of pre-trained language models such as BERT (Devlin et al. 2019), RoBERTa (Roberts, Raffel, and Shazeer 2020), XLNET (Z. Yang et al. 2019a), BART (Lewis et al. 2020), T5 (Raffel et al. 2020a), GPT series models (Brown et al. 2020; OpenAI 2023), LLaMA (Touvron et al. 2023), and several others. These models have been shown to perform well on a variety of natural language tasks. Despite this impressive performance, we note that there exist numerous reliability and efficiency related concerns with these models such as

- They are **vulnerable to 'hallucination'** in their output as they often generate text that is factually incorrect
- They are **not absolutely perfect in their predictions**, i.e., they often make incorrect predictions
- They are **not quite effective in performing tasks that involve 'negation'**
- Their **inference is computationally very expensive** due to the large number of parameters in the model architecture

- They typically **require a large number of high-quality examples for training**
- Their **cost of inference considerably increases when retrieval** of external knowledge is required
- The performance **comparison of the competing models is also very expensive** both in terms of computation and latency
- The **decoding procedure during text generation is computationally very expensive** due to the autoregressive nature of inference where one token is generated at a time

The above list clearly shows that developing and using an NLP system involves numerous critical efficiency and reliability related challenges. With the objective of taking a step closer towards enabling the widespread adoption, this dissertation studies the following question: "*how to effectively address these critical efficiency and reliability related concerns of NLP systems?*"

We elaborate on these aspects and provide an overview of our solutions in Section 1.2. Followed by this, we highlight the impact of our solutions in Section 1.3. We provide a summary of all the chapters of this dissertation in Section 1.4. Finally, in Section 1.5, we list down the corresponding papers in which these research ideas have appeared.

1.2 Solutions to Address the Efficiency and Reliability Challenges

In this section, we elaborate on the efficiency and reliability challenges and provide an overview of our solution to address those challenges.

Addressing the Hallucination Problem of LLMs: Hallucination in the context of language models refers to the generation of text that seems syntactically sound, fluent, and natural but is factually incorrect, nonsensical, or unfaithful to the provided source input (Maynez et al. 2020; Holtzman et al. 2020; Ji et al. 2023). Recently developed language models often tend to 'hallucinate' in their output which critically hampers their reliability. These hallucinations can lead to serious consequences such as the spreading of misinformation and violation of privacy. It also raises safety concerns for real-world applications. For example in medical applications, a hallucinated report generated from a patient's information can pose serious risks to the patient. It can even provoke a life-threatening incident for the patient. Hallucinations hamper the reliability and trustworthiness of the model which makes it important to address this problem.

Addressing the above problem, we propose an approach that actively detects and mitigates hallucinations during the generation process using a retrieval augmented methodology in Chapter 3. This is crucial as we show that when a sentence generated by a model is hallucinated, the chances of hallucination in the subsequently generated sentences increase, i.e., hallucinations often propagate in the model's output. This can be attributed to the autoregressive nature of the LLMs and the discrepancy between the training and inference time decoding. Specifically, during the training time, the model is encouraged to predict the next token conditioned on the ground-truth prefix tokens. However, at inference time, the model generates the next token conditioned on the historical tokens previously generated by itself. Thus, actively detecting and mitigating hallucinations during the generation process also facilitates in preventing the propagation of hallucinations in the generation. Specifically, we first identify the candidates of potential hallucination leveraging the model's *logit output values*, check their correctness through a *validation* procedure, mitigate the detected hallucinations via *prompting*, and then continue with the generation process. This active intervention also facilitates in preventing the propagation of hallucinations in the LLM's output. We demonstrate the effectiveness and wide applicability of our approach through experiments with article generation task, multi-hop QA, and false premise QA tasks (Varshney, Yao, et al. 2023).

The above work focuses on general tasks; however, an important component of language is 'negation'. Negation is important because it adds depth and nuance to the understanding of language and is also crucial for logical reasoning and inference. To this end, we extend the above research on hallucination and further study it in tasks involving negation in Chapter 4. Specifically, we study four tasks with negation: 'false premise completion', 'constrained fact generation', 'multiple choice question answering', and 'fact generation'. We show that open-source state-of-the-art LLMs such as LLaMA-2-chat, Vicuna, and Orca-2 hallucinate considerably on all these tasks involving negation which underlines a critical shortcoming of these models. Addressing this problem, we further study numerous strategies to mitigate these hallucinations and demonstrate their impact.

Note that the models always output predictions; however, reliability can also be improved via selective prediction in which we enable the systems to abstain from making predictions when they are likely to be incorrect. To this end, we study this aspect in the subsequent chapter.

Reducing Prediction Errors via Selective Prediction: As motivated above, we note that another aspect of reliability is focused on reducing the prediction errors of the models. This is important because NLP models are not absolutely perfect, i.e., they often make incorrect predictions that hamper their reliability. Selective prediction partly addresses these concerns by enabling systems to abstain from making predictions when they are likely to be incorrect (Varshney, Mishra, and Baral 2022b; Kamath, Jia, and Liang 2020). Avoiding incorrect predictions allows them to maintain high task accuracy and thus makes them more reliable. In Chapter 5, we first provide mathematical formulation of the selective prediction task and then present several selective prediction methods such as maximum softmax probability, monte-carlo dropout, label smoothing, and calibration in in-domain, out-of-domain, and adversarial settings.

We note that while the selective prediction approaches are advantageous, they leave us with a pertinent question '*what to do after abstention*'. We address this question in the subsequent chapter.

Increasing the Coverage of a Selective Prediction System via Post-Abstention Strategies: In Chapter 6, we study the question of '*what to do after abstention*' and present 'post-abstention' approaches that allow re-attempting the abstained instances with the aim of increasing *coverage* of the given selective prediction system without significantly sacrificing its *accuracy*.

Given the scale at which the natural language processing research has reached, developing and using an NLP system also involves a number of efficiency related challenges. To this end, we next study approaches to jointly improve the efficiency and accuracy.

Jointly Improving Inference Efficiency and Accuracy: The large size of the models (number of parameters) makes their inference slow and computationally expensive. This poses a practical challenge limiting their widespread adoption in resource constrained real-world applications. Improving the efficiency of systems provides multiple benefits such as it reduces the latency of predictions enabling their use in time-sensitive applications also, it reduces the computational requirements to use these models making them fit for low-resource settings also. Focusing on this problem, we first study instruction tuning LLMs with additional explicit **L**osses from the **I**n**T**ermediate lay**E**rs (`LITE`) and show that it enables these layers to acquire 'good' generation ability without affecting the generation ability of the final layer. We then perform '*dynamic confidence-based early exiting*' at token level from the intermediate layers which improves the computational efficiency of text generation without sacrificing the quality of the generation. We conduct comprehensive experiments by instruction tuning LLaMA-2 models on the Alpaca dataset (Taori et al. 2023) and evaluate on four different instruction test sets. We show that dynamic confidence-based early exiting achieves consistent and considerable inference cost improvements while maintaining the generation quality of the outputs (Varshney, Chatterjee, et al. 2023).

The above dynamic early exiting method requires re-training (tuning with LITE) the model. However, this raises a question of what if we have availability of models of varying sizes. We address this question in Chapter 8 where we propose a method of 'model cascading' which utilizes a collection of models of varying capacities to accurately yet efficiently output predictions. Cascading aims at improving the computational efficiency of the system without sacrificing its prediction performance. Note that this idea of cascading can be extended to a more general task of open-domain question answering and we study this in the subsequent chapter.

Improving QA Reader Efficiency while Leveraging External Knowledge: A typical open-domain QA system involves Retriever-reader pipeline (Danqi Chen et al. 2017; Karpukhin et al. 2020; Khattab and Zaharia 2020; Izacard and Grave 2021) in which the retriever finds top-N relevant passages and the reader leverages them to predict the answer. Prior work has shown that the reader's performance tends to improve (up to a certain extent) with the increase in the value of N. Thus, state-of-the-art models use a large number of passages (e.g. 100). While this strategy results in a high prediction performance, it makes the inference of the reader computationally very expensive. For instance, Fusion-in-Decoder reader model (FiD) (Izacard and Grave 2021) requires approximately 70×10^{11} floating-point operations (FLOPs) for an inference with 100 passages. This high inference cost limits the widespread adoption of such systems in applications that prefer efficient systems to be able to achieve low response times.

Building on top of the concept of cascading, we propose a method of leveraging the external knowledge efficiently while correctly answering open-domain questions in Chapter 9. Specifically, we propose an approach that utilizes both the 'closed-book' (parametric knowledge) and the 'open-book' (external knowledge) inferences in an efficient manner. Furthermore, instead of using a large fixed number of passages for open-book inference, we dynamically read the external knowledge in multiple 'knowledge iterations'. We demonstrate that our approach improves both the inference efficiency and the prediction accuracy of the reader. In the next chapter, we focus on improving the data sample efficiency of training.

Improving Data Sample Efficiency of Training: Training a model to perform a task often requires a large amount of data which is typically collected via crowdsourcing. Crowdsourcing data presents several challenges, such as it is very expensive, time-consuming, and often yields a large number of trivial/easy examples. These issues hamper the quality of the data and in turn the performance of the models trained on that data. In Chapter 10, we address these limitations and propose a method called "PHL triplet generation" which focuses on the creation of high-quality non-trivial examples. In this approach, we first procedurally create data instances using a set of sentence transformations in an automated way and train a model using these synthetically created examples. Then, for collecting the non-trivial examples, we explore an adversarial technique of data collection. Specifically, the data creators need to create a data instance on which the trained model fails to give the correct prediction. Thus, it discourages the data creators from creating trivial examples as the model is already trained on such trivial examples and will not get fooled. This explore several unsupervised settings and show that our approach achieves state-of-the-art results. This work makes the crucial step of data collection efficient and also improves the quality of the collected data (Varshney et al. 2022). After the completion of training, we focus on improving the efficiency of evaluations in the next chapter.

Improving Efficiency of Evaluations: Success of transformer-based models such as BERT (Devlin et al. 2019) has fostered development of a number of other large-scale pre-trained language models (Y. Liu et al. 2019b; K. Clark et al. 2020; Sanh et al. 2019). Furthermore, there are numerous different pre-training objectives. So, essentially, we have a lot of model options and comparing the performance of such as a large number of models is very expensive. In Chapter 11, we explore the question "can we efficiently compare the performance of models?" Addressing this question, we propose an approach of difficulty sampling in which we first argue that not all the evaluation examples are equally difficult, then we develop a method to quantify the difficulty of an instance, and finally, we propose an approach to sample examples based on the difficulty scores. This work is one of the pioneering works in this important area of efficient evaluations.

In conclusion, this dissertation provides effective solutions to address the critical challenges pertinent to the efficiency and reliability of NLP systems.

1.3 Impact of our Solutions

Addressing the Hallucination Problem of LLMs: On the article generation task, our active detection and mitigation approach successfully reduces the hallucinations from 47.5% to 14.5%. We also show the individual efficacy of our detection and mitigation techniques. Specifically, the detection technique achieves a recall of ~ 88% and the mitigation technique successfully mitigates 57.6% of the correctly detected hallucinations. Importantly, our mitigation technique does not introduce new hallucinations even in the case of incorrectly detected hallucinations, i.e., false positives. We also show that our approach can be adapted to improve the performance on multi-hop bridge questions where the percentage hallucinations reduces from 54% to just 26%. We also demonstrate its efficacy on the adversarial false premise question answering task where the hallucination reduces to just 24%.

Reducing Prediction Errors via Selective Prediction: Our calibration technique results in an improvement in selective prediction performance of 15.81% and 5.64% in the in-domain and out-of-domain settings, respectively.

Increasing the Coverage of a Selective Prediction System via Post-Abstention Strategies: The post-abstention method results in overall risk improvements of up to 21.81 and 24.23 in the in-domain and out-of-domain settings respectively.

Jointly Improving Inference Efficiency and Accuracy: We conduct comprehensive experiments with 10 diverse NLU datasets in multiple task settings that differ in the number of models available for cascading (K value from Section 8.2). We first demonstrate that cascading achieves considerable improvement in computational efficiency. For example, in case of QQP dataset, cascading system achieves 88.93% computation improvement over the largest model (M_3) in K=3 setting i.e. it requires just 11.07% of the computation cost of model M_3 to attain equal accuracy. Then, we show that cascading also achieves improvement in prediction accuracy. For example, on CB dataset, the cascading system achieves 2.18% accuracy improvement over M_3 in the K=3 setting. Similar improvements are observed in settings with different values of K. We also show that introducing additional model in the cascade further increases the efficiency benefits.

Improving QA Reader Efficiency while Leveraging External Knowledge: Comparing with the top-performing Fusion-in-Decoder reader, our dynamic reading approach matches its accuracy by utilizing just 18.32% of its reader inference cost (measured in FLOPs) and also outperforms it by achieving up to 55.10% accuracy on NQ Open.

Improving Data Sample Efficiency of Training: Comprehensive experiments with several NLI datasets show that the proposed approach results in accuracies of up to 66.75%, 65.9%, 65.39% in PH, P, and NPH settings respectively, outperforming all existing unsupervised baselines. Furthermore, fine-tuning our model with as little as 0.1% of the human-annotated training dataset (500 instances) leads to 12.2% higher accuracy than the model trained from scratch on the same 500 instances.

Improving Efficiency of Evaluations: Evaluation conducted using instances selected based on difficulty scores shows high correlation with the evaluation conducted using the entire dataset. Specifically, using just 5% instances (selected via difficulty analysis) achieves as high as 0.93 Kendall correlation with evaluation using complete dataset.

1.4 Summary

A summary of the main contributions of this thesis is provided below:

- In Chapter 2, we present the background and the related work of our research problem. Specifically, we first describe the hallucination problem of language models with various causes and approaches to address this phenomenon. Then, we detail the selective prediction task and also provide the mathematical formulation of the task involving coverage and risk. Finally, we provide background for the efficiency challenges.
- In Chapter 3, we address the problem pertaining to the hallucination of language models. We highlight how hallucinations hamper the reliability and trustworthiness of the model and then describe our approach to address the problem. Specifically, our approach actively detects and mitigates the hallucinations during the generation process using a retrieval augmented methodology. We conduct comprehensive experiments and demonstrate the efficacy of the approach across various tasks.
- The work in Chapter 3 focuses on general tasks; however, an important component of language is 'negation'. Negation is important because it adds depth and nuance to the understanding of language. To this end, in Chapter 4, we extend our research on hallucinations and further study it in tasks involving negation. Specifically, we study four tasks with negation: 'false premise completion', 'constrained fact generation', 'multiple choice question answering', and 'fact generation'. We show that open-source state-of-the-art LLMs such

as LLaMA-2-chat, Vicuna, and Orca-2 hallucinate considerably on all these tasks involving negation which underlines a critical shortcoming of these models. Addressing this problem, we further study numerous strategies to mitigate these hallucinations and demonstrate their impact.

- In Chapter 5, we note that the reliability can also be improved via selective prediction in which we enable the systems to abstain from making predictions when they are likely to be incorrect. We underline the advantages of selective prediction and study various approaches such as maximum softmax probability, monte-carlo dropout, label smoothing, and calibration in in-domain, out-of-domain, and adversarial settings.
- In Chapter 6, we note that while the selective prediction approaches are advantageous, they leave us with a pertinent question '*what to do after abstention*'. To this end, we study this question with 'Post-Abstention', a task that allows re-attempting the abstained instances with the aim of increasing *coverage* of the system without significantly sacrificing its *accuracy*. We first provide mathematical formulation of this task and then explore several methods to solve it. Then, we turn our focus to jointly improving the efficiency and accuracy.
- In Chapter 7, we address the high inference cost of LLMs and propose a dynamic confidence based early exiting approach that improves the computational efficiency of the text generation without sacrificing the quality of the generation. We present comprehensive experiments by instruction tuning LLaMA-2 models on the Alpaca dataset and demonstrate efficacy on four different instruction test sets.

- In Chapter 8, we present model cascading that focuses on efficiently yet accurately output predictions. We show the efficacy of cascading on various classification tasks. We also study the impact of introducing additional models in the cascade and show that it further increases the efficiency improvements.
- In Chapter 9, we adapt this cascading technique for open-domain question answering task and improve the efficiency of the reader model in the retriever-reader pipeline. Specifically, we present a dynamic reading approach that efficiently leverages the 'closed-book' and 'open-book' inferences.
- In Chapter 10, we focus on synthetically creating task specific training data to reduce the requirement of collecting expensive crowdsourced data. We present this work for the natural language inference task which requires determining whether a hypothesis is true (Entailment), false (Contradiction), or undetermined (Neutral) given a premise statement.
- In Chapter 11, we cover instance-level difficulty analysis of evaluation data. This is motivated from the fact that not all the evaluation instances are equally difficult, i.e., some instances are trivially easy, some are moderately difficult, and some are extremely difficult. Thus, we propose a method to calculate the difficulty score of an instance and demonstrate various applications of these scores such as conducing efficient yet accurate evaluations withe fewer instances.

1.5 Related Publications

The ideas in this dissertation have appeared in the following papers:

- Neeraj Varshney and Chitta Baral. 2023. Post-Abstention: Towards Reliably Re-Attempting the Abstained Instances in QA. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 967–982, Toronto, Canada. Association for Computational Linguistics.
- Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022. ILDAE: Instance-Level Difficulty Analysis of Evaluation Data. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 3412–3425, Dublin, Ireland. Association for Computational Linguistics.
- Neeraj Varshney and Chitta Baral. 2022. Model Cascading: Towards Jointly Improving Efficiency and Accuracy of NLP Systems. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11007–11021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Neeraj Varshney, Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. 2022. Unsupervised Natural Language Inference Using PHL Triplet Generation. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2003–2016, Dublin, Ireland. Association for Computational Linguistics.
- Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. 2022. Investigating Acceleration of LLaMA Inference by Enabling Intermediate Layer Decoding via Instruction Tuning with 'LITE'. In Findings of the Association for

Computational Linguistics: NAACL 2024, Mexico City, Mexico. Association for Computational Linguistics.

- Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022. Investigating Selective Prediction Approaches Across Several Tasks in IID, OOD, and Adversarial Settings. In Findings of the Association for Computational Linguistics: ACL 2022, pages 1995–2002, Dublin, Ireland. Association for Computational Linguistics.
- Neeraj Varshney, Himanshu Gupta, Eric Robertson, Bing Liu, and Chitta Baral. 2023. A Unified Evaluation Framework for Novelty Detection and Accommodation in NLP with an Instantiation in Authorship Attribution. In Findings of the Association for Computational Linguistics: ACL 2023, pages 1794–1818, Toronto, Canada. Association for Computational Linguistics.
- Neeraj Varshney, Swaroop Mishra, and Chitta Baral. 2022. Towards Improving Selective Prediction Ability of NLP Systems. In Proceedings of the 7th Workshop on Representation Learning for NLP, pages 221–226, Dublin, Ireland. Association for Computational Linguistics.
- Neeraj Varshney, Man Luo, and Chitta Baral. 2023. Can Open-Domain QA Reader Utilize External Knowledge Efficiently like Humans? Workshop on Knowledge Augmented Methods for NLP, Washington, DC, USA. The Association for the Advancement of Artificial Intelligence (AAAI).
- Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, Dong Yu. 2023. A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation, Preprint.

Chapter 2

BACKGROUND AND RELATED WORK

2.1 Hallucinations of Large Language Models

Advancements in the field of natural language processing have led to the development of models that possess an impressive ability to generate fluent and coherent text. However, these models are vulnerable to hallucinate in their output. Prior work (Maynez et al. 2020; Huang et al. 2021; Ji et al. 2023) has categorized text hallucinations into two classes: Intrinsic (when the generated output contradicts the source content) and Extrinsic (when the generated output cannot be verified from the source content, i.e., it that can neither be supported nor contradicted by the source).

One thread of research pertaining to hallucinations has focused on studying different causes of this phenomenon such as **training data quality** (Wang 2019; Katherine Lee et al. 2022), **source-target divergence** (Dhingra et al. 2019) (when a model is trained on noisy data with source-reference divergence, it may learn to generate text that is not necessarily grounded or faithful to the given source), **ill-suited modeling** (Aralikatte et al. 2021; Feng et al. 2020; H. Li et al. 2018), **stochasticity during inference** (Dziri et al. 2021; Tian et al. 2019; N. Lee et al. 2022) (decoding strategies that improve the generation diversity, such as top-k sampling, top-p, and temperature parameters, often result in increased hallucinations which could be attributed to the introduction of "randomness/stochasticity" while selecting tokens (from top-k or top-p) instead of choosing the most probable token while decoding), and **parametric knowledge bias** (Longpre et al. 2021; W. Zhou et al. 2023; Michel et al. 2019) in which Models often tend to prioritize the parametric knowledge (knowledge acquired during pre-training and implicitly stored in the parameters of the model) over the provided contextual knowledge resulting in hallucinations.

The other thread focuses on addressing the hallucination problem (Manakul, Liusie, and M. J. Gales 2023; Azaria and Mitchell 2023b; N. Lee et al. 2022; Du et al. 2023; T. Zhang et al. 2023). Manakul, Liusie, and M. J. Gales 2023 proposes a sampling-based hallucination detection approach in which they first sample multiple responses from the model and then measure the information consistency between the different responses. They posit that when a language model knows a given concept well, the sampled responses are likely to be similar and contain consistent facts; on the other hand, for hallucinated facts, stochastically sampled responses are likely to diverge and may completely contradict one another.

Azaria and Mitchell 2023b leverages LLM's internal state to identify the truthfulness of a statement. Using an annotated dataset, they train a separate classifier that takes the LLM's activation values as input and predicts its truthfulness. Kadavath et al. 2022 have shown the utility of model's uncertainty values in detecting incorrectness in the model's responses by demonstrating that larger models are well-calibrated on multiple-choice and true/false questions. N. Lee et al. 2022 hypothesize that the randomness of sampling is more harmful to factuality when it is used to generate the latter part of a sentence than the beginning of a sentence and propose a new sampling algorithm named factual-nucleus sampling that dynamically adapts the 'nucleus' p along the generation of each sentence.

Du et al. 2023 propose an approach motivated by *The Society of Mind* and *multi-agent settings* in which multiple models individually propose and jointly debate their responses and reasoning processes to arrive at a common answer.

Gou et al. 2023; A. Chen et al. 2023; Zhao et al. 2023; Chern et al. 2023 propose to use external knowledge/tools to address the hallucination problem of LLMs. FactScore Min et al. 2023 presents an evaluation method that breaks the model's generation into a series of atomic facts and computes the percentage of atomic facts supported by a reliable knowledge source. A. Chen et al. 2023 finetunes a T5-large model as compact editor to denoise the corruptions to detect incorrectness in a given sentence.

We also study hallucinations of LLMs in tasks involving negation. This is because the above works investigating hallucinations lack comprehensively studying the crucial aspect of 'negation'. Negation is important because it adds depth and nuance to the understanding of language. It helps understand the opposite or absence of a statement, providing a more precise and nuanced interpretation and it is also crucial for logical reasoning and inference. Negation also helps prevent misinterpretation of statements, i.e., without the ability to recognize negation, one might misunderstand the intended meaning of a sentence, leading Furthermore, we humans arguably use affirmative expressions (without negation) more often than expressions with negation (Hossain et al. 2020; Ettinger 2020); this implies that texts containing negation could be underrepresented in the training/tuning data of the models making it even more important to study.

Prior studies on negation have primarily focused on classification tasks like natural language inference and masked word prediction. Hosseini et al. 2021 propose to fine-tune BERT (Devlin et al. 2019) with an unlikelihood objective and evaluate on negated LAMA dataset (Kassner and Schütze 2020). Hossain et al. 2020; Hossain and Blanco 2022; Truong et al. 2023 focus on natural language inference and cloze completion tasks with datasets like SNLI (Bowman et al. 2015), Multi-NLI (Williams, Nangia, and Bowman 2018). Ye et al. 2023 have studied negation in logical reasoning context. A more recent work Jang, Ye, and Seo 2023 study the performance of LLMs on transformed prompts of various datasets where the transformation is performed by replacing words like 'correct' with 'incorrect', 'appropriate' with 'inappropriate', and 'natural' with 'unnatural'. This transformation results in prompts such as "Complete the given sentence with the *inappropriate* ending".

2.2 Selective Prediction

2.2.1 Formulation

A selective prediction system comprises of a predictor (f) that gives the model's prediction on an input (x), and a selector (g) that determines if the system should output the prediction made by f i.e.

$$(f,g)(x) = \begin{cases} f(x), & \text{if g(x) = 1} \\ Abstain, & \text{if g(x) = 0} \end{cases}$$

Usually, g comprises of a confidence estimator $\tilde{g}$ that indicates $f's$ prediction confidence and a threshold th that controls the abstention level:

$$g(x) = \mathbb{1}[\tilde{g}(x)) > th]$$

An SP system makes trade-offs between *coverage* and *risk*. For a dataset D, coverage at a threshold th is defined as the fraction of total instances answered by the system (where $\tilde{g} > th$) and risk is the error on the answered instances:

$$coverage_{th} = \frac{\sum_{x_i \in D} \mathbb{1}[\tilde{g}(x_i)) > th]}{|D|}$$

$$risk_{th} = \frac{\sum_{x_i \in D} \mathbb{1}[\tilde{g}(x_i)) > th] l_i}{\sum_{x_i \in D} \mathbb{1}[\tilde{g}(x_i)) > th]}$$

where, l_i is the error on instance x_i.

With decrease in th, coverage will increase, but the risk will usually also increase. The overall SP performance is measured by the *area under Risk-Coverage curve* (El-Yaniv et al. 2010) which plots risk against coverage for all threshold values. **Lower the AUC, the better the SP system** as it represents lower average risk across all confidence thresholds. We note that 'confidence calibration' and 'OOD detection' are related tasks but are non-trivially different from selective prediction as detailed in 2.2.2.

2.2.2 Tasks Related to Selective Prediction

2.2.2.1 Confidence Calibration

Selective Prediction is closely related to *confidence calibration* (Platt et al. 1999) i.e aligning model's output probability with the true probability of its predictions. Calibration focuses on adjusting the overall confidence level of a model, while selective prediction is based on relative confidence among the examples i.e systems are judged on their ability to rank correct predictions higher than incorrect predictions.

2.2.2.2 Out-of-Domain Detection

Using OOD Detection systems for selective prediction (abstain on all detected OOD instances) would be too conservative as it has been shown that models are able to correctly answer a significant fraction of OOD instances (Talmor and Berant 2019; Hendrycks et al. 2020; Mishra, Arunkumar, Bryan, et al. 2020).

2.3 Efficient Decoding of Large Language Models

Improving the inference efficiency of LLMs is an important research direction and is receiving considerable attention from the NLP community. In this section, we review some of the existing methods and differentiate our work from them.

Reducing model size: Since model size plays a crucial role in increasing the inference cost and latency, techniques like **quantization** (Dettmers et al. 2022; Yao et al. 2022; Frantar et al. 2023), **knowledge distillation** (Hsieh et al. 2023; Jiao et al. 2020; Z. Li et al. 2022), **model compression and network pruning** (Wang, Wohlwend, and Lei 2020; Guo, Rush, and Kim 2021) have been shown to be effective in improving the inference efficiency.

Furthermore, during sampling, a cache of the keys and values can be maintained for every attention layer which reduces the computations at inference time (**KV caching**). However, it increases the GPU VRAM memory requirement of inference.

Another technique **speculative sampling** (Leviathan, Kalman, and Matias 2023; C. Chen et al. 2023) first generates a draft of K tokens from a smaller auto-regressive model and then scores the draft using the target model. This results in generation of more than one token (on average) from the target model in a single pass.

Early exiting and cascading based inference techniques have been shown to be effective for classification tasks with BERT-style models, such as DeeBERT (Xin et al. 2020) that speeds up BERT inference by inserting extra classification layers between each encoder layer, PoWER-BERT (Goyal et al. 2020) that focuses on progressive word-vector elimination (based on significance computed using self-attention) along the encoder pipeline, DynaBERT (Hou et al. 2020) that adjusts the size of the model by selecting adaptive width and depth, and cascading (Varshney and Baral 2022; Lei Li et al. 2021; Yue et al. 2023; Cheng, Kasai, and Yu 2023) in which sequential inference is done through models of bigger and bigger size with conditional exiting to output predictions efficiently. Our work is also related to Confident Adaptive Language Modeling (CALM) (Schuster et al. 2022) and Depth-

Adaptive Transformers (Elbayad et al. 2020) in which early exiting is performed by learning additional classifiers attached to the decoder layers.

Din et al. 2023 proposed to short-cut away transformer inference in between certain layers by learning linear transformations across layers in the network, i.e., **casting internal representations**. O'Brien and Lewis 2023; Gera et al. 2023 explore leveraging the intermediate layers for contrastive decoding to improve reasoning.

S. Yang et al. 2023 propose an approach called Predictive Pipelined Decoding (PPD) that focuses on lowering the latency by utilizing additional compute resources. Specifically, it accelerates the decoding by parallelizing processes, each of which starts to decode from the top-k predicted tokens of the specific transformer layer. Simultaneously, the main process continues to compute the output of the final layer and predicts the next token.

N. Yang et al. 2023 propose an inference-with-reference decoding method that exploits the overlap between an LLM's output and available reference. Specifically, it first selects a text span from the reference and copies its tokens to the LLM decoder and then checks if they are acceptable based on the output token probabilities. He et al. 2023 also uses retriever to generate the draft tokens in speculative decoding.

H. Jiang et al. 2023 propose to compress the prompts to accelerate the inference. Santilli et al. 2023 propose a parallel decoding strategy for translation using Jacobi and Gauss-Seidel fixed-point iteration methods

2.4 Efficient Inference of Models

In recent times, several techniques have been developed to improve the efficiency of NLP systems, such as **network pruning** (Wang, Wohlwend, and Lei 2020; Guo, Rush, and Kim 2021; T. Chen et al. 2020), **quantization** (Shen et al. 2020; W. Zhang et al. 2020; Tao et al. 2022), *knowledge distillation* (K. Clark et al. 2019; Jiao et al. 2020; Z. Li et al. 2022; Mirzadeh et al. 2020), and **input reduction** (Modarressi, Mohebbi, and Pilehvar 2022). Our work is more closely related to dynamic inference (Xin et al. 2020) and adaptive model size (Goyal et al. 2020; Kim and Cho 2021; Hou et al. 2020; Soldaini and Moschitti 2020).

Xin et al. 2020 proposed Dynamic early exiting for BERT (DeeBERT) that speeds up BERT inference by inserting extra classification layers between each transformer layer. It allows an instance to choose conditional exit from multiple exit paths. All the weights (including newly introduced classification layers) are jointly learnt during training.

Goyal et al. 2020 proposed Progressive Word-vector Elimination (PoWER-BERT) that reduces intermediate vectors computed along the encoder pipeline. They eliminate vectors based on significance computed using self-attention mechanism. Kim and Cho 2021 extended PoWER-BERT to Length-Adaptive Transformer which adaptively determines the sequence length at each layer. Hou et al. 2020 proposed a dynamic BERT model (DynaBERT) that adjusts the size of the model by selecting adaptive width and depth. They first train a width-adaptive BERT and then distill knowledge from full-size models to small sub-models.

Lastly, cascading has been studied in machine learning and vision with approaches such as Haar-cascade (Soo 2014) but is underexplored in NLP (Lei Li et al. 2021). We further note that cascading is non-trivially different from 'ensembling' as ensembling always uses all the available models instead of carefully selecting one or more models for inference.

In open-domain QA, especially for the retriever-reader systems, efficiency from the perspectives of retrieval (Zhao, Lu, and Lee 2021; Luo et al. 2022) and on-disk memory (Min et al. 2021; Izacard et al. 2020) has been explored. However, efficiently leveraging external knowledge to improve the computation efficiency of reader inference is also important and has remained underexplored.

2.5 Efficient Evaluations

Adaptive evaluation (Weiss 1982) is used in educational settings for evaluating performance of students. It uses Item Response Theory (IRT) (Baker and Kim 2004) from psychometrics that requires a large number of subjects and items to estimate system parameters (Lalor, Wu, and Yu 2016; Lalor et al. 2018). Moreover, adaptive evaluation is computationally very expensive as it requires calculating performance after each response to select the next instance based on the previous responses of the subject. Thus, it is not fit for our setting as we intend to improve the computational efficiency. In contrast, our approach is much simpler and efficient as it does not incur any additional cost during the evaluation.

Chapter 3

IMPROVING RELIABILITY BY ACTIVELY DETECTING AND MITIGATING HALLUCINATIONS OF LARGE LANGUAGE MODELS

Recently developed large language models (LLMs) have achieved remarkable success in generating fluent and coherent text. However, these models often tend to 'hallucinate' which critically hampers their reliability. Hallucination in the context of language models refers to the generation of text that seems syntactically sound, fluent, and natural but is factually incorrect, nonsensical, or unfaithful to the provided source input (Maynez et al. 2020; Holtzman et al. 2020; Ji et al. 2023). Hallucinations hamper the reliability and trustworthiness of the model which makes it important to address this problem.

In this chapter, we address this crucial problem and propose an approach that actively detects and mitigates hallucinations during the generation process. Specifically, we first identify the candidates of potential hallucination leveraging the model's *logit output values*, check their correctness through a *validation* procedure, mitigate the detected hallucinations via *prompting*, and then continue with the generation process. This active intervention also facilitates in preventing the propagation of hallucinations in the LLM's output. Through extensive experiments with GPT-3.5 (text-davinci-003) on the 'article generation task', we first show that the proposed approach successfully reduces the hallucinations from 47.5% to 14.5%. Then, we further demonstrate the effectiveness and wide applicability of our approach through additional experiments with different types of questions (multi-hop and false premise) and with another LLM from a different model family (Vicuna). In summary, this work contributes to improving the reliability and trustworthiness of LLMs, a crucial step en route to enabling their widespread adoption.

3.1 Why Do LLMs Hallucinate?

Recent research on LLM hallucinations has uncovered a variety of reasons for this phenomenon:

Source-Reference Divergence: When a model is trained on data with source-reference (target) divergence, it may learn to generate text that is not necessarily grounded or faithful to the given source. While collecting the data, source-reference divergence can happen unintentionally or intentionally.

Unintentional Source-Reference Divergence: It is possible that the data is heuristically created and the target may contain information that is not always supported by the source. For instance, if you take news about an incident from two different websites as a source-reference pair then the reference may contain information that is absent in the source causing the divergence.

Intentional Source-Reference Divergence: Some tasks by nature do not always demand information alignment between the source and the target, especially those that value diversity in the generated output.

Another factor is the **presence of duplicates in the training corpus**. Specifically, duplicated examples in the training corpus can bias the model towards generating some highly frequent tokens/phrases causing hallucination.

Using such noisy data for training is one of the factors contributing to the hallucination phenomenon.

Stochasticity in Decoding Technique: It has been shown that decoding strategies that improve the generation diversity, such as top-k sampling, top-p, and temperature parameters, often result in increased hallucinations. This could be attributed to the introduction of "randomness/stochasticity" while selecting tokens (from top-k or top-p) instead of choosing the most probable token while decoding.

Parametric Knowledge Bias: Models have been shown to often prioritize the parametric knowledge (knowledge acquired during pre-training and implicitly stored in the parameters of the model) over the provided contextual knowledge resulting in hallucinations.

Discrepancy between Training-time and Inference-time Decoding: One of the common ways of training a model uses teacher-forced maximum likelihood estimation (MLE) method, where the decoder is encouraged to predict the next token conditioned on the ground-truth prefix sequences. However, at inference time, the model generates the next token conditioned on the historical sequences previously generated by the model itself. Such a discrepancy can lead to hallucinated generation, especially when the target sequence becomes long.

Also, generative LMs like GPT-3 are trained to model the statistical correlations between subword tokens, and thus in reality they can only acquire a limited capability to generate factually accurate text.

3.2 Active Detection and Mitigation Approach

In this work, we propose to actively 'detect' and 'mitigate' hallucinations during the generation process. This is crucial as we will show that when a sentence generated by a model is hallucinated, the chances of hallucination in the subsequently generated sentences increase, i.e., hallucinations often propagate in the model's output. This can be attributed to the autoregressive nature of the LLMs and the discrepancy between the training and inference time decoding. Specifically, during the training time, the model is encouraged to predict the next token conditioned on the ground-truth prefix tokens. However, at inference time, the model generates the next token conditioned on the historical tokens previously generated by itself. Thus, actively detecting and mitigating hallucinations during the generation process also facilitates in preventing the propagation of hallucinations in the generation.

We divide our approach into two stages: **Detection** and **Mitigation**. Figure 1 illustrates the key steps of our approach. In order to address the complex task of detecting and mitigating hallucinations, we decompose it into multiple simpler steps.

In the hallucination detection stage (Section 3.2.1), we first identify the candidates of potential hallucination, i.e., the key 'concepts' of the generated sentence. Next, leveraging the logit output values of the model, we calculate model's 'uncertainty' on the identified concepts. We demonstrate that this uncertainty score provides a signal for hallucination. However, we note that this is an additional signal and not a necessary requirement for our approach. Then, we check the correctness of the 'uncertain' concepts through a *validation* procedure (where we retrieve the relevant

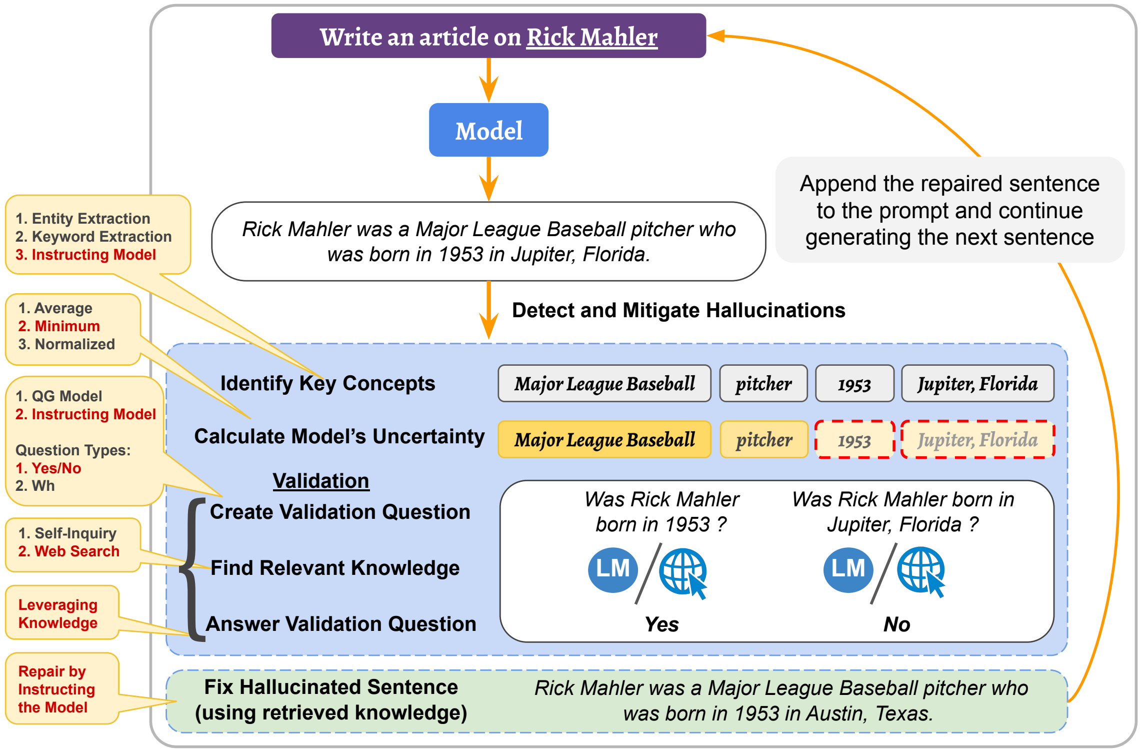


Figure 1. Illustration of the proposed active detection and mitigation approach. Different techniques for each step are mentioned on the left with the preferred technique highlighted in red.

knowledge) to detect hallucinations. This is followed by hallucination mitigation (Section 3.2.2) where we 'repair' the sentence via prompting using the retrieved knowledge as evidence. We can also utilize this knowledge as context for subsequent generation. We conduct a systematic study exploring multiple techniques for each step of the approach (as shown in Figure 1).

3.2.1 Hallucination Detection

3.2.1.1 STEP: Identify Key Concepts

We start by identifying the candidates of potential hallucination, i.e., the important concepts from the generated sentence. We identify these concepts because validating the correctness of the entire sentence at once is infeasible as a sentence often contains multiple different facets all of which can not be validated at once. In contrast, individually validating correctness corresponding to the concepts provides opportunities for accurately detecting hallucinations. Note that a concept is essentially a span of text consisting of one or more tokens. We study the following techniques for this step:

Entity Extraction: Entities are typically important parts of a sentence, thus, we explore using an off-the-shelf entity extraction model to identify the concepts. A limitation of this method is that a concept need not necessarily be an entity.

Keyword Extraction: Addressing the above limitation and additionally identify the non-entity concepts, we explore using an off-the-shelf keyword extraction model. For keyword extraction, we explore a model[1] that uses Keyphrase Boundary Infilling with Replacement (KBIR) as its base model and is fine-tuned on the KPCrowd dataset (Kulkarni et al. 2021).

***Instructing the Model*:** Since state-of-the-art LLMs perform remarkably

[1] https://huggingface.co/ml6team/keyphrase-extraction-kbir-kpcrowd

Text	Entity Extraction	Keyword Extraction	Instructing Model
John Russell Reynolds was an English physician and neurologist who made significant contributions to the field of neurology.	John Russell Reynolds, English	John Russell Reynolds, English, physician, neurologist, significant contributions, field, neurology	John Russell Reynolds, English, physician, neurologist, neurology
He was born in London in 1820 and studied medicine at the University of London.	London, 1820, the University of London	born, London, 1820, studied medicine, University, London	London, 1820, medicine, University of London
After college, he worked as a lawyer for the PGA Tour, eventually becoming the Tour's Deputy Commissioner in 1989.	the PGA Tour, Tour, 1989	college, worked, lawyer, PGA, Tour, eventually, Tour, Deputy Commissioner	college, lawyer, PGA Tour, Deputy Commissioner, 1989

Table 1. Examples of concepts identified by different techniques.

well on a wide range of tasks, in this technique, we directly instruct the model to identify the important concepts from the generated sentence.

Table 1 shows examples of concepts identified using the three methods, i.e., Entity Extraction, Keyword Extraction, and Instructing the Model. It shows that the entity extraction model misses many important concepts while the keyword extraction model identifies a lot of insignificant concepts also. In contract, instruction technique successfully identifies all the important concepts. Moreover, it doesn't require calling a task-specific tool (entity or keyword extraction model). Thus, we regard it as our preferred technique for this step.

3.2.1.2 STEP: Calculate Model's Uncertainty

LLMs also provide logit values in their output. Thus, we study if these values can be utilized to detect hallucinations. Consider a concept consisting of n tokens and having the maximum softmax probabilities as $p_1, p_2, p_3, ..., p_n$ for the n token positions. We study three different techniques for calculating a **probability score** for a concept:

Average (AVG $[p_1, p_2, ..., p_n]$) , **Normalized Product** ($[p_1 \times p_2 \times ... \times p_n]^{1/n}$) , and ***Minimum*** (MIN $[p_1, p_2, ..., p_n]$) . Here, 'MIN' is our preferred technique as the others may average out the effect of model's uncertainty on the tokens while low probability in even one token of a concept provides sufficient evidence of the model being uncertain in its generation. For e.g., if the model is uncertain about name of the USA president then its uncertainty on the first token ('Joe') would be high but on the next token ('Biden') would be very low as token 'Joe' is frequently followed by token 'Biden' in raw text. Thus, Averaging or Normalizing the probabilities will have a limited capability to capture this signal in comparison to Minimum.

In 3.3.1.2, we show that this score (especially 'MIN') indeed provides a signal for hallucination, i.e., the more uncertain the model is on a concept (low probability score), the more likely it is to be hallucinating about that concept. calculation techniques. Thus, we utilize this signal and check for hallucinations for the uncertain concepts using our validation procedure (3.2.1.3-3.2.1.5). Figure 11 compares the performance of the three probability calculation techniques.

In the absence of logit output values, all or some heuristically selected concepts (depending on the computational and latency budget of the system) can be passed to the validation stage for detecting hallucinations.

3.2.1.3 STEP: Create Validation Question

Our validation procedure for a concept starts with creation of a question that tests the correctness of the information (in the generated sentence) pertaining to the concept. We study creating **Yes/No Questions** as illustrated in Table 2 using **Question Generation Tool** and ***Instructing the Model***.

In instruction technique, we directly prompt the model to create a validation question checking the correctness of the information about the selected concept. Similar to the concept identification step, it is our preferred technique as it does not require calling a task-specific tool. We note that instead of Yes/No questions, **Wh-questions** can also be used for validation. We prefer Yes/No questions as it is relatively easier to verify their answers. We explore Wh-questions for a study in Section 3.4.2.

Table 2 shows examples of validation questions corresponding to each concept created via the instruction technique. It shows examples of both the question types, i.e., Yes/No and Wh questions. We prefer Yes/No questions as it is relatively easier to verify the answer of these questions.

We have also conducted evaluations of the efficacy of the instructions. Specifically, for the concept identification step, we studied randomly sampled 50 sentences. The instruction technique identified 155 concepts in total. It missed only 2 concepts (that

Input	Generated Sentence	Concept	Validation Question
Write an article about John Russell Reynolds	Reynolds was born in **London** in **1820** and studied **medicine** at the **University of London**.	**London**	[Y/N] Was John Russell Reynolds born in London? [Wh] Where was John Russell Reynolds born?
		1820	[Y/N] Was John Russell Reynolds born in 1820? [Wh] What year was John Russell Reynolds born?
		medicine	[Y/N] Did John Russell Reynolds study medicine? [Wh] What did John Russell Reynolds study at the University of London?
		University of London	[Y/N] Did Reynolds study medicine at the University of London? [Wh] What university did John Russell Reynolds study medicine at?

Table 2. Examples of validation questions corresponding to the identified keyphrases generated by Instructing the Model technique.

too these missed concepts can only be loosely regarded as important in the context of the sentence). Furthermore, the efficacy of the validation and mitigation instructions is presented in Table 6 and 7, respectively.

We note that the overall efficacy of these techniques (and how well they serve their purpose) is evaluated by the overall improvement in reducing the hallucinations. We also note that the LLM can be prompted in a different way also to achieve the same objective; however, the purpose of this work is to show that the complex task of addressing hallucinations in an end-to-end manner can be decomposed into simpler steps that can be solved via instructing the model.

3.2.1.4 STEP: Find Relevant Knowledge

We explore two ways of retrieving the relevant knowledge to answer the validation question.

***Web Search*:** Web search provides several benefits such as generality, wide coverage, and information freshness. We use Bing search API for retrieving the knowledge. However, we note that any other search API or knowledge corpus can also be utilized for this purpose.

Self-Inquiry: Here, we leverage the parametric knowledge of the LLM and directly prompt it to answer the validation question. Though it does not require external knowledge, it has drawbacks such as lack of a reliable strategy to extract the parametric knowledge and knowledge staleness.

Note that our proposed approach has several benefits pertaining to retrieval: (a) it does not retrieve knowledge when it is not required, i.e., when the model is already sufficiently confident (since we show that it is less likely to hallucinate in such scenarios), (b) it individually retrieves knowledge pertinent to the concept(s) on which the calculated probability score is low thus providing it sufficient and relevant context for accurate validation and mitigation (Section 3.2.2).

3.2.1.5 STEP: Answer Validation Question

Now, we prompt the model to answer the validation question leveraging the retrieved knowledge as context and verify its response. If the validation procedure succeeds for all the uncertain concepts then we continue generating the next sentence; otherwise, we interrupt the generation and mitigate the potential hallucination in the sentence before continuing the subsequent generation.

3.2.2 Hallucination Mitigation

For mitigating hallucination in the generated sentence, we instruct the model to repair the generated sentence by removing/substituting the hallucinated information and incorporating the correct information using the retrieved knowledge as evidence (Table 3 shows the instructional prompt).

We note that the result of our validation procedure is contingent on the retrieved knowledge and the model's ability to leverage that knowledge in answering the validation question. In Section 3.3.2, we show that our approach performs well on this task and achieves a high recall demonstrating its efficacy at detecting hallucinations. Moreover, we show that our mitigation approach does not introduce new hallucinations even in the case of incorrectly detected hallucinations (false positives).

Table 3 shows the instructional prompts used for different steps of the approach.

3.2.3 Design Decisions

Why the task of addressing hallucinations is broken down into multiple steps? We note that dealing with the hallucination problem is a complex task and prior work has shown that breaking down a complex task into simpler sub-tasks helps

Step	Prompt
Input Prompt	`Write an article about {topic}`
Identify Important Concepts	`Identify all the important keyphrases from the above sentence and return a comma separated list.`
Create Validation Question	`For the above sentence about {topic}, generate a yes/no question that tests the correctness of {concept}.`
Answer Validation Question	`{search results} Answer the below question about topic in Yes or No based on the above context. {validation question}.`
Repair Hallucinated Sentence	`The above sentence has information that can not be verified from the provided evidence, repair that incorrect information and create a new sentence based on the provided evidence.`

Table 3. Instructional Prompts corresponding to different steps of our approach.

the model in solving the task better and achieve higher performance (Wei et al. 2022; D. Zhou et al. 2023; Khot et al. 2023). Thus, we break down this task into individual sub-tasks which are considerably easier for the model. For the same reason, we also break down the validation procedure into several steps. We also note that creating multiple steps can increase the chances of propagation of error from one to the other; however, the individual steps in our approach are very simple, and the models perform remarkably well on these steps.

Why validation is done using the web search? Our preferred technique for retrieving knowledge is web search because the web is more likely to contain the updated knowledge in comparison to a knowledge corpus whose information can become stale, outdated, and obsolete.

Why "active" detection & mitigation and not "post-hoc" after complete response generation? We note that our detection and mitigation techniques can also be applied in a "posthoc" manner after complete response generation. However, it has several limitations which are addressed by our "active" approach. The "active" approach prevents the propagation of hallucinations in the subsequently generated sentences, i.e., if hallucination is detected in the initially generated sentences then it would be mitigated and course correction would be done for the subsequently generated sentences. However, the "post-hoc" approach does not provide such an opportunity of course correction. In other words, in the "active" approach, the model sees the mitigated / corrected sentences while generating the subsequent sentences; thus, its output will be more correct, coherent, and fluent. In contrast, in the "posthoc" approach, the generated sentences are based on the initially generated previous sentences and thus the mitigated sentence will not be able to influence the generation of subsequent sentences; thus, the output would not be as coherent and fluent as the active approach.

Also, applying it in a post-hoc manner will fix the sentences individually thus, redundant information could be present in multiple sentences hampering the quality of the response.

For example, for the topic "Twila Shively", the model generated "*Twila Shively is a renowned American artist and sculptor who has been creating art for over four decades. She is best known for her large-scale sculptures, which often feature abstract shapes and forms. ...*" which is completely hallucinated.

After applying our approach in a post-hoc manner gives "*Twila Shively was an American competitive baseball player who played from 1945 through 1950 in the All-*

American Girls Professional Baseball League. Twila Shively is known for playing baseball. ..."

In contrast, active approach results in "*Twila Shively was an American competitive baseball player who played from 1945 through 1950 in the All-American Girls Professional Baseball League. She was born in Decatur, Illinois on March 20, 1922 and passed away on November 25, 1999. Twila began playing softball at the age of eight and quickly moved up in the softball ranks in Chicago. ...*"

Thus, the active approach results in an output of much higher quality and doesn't suffer from issues such are incoherence, consistency, repetition, etc.

Why the unit of generation is a sentence? We select a unit as a sentence over multiple sentences and (also over just a few words instead of a sentence) because of the following reasons:

Why not multiple sentences? In autoregressive generation, the generation depends on the context including the model's previously generated text. Thus, if we consider multiple sentences as a unit in our approach (let's say 3 sentences) and if one of the initial sentences is hallucinated (and thus replaced with the corrected sentence), the subsequent sentences (i.e., the remaining sentences of the unit) may not stand relevant (as they were based on a sentence that has been replaced) and it may make the generation incoherent. Furthermore, the propagation of hallucination is another negative contributor as the next sentences may carry forward the hallucination of the previous incorrect sentences. Thus, the subsequent sentences in the unit would need to be regenerated. This implies that using multiple sentences as a unit may not return that benefit (that too at the extra cost of generating multiple sentences at once).

Why not a phrase or a set of words? We note that using a few words (i.e., a window of text) may not have sufficient information to test the correctness of the concepts in the generation. For instance, if the window is of the following words: "Rick Mahler won three gold medals and 2 silver medals at the", it doesn't have sufficient information to validate the correctness of the individual concepts. On the other hand, a sentence typically provides richer context to validate the correctness of the concepts of the sentence.

Because of the above two reasons, we use a sentence as the unit in our method.

Order of Validation of Concepts Validation of different concepts can be done in a sequence (in ascending order of their calculated probability score) or in parallel. However, running this in parallel would require starting multiple threads which may not be supported by all machines. Thus, in this work, we study only the sequential validation strategy but note that it can be made more efficient by running it in parallel. We regard this sequential validation as a greedy exiting strategy as we proceed to the mitigation stage on detection of the first hallucinated concept.

3.3 Experiments and Results

We first highlight the two findings that motivate our approach (in 3.3.1.1). Then, we show the individual efficacy of our detection and mitigation techniques in 3.3.2.

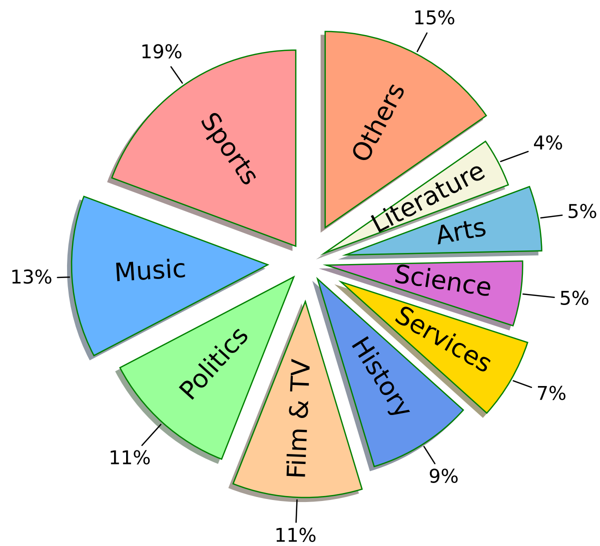


Figure 2. Distribution of instances across different domains in our topic set for the article generation task.

Finally, in 3.3.3, we show the effectiveness of our proposed active detection and mitigation approach.

Data and Annotation: In our experimental setup, we prompt the LLM to write about a given topic. We use topics from diverse domains as shown in Figure 2. In each domain, we include different

kinds of topics; for instance, Sports includes sportspersons, teams, and games; Music includes musicians, songs, music labels, and bands; Politics includes politicians, political parties, and elections, etc. We use a total of 150 topics in our data. For selecting the names of people, we randomly sample from the top 20% of longest articles in WikiBio dataset (Lebret, Grangier, and Auli 2016) as done in Manakul, Liusie, and M. J. Gales 2023. Similarly, we sample from the longest Wikipedia articles for the other topics. This is done to avoid obscure or ambiguous topics.

For each topic, we give the following input prompt to the models: `‘Write an` `article about <topic>'`. Then, we (the authors) annotate the correctness of the first five generated sentences. For this annotation, we look at search results from the web to find the relevant knowledge that either supports or contradicts the information in the sentence. In some cases, multiple web searches were required to check the correctness of different facets of a sentence. Furthermore, in a small number of cases, we could not find information supporting or contradicting the information in the generated sentence, we marked it as a case of extrinsic hallucination. We opt for this **expert annotation strategy** because despite the annotation task being a simple binary classification task, it requires considerable effort to check the correctness which can not reliably be collected via crowdsourcing. In addition to the **sentence-level annotation**, we also annotate correctness at **concept-level** (detailed in 3.3.1.2).

Table 4 shows the statistics of the sentences generated by the GPT-3.5 (text-davinci-003 with **temperature 0**) model. A sentence has ~ 18 words on average and each sentence has ~ 3.2 key concepts that are identified by our instruction technique.

Statistic	**Mean ± Std**
# Words in a Sentence	18.6 ± 5.55
# Key Concepts in a Sentence	3.27± 1.63
# Words in a Key Concept	1.79 ± 1.02

Table 4. Statistics of the generated sentences for the article generation task.

Table 5 shows examples of sentence-level and concept-level hallucination annotations.

Human Annotation and Agreement with Expert Annotation We additionally compile human annotations from two annotators on randomly sampled 10 topics

Sentence #	Sentence	Sentence-level Correctness
Sentence 1	Eleanor Arnason is an award-winning science fiction and fantasy author who has been writing since the 1970s.	Correct
Sentence 2	She is best known for her novel A Woman of the Iron People, which won the James Tiptree Jr. Award in 1991.	Correct
Sentence 3	Her work has been praised for its exploration of gender, race, and identity, as well as its imaginative world-building.	Correct
Sentence 4	Arnason was born in Minneapolis, Minnesota in 1942.	Hallucination
Sentence 5	She attended the University of Minnesota, where she earned a degree in English literature.	Hallucination

Table 5. Examples of both sentence and concept-level annotations for the input: "Write an article about Eleanor Arnason". Annotation for correct concepts is represented in green while annotation for hallucinated concept is represented in red.

(50 sentences). Specifically, we asked them to mark the correctness of the sentence by searching over the web, the same annotation procedure followed for expert annotation detailed in Section 3.3. Cohen's kappa of the annotators with the expert annotation is 0.84 and 0.92 respectively and the kappa within themselves is 0.84. This shows the high agreement and correctness of our annotations. We note that we use our expert annotations for all the results as they are more accurate and reliable.

Since the generation is for a variety of topics of different domains and would be beyond the common knowledge of a typical human, thus, we use web search to gather the relevant information to check the correctness of the generation. Multiple web searches were required in some cases because a generation can contain multiple facets of information all of which can not be validated in a single web search.

For example, sentences like "*Steven Threet is best known for his time at the University of Michigan, where he was a three-year starter and led the Wolverines to a Big Ten Championship in 2008.*", "*Rick Mahler was a Major League Baseball pitcher who played for the Atlanta Braves, Cincinnati Reds, and St. Louis Cardinals from 1979 to 1994.*" contain multiple facets that need to be validated separately because a single web search may not return all the information that is necessary to validate the correctness of all the facets of such sentences.

3.3.1 Motivating Findings

3.3.1.1 Propagation of Hallucination

Since we consider five sequentially generated sentences generated by the model for each topic, we investigate the relationship between '*hallucination in a generated sentence*' and '*hallucination in any previously generated sentences*' for an input. Since there are two binary variables, there exist four possibilities in this relationship, represented by YY, NY, YN, and NN in Figure 3. The figure demonstrates this relationship for sentences 2, 3, 4, and 5 (since no previously generated sentence for sentence 1) aggregated over all the topics in our dataset. Observations are as follows:

(a) **YY** > **NY**: Cases YY and NY correspond to the scenario when there is a previous hallucination. It can be observed that YY is considerably greater than NY implying that *when there is hallucination in the previously generated sentences, a sentence is more often hallucinated.*

(b) **YY** > **YN**: In YY and YN, the generated sentence is hallucinated. Here, YY is greater than YN implying that *a generated sentence is hallucinated more when there is hallucination in the previously generated sentences as compared to when there is no previous hallucination.*

(c) **NN** > **YN**: *When there is no hallucination in the previously generated sentences, a generated sentence is more likely to be correct, i.e., it is less often hallucinated.*

(d) **NN** > **NY**: *A generated sentence is 'correct' more when there is no previous hallucination as compared to when there is a previous hallucination.*

This shows that hallucination in a sentence increases the chances of hallucinations in the subsequently generated sentences, i.e., hallucination often propagates and thus actively detecting and mitigating them can fix the current hallucination and also prevent its propagation in the output.

3.3.1.2 Logits Provide Signal for Hallucination

To study the relationship between logit values and hallucination, we annotate correctness at concept-level also (in addition to sentence-level annotations described earlier). Specifically, for each identified concept, we mark whether the information about it in the generated sentence is hallucinated or not. Table 5 shows examples of both sentence and concept-level annotations. Figure 4 shows the trend of hallucination with our calculated probability scores. For sentence-level (Figure 10), we use the

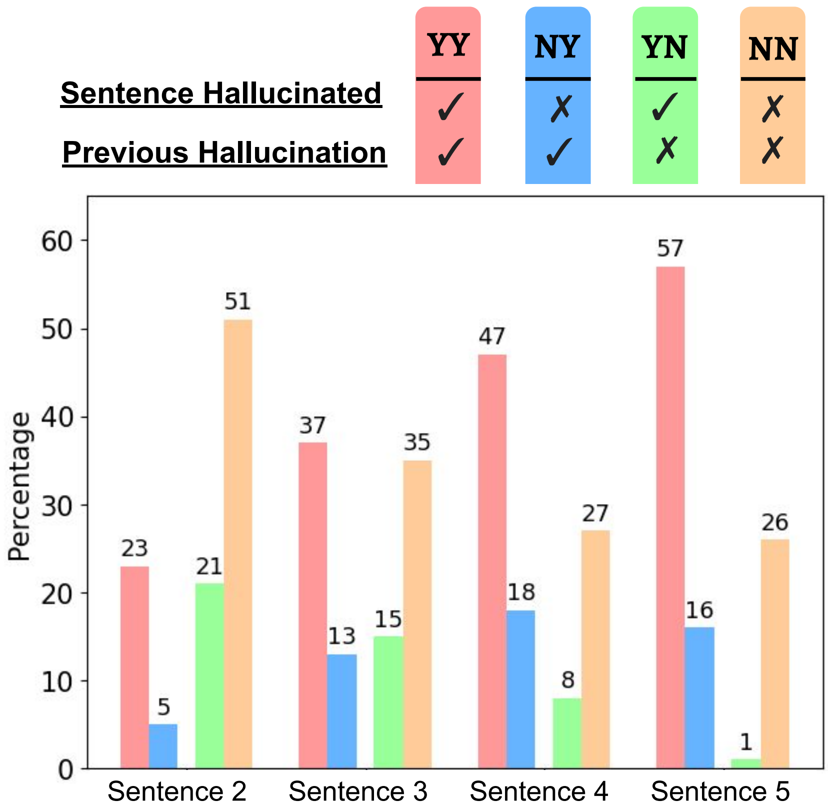


Figure 3. Demonstrating relationship between 'hallucination in a generated sentence' and 'hallucination in previously generated sentences'. Bars YY, NY, YN, and NN correspond to four possibilities.

minimum across tokens of all its identified concepts as the probability score, and for concept-level, we use the minimum across the concept's tokens as the probability score. The figure shows that as the probability score increases (or uncertainty decreases), the tendency to hallucinate decreases. This shows that the probability values can be utilized as a signal for hallucination, i.e., the low probability concepts can be considered as candidates of potential hallucination and their correctness in the sentence can be validated for detecting hallucinations.

We compare efficacy of different probability calculation techniques at detecting

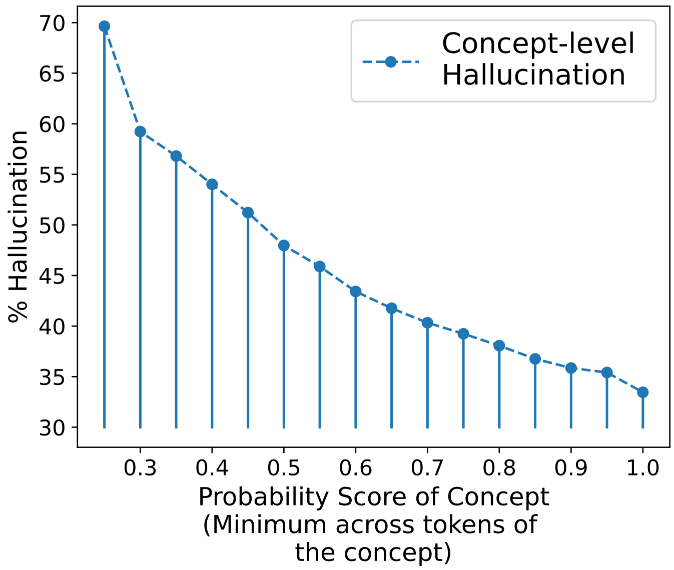


Figure 4. Trend of hallucination with the calculated probability score (MIN) at concept level. As the score increases, the tendency to hallucinate decreases.

hallucinations (in Section 3.7.4) and show that the 'MIN' technique achieves the highest area under the Precision-Recall curve.

3.3.2 Hallucination Detection and Mitigation

Detection: In Table 6a and 6b, we compare the detection performance of self-inquiry and web search techniques at both sentence and concept-levels. For sentence-level results, we predict the sentence to be hallucinated if the validation procedure fails for any identified concept. Note that in these results, we do not leverage the uncertainty score to select concepts for validation, instead we validate all the identified concepts. We study the relationship of recall with probability thresholds in

Figure 8. The tables show that **web-search technique achieves considerably high recall and precision** in detecting the hallucinations. Here, we emphasize

(a) **Sentence level**

Technique	Accuracy	Hallucinated		Not Hallucinated	
		Prec.	**Rec.**	**Prec.**	**Rec.**
Self-Inquiry	0.62	59.89	63.76	65.23	61.42
Web-Search	0.681	61.82	**85.96**	80.39	52.03

(b) **Concept level**

Technique	Accuracy	Hallucinated		Not Hallucinated	
		Prec.	**Rec.**	**Prec.**	**Rec.**
Self-Inquiry	0.65	47.96	45.85	73.37	74.98
Web-Search	0.75	58.17	**87.68**	91.69	68.30

Table 6. Hallucination detection performance of self-inquiry and web-search techniques. It also shows separate precision and recall on both hallucinated and non-hallucinated instances.

on the high 'recall' as we show that our mitigation approach does not introduce new hallucinations even in the case of incorrectly detected hallucinations, i.e., false positives.

Mitigation: On sentences where our validation procedure (using Web search) reports hallucinations, we apply our mitigation technique. We note that a sentence that is reported as hallucination can either be actually hallucinated (true positive) or not hallucinated (false positive). Table 7 shows the result of our method. It successfully mitigates the hallucination on 57.6% of the correctly detected hallucinations (True Positives). Furthermore, it achieves this at minimal 'deterioration' (3.06%), i.e., it incorrectly converts a minimal 3.06% of the non-hallucinated instances to sentences having incorrect information (hallucinated).

Analyzing Mitigation Failures: Table 10 and 11 show examples where our mitigation technique successfully mitigates and fails to mitigate the hallucinations, respectively. We observe that in many of the failure cases, our technique fixes some

Is Hallucinated? Before	After	Percentage
✓	✗	40.81%
✓	✓	30.04%
✗	✗	28.26%
✗	✓	0.89%

Table 7. Hallucination mitigation results after modifying the reported hallucinations.

hallucinated content of the sentences but fails to fix ALL the hallucinated content from them. Examples 1 and 2 in Table 11 correspond to this type of failure. Furthermore, in some of the failure cases, our technique results in a sentence that is no longer hallucinated but is not completely related to the topic. For instance, the fourth example in Table 11 about the topic 'Harry S. Kennedy'; the model generates "*Harry S. Kennedy was ... 35th President ...*" which is wrong and our mitigation technique modifies it to "*John F. Kennedy was ...*" which is factually correct but not related to the topic 'Harry S. Kennedy'. We attribute this to the mitigation step which is contingent on the information in the retrieved knowledge.

3.3.3 Active Detection and Mitigation

The two findings in 3.3.1 motivate our approach in which we actively detect hallucinations leveraging the logit values and mitigate them during the generation process which further helps in preventing their propagation. Specifically, we iteratively generate sentences and when our detection method reports hallucination (by validating

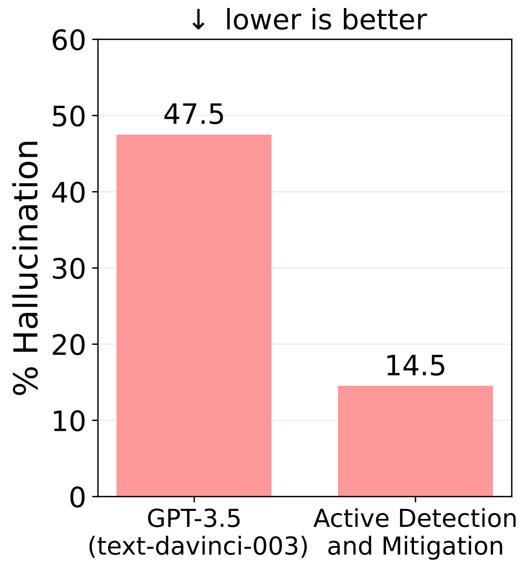


Figure 5. Comparing % hallucination in the output of GPT-3.5 with our active detection and mitigation approach on the 'article generation task'.

uncertain concepts), we repair the sentence using our mitigation method and then continue generating the next sentence. We demonstrated separate detection and mitigation efficacy in 3.3.2. Figure 5 compares the hallucination percentage in GPT-3.5's output and our "active" approach. It reduces the hallucination percentage from 47.4% to 14.53% which proves that the active intervention indeed successfully prevents hallucination propagation. In Figure 7, we plot this comparison for different categories of hallucinations and show that our approach does well in all the categories.

3.4 Additional Experiments

To further demonstrate our approach's wide applicability, we present three additional studies and discuss other usecases in Section 3.8.

3.4.1 Efficacy with Another LLM

Here, we compare hallucination % in the output of Vicuna-13B (on the 'article generation task') and with our proposed active detection and mitigation approach. We select Vicuna (v1.5) because it is the SOTA open-source model. Our approach considerably reduces the hallucinations (from 56% to just 18%) similar to the case with GPT-3.5 model. This study is conducted on 10 randomly sampled topics (i.e., 50 generated sentences) from the topic set described in Section 3.3. We note that similar to the setup with GPT-3.5 where we used instructional prompts with GPT-3.5 itself for all the steps of the approach (i.e., identifying key concepts, creating validation questions, etc.), following the same, here we use Vicuna-13B for all those steps. This result demonstrates **generality and applicability of our approach in reducing hallucinations of LLMs**.

3.4.2 Multi-hop Questions

We show that our approach can be adapted to improve the performance on multi-hop bridge questions (Table 12). Recall that our approach works by mitigating hallucination/incorrectness in the sentences generated by the model. Thus, if we can enable the model to answer these multi-hop questions step by step, then our active detection and mitigation approach can be applied to these steps, leading to correct predictions. To this end, we prompt the model and provide in-context examples demonstrating it to answer a given multi-hop question step by step. Section 3.7.6

GPT-3.5	GPT-3.5 few-shot	GPT-3.5 w/ know	Our Approach
54%	50%	38%	26%

Table 8. % Hallucination with different strategies on Multi-hop bridge questions. Lower is better.

shows the corresponding prompt used for this purpose. Specifically, for a test question, the model generates the answer in multiple steps (one step at a time) and for each step, we apply our technique in which we first identify the low probability concepts from the sentence, validate their correctness using web search results, mitigate the hallucination (if detected), and then proceed to generate the next step. In our case study, we sample 50 multi-hop bridge questions from the validation set of HotpotQA (Z. Yang et al. 2018).

Main Result (Table 8): First, we show the performance of GPT-3.5 which answers 54% of the questions incorrectly. GPT-3.5 with in-context examples results in a slight improvement over the zero-shot performance. GPT-3.5 leveraging the knowledge retrieved from the web (using the question as search query) as context improves the performance and results in fewer incorrect predictions. Finally, we show the performance of our active detection and mitigation approach which results in considerably fewer hallucinations (just 26%), i.e., a higher percentage of correct answers. Table 13 shows examples of responses generated using our approach. This demonstrates our approach's effectiveness in improving performance on multi-hop QA.

3.4.3 False Premise Questions

LLMs perform remarkably well on a wide range of questions that are factually correct and make the right assumptions. However, users in real world often ask questions that are based on false premises such as "Why energy is <u>absorbed</u> in exothermic reactions?” and “Why do floppy disks have <u>higher</u> storage capacity than USB drives?”. We observe that SOTA models often struggle to appropriately respond to such questions; thus, they serve as another challenging evaluation setting. This is also a result of ‘sycophancy’ (Jerry Wei et al. 2023) demonstrated by LLMs. To this end, we conduct a study and compile a set of 50 such adversarial questions, i.e., questions for which GPT-3.5 gives incorrect response. Furthermore, we also create a true premise question corresponding to each false premise question (Table 14).

Approach: An ideal response to such questions is application dependent; some applications may require identifying such questions and then abstaining on them like the selective prediction systems (Kamath, Jia, and Liang 2020; Xin et al. 2021; Varshney and Baral 2023) while some applications may also require suggesting a ‘rectified’ question and providing response to that rectified question like the search engines. Our approach supports these requirements by using the validation and mitigation step on the given question.

Specifically, we first retrieve knowledge (via Bing Search using the question as query). Then, we apply our validation and mitigation technique, i.e., conditioned on the retrieved knowledge, we prompt the model to respond ‘Yes’ if the question makes factually correct assumptions, otherwise respond ‘No’. If the response is No, then we proceed to modify the question using the mitigation step. Table 15 shows the corresponding instructional prompts. This step enables identifying false premise questions and rectifying them to facilitate the system in providing an appropriate response. Importantly, we also show that our approach does not incorrectly modify a

GPT-3.5	GPT-3.5 w/ know	Our Approach
100%*	78%	24%

Table 9. % Hallucination with different strategies on false premise questions. * indicates that the questions are adversarial. Lower is better.

true premise question. This is crucial because if the user's question is correct then the system's response must be pertinent to that.

Main Result (Table 9): As mentioned above, the questions in our evaluation set are adversarially collected, i.e., GPT-3.5 gives incorrect response to all of them. We evaluate the performance of GPT-3.5 when retrieved knowledge (via bing search) is given as additional context. We find that even with the knowledge, it manages to answer only 24% false premise questions correctly, i.e., hallucinates on the remaining 76%. In contrast, our approach answers 76% questions correctly and hallucinates only on 24%. Furthermore, we note that even in some of these 24% hallucinated responses, some of the individual sentences in the responses are correct. However, since we focus on complete answer correctness, we consider them as incorrect. Table 17 shows examples of responses on false premise questions generated by the GPT-3.5, GPT-3.5 with retrieved knowledge, and our active detection and mitigation approach.

3.5 Advantages of the Proposed Approach

In addition to the effectiveness and wide applicability of our approach in addressing hallucinations of LLMs (as demonstrated through extensive experiments), it has numerous other advantages:

1. It **circumvents the need for modifying the internals of LLMs** to address their hallucination problem making it a plug-and-play yet effective solution.
2. It **improves the explainability and interpretability of the LLM's output** as the generation can be attributed back to the retrieved knowledge.
3. The knowledge retrieval step **allows opportunities to use proprietary/domain-specific knowledge during the generation** process. Thus, allowing it access to the updated information.
4. Our retrieval method retrieves knowledge pertinent to the sentence and thus **enables accurate hallucination detection and mitigation**.
5. Active intervention **allows opportunities for course correction** during the generation process.

3.6 Limitations of the Proposed Approach

3.6.1 Impact on Inference Efficiency

Our approach results in improvements in the form of reduced hallucinations and thus makes the model more reliable; however, it comes at the expense of increased inference cost. However, we believe that at current time, to enable the widespread adoption of LLMs, it is more important to address their reliability and trustworthiness concerns because computational advancements are ongoing at a rapid pace. Moreover, even larger models with multi-fold times more parameters such as PaLM (540B) (Chowdhery et al. 2022), Gopher (280B) (Rae et al. 2021), and MT-NLG (530B)

(Smith et al. 2022) are also being developed which have even higher inference cost showcasing a larger focus of the community on developing better performing systems. Though it may not be a problem for all use cases, we provide a detailed discussion on it for all the steps with suggestions on their lower-cost alternatives.

Identifying Important Concepts: Firstly, we note the importance of this step because validating the correctness of the entire sentence at once is infeasible as a sentence can contain multiple different facets all of which can not be validated at once. In contrast, individually validating correctness corresponding to the concepts provides opportunities for accurately detecting incorrectness. Thus, if we skip this step and directly proceed to the validation step for the entire sentence then it will have limitations. For example, sentences like "*Steven Threet is best known for his time at the University of Michigan, where he was a three-year starter and led the Wolverines to a Big Ten Championship in 2008.*" contain multiple facets that need to be validated separately because a single web search may not return all the information that is required to validate the entire correctness.

This step incurs the cost of inference in which the input is the instruction (provided in Table 3) and the sentence. We mention the benefits of "instructing the model" technique in Section 3.2.1.1.

We discuss other lower-cost alternatives for this step below: A simple yet efficient method is to leverage a relatively smaller LLM for this step. This is feasible because identifying the concepts is an "easy" step and even smaller LLMs are typically very effective in this. Moreover, even a more smaller model such as T5 can also be finetuned for this particular task which can considerably reduce the cost. Smaller models have low inference cost (both in terms of FLOPs and latency). Furthermore, the other techniques already discussed in the paper, namely Entity Extraction and keyword extraction are other lower-cost alternatives. Specifically, the KBIR model is built on top of RoBERTa architecture which is even more efficient.

In summary, smaller models (smaller LLMs or task-specific finetuned models) can be utilized for this task to make it more efficient.

Calculating Model's Uncertainty: This is not a resource intensive task as it just requires calculating the score from the logit values.

Creating Validation Question: Similar to the first step, creating a validation query for a concept is also a task at which even smaller models (that even have only a few million parameters) do quite well. A lot of existing research on question generation uses the T5 models. Creating a validation question using an LLM requires taking the instruction (filled with the concept) (Table 3) and the sentence as input.

Another cost-effective alternative for this step is to simply mask out the selected concept from the sentence and use it as the validation query for the web search. Though, it requires some heuristics to create an appropriate validation query (such as selecting only a window of tokens on both sides of the concept after masking as the validation query, this would be required because using the entire sentence would have many different facets, and web search may not return relevant results). This would definitely make it much more efficient but it will lose effectiveness in creating "high-quality" queries pertinent to the concept and thus may not result in slight degradation in the validation procedure.

Answering Validation Question and Mitigation Steps: These steps are more costly than the others because they also take the retrieved knowledge as input. We note that these are crucial steps of the method. They can be made more efficient (though it will compromise the effectiveness) by combining them into a single step, i.e., validation and mitigation can be done using a single instructional prompt. However, we note that this is a relatively difficult task as compared to the previous steps and thus decomposing it into two individual steps provides better results. Thus, making this step more efficient will have tradeoffs with the performance.

Overall, these steps can be made more efficient (in terms of both computation cost and latency) using smaller LLMs or external task-specific tools. In contrast, the methodology highlighted in red in Figure 1 uses the same model for all the steps. Furthermore, we note that in resource-constrained applications, the suggested efficient alternatives can be utilized.

We present an empirical analysis of the latency where we compare the latency of all the steps of the methodology. Figure 6 shows the comparison of latency of various steps (at a sentence level). We note that the latency of the mitigation step is low as it is only conditionally called for some sentences. We show the average mitigation latency for sentences on which it is called in the Mitigation* bar. We conduct this study for 10 topics (i.e., 50 sentences) for the GPT-3.5 (text-davinci-003) model.

Comparison of Overall Latency with the Generation: The overall latency of the method is 2.58 times that of the regular generation (5354.20 against 2071.69).

Why the latency of the generation step is high? This is because for the later sentences, it also takes the context in the input.

Why the latency of validation is high? This is because validation procedure

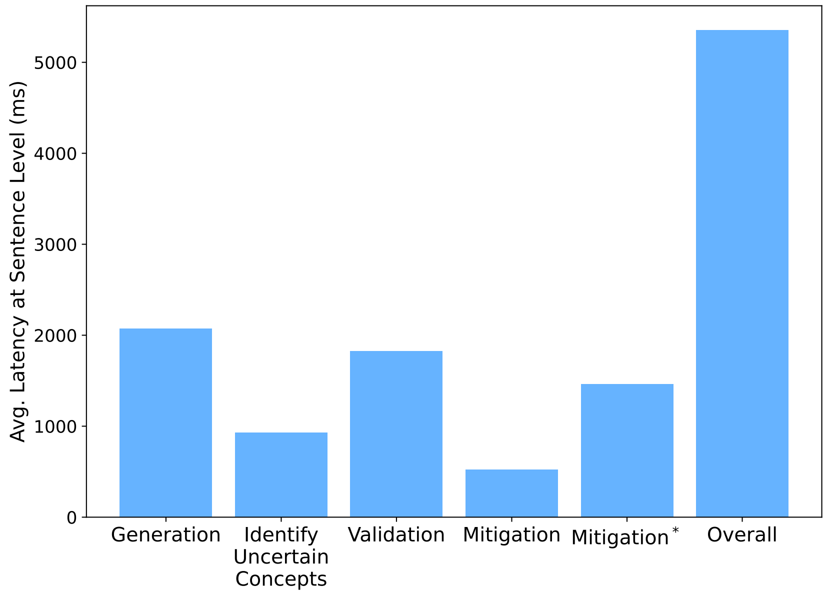


Figure 6. Comparing latency of various steps of the methodology (at a sentence level). Note that the latency of mitigation is low as it is only conditionally called for some sentences. We show the average mitigation latency for sentences on which it is called in the Mitigation* bar.

includes three steps (validation question creation, retrieval, and answering validation question). Furthermore, validation could be required for multiple concepts.

What does Mitigation* represent? Note that the mitigation step is only conditionally executed for some sentences. We show the average mitigation latency for sentences on which it is called in the Mitigation* bar.

3.6.2 Correctness of Retrieved Knowledge

Web searches can sometimes return information that is fabricated. Though we use the top web search results as our context (primarily from the reliable sources), there remains a chance that the knowledge is incorrect which can result in incorrect hallucination detection.

3.6.3 Error Propagation

Multiple sequential steps can increase the chances of propagation of error from one to the other; however, we note that the individual steps in our approach are very simple, and the LLMs perform remarkably well on these steps. Furthermore, our mitigation technique does not introduce new hallucinations even in the case of incorrectly detected hallucinations, i.e., false positives.

3.7 Further Analysis

3.7.1 Active Detection and Mitigation Performance Analysis

Figure 5 compares the percentage of hallucination in the output of GPT-3.5 model and our approach. It reduces the hallucination percentage from 47.4% to 14.53%. This proves that the active intervention during the generation process also does well in preventing the propagation of hallucination in the model's output. In Figure 7, we plot this comparison for different categories of hallucination and show that our approach does well in all the categories.

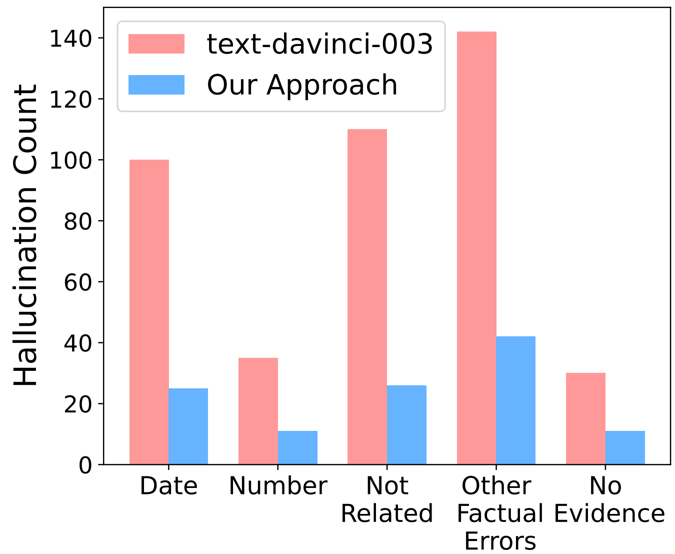


Figure 7. Comparing hallucinations across different categories of text-davinci-003 and generation from our approach.

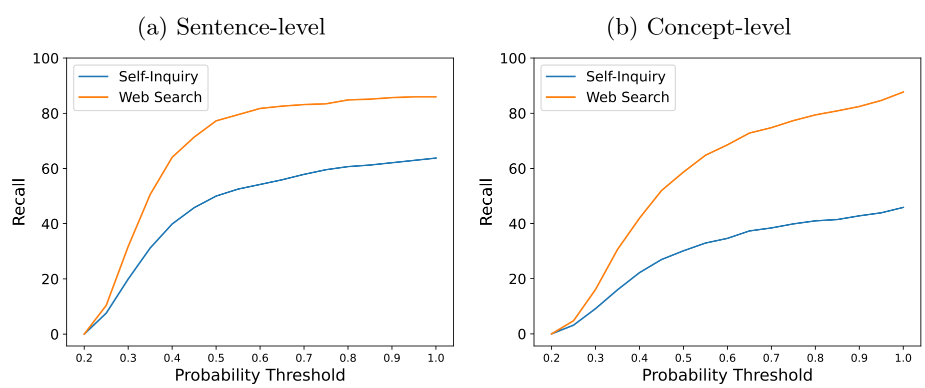


Figure 8. Recall of hallucination detection vs Probability threshold plot for Self Inquiry and web search techniques at both sentence-level and concept-level.

3.7.2 Recall of Hallucination Detection vs Probability Threshold

Figure 8 compares the recall of hallucination detection for self-inquiry and web search techniques at different probability thresholds. **Web search considerably outperforms self-inquiry at all thresholds** and hence is better at detecting

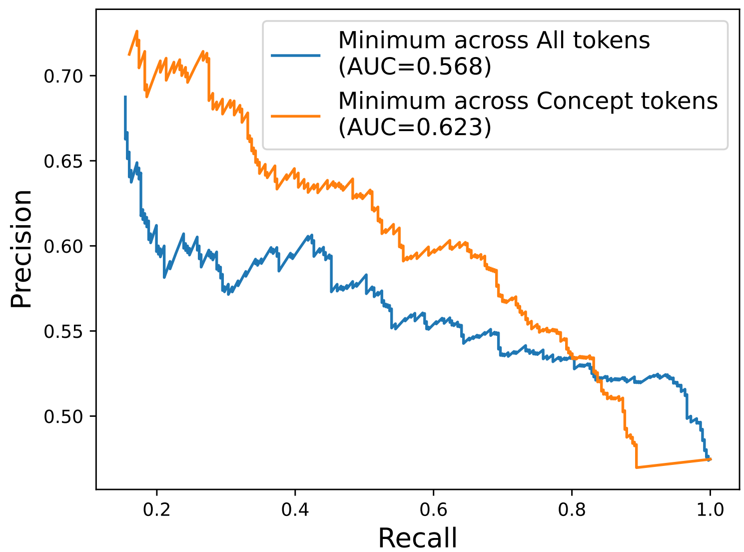


Figure 9. Demonstrating the benefit of identifying concepts from a sentence for detecting hallucinations. The figure shows precision-recall curves for the sentence level hallucination detection task corresponding to two methods that use the probabilities calculated from the logit output values. The blue curve corresponds to the technique in which we use the minimum probability across all tokens of the sentence and the orange curve is for the technique in which we use the minimum over only the tokens of the identified concepts.

hallucinations. Selecting the probability threshold depends on the tolerance level of the application. For instance, in a high-risk application domain like biomedical, we can keep a very high threshold, and in a low-risk domain like movie recommendation, we can relatively lower threshold. In this work, we use a probability threshold of 0.55. However, we note that it can be adjusted as per the application requirements.

3.7.3 Hallucination Mitigation Analysis

Tables 10 shows examples where our mitigation technique successfully mitigates the hallucinations. Table 11 shows examples where our technique fails to mitigate

Topic	Rick Mahler
Original Sentence	Mahler was born in San Diego, California in 1956. (✗)
Hallucination Reason	He was born in Austin in 1953.
Modified Sentence	Rick Mahler was born in Austin, Texas on August 5, 1953. (✓)
Topic	Matthew Aylmer, 1st Baron Aylmer
Original Sentence	Baron Aylmer, was an English naval officer and politician who served as a Member of Parliament and later as a peer in the House of Lords. (✗)
Hallucination Reason	He was in House of Commons not House of Lords.
Modified Sentence	Baron Aylmer was an Anglo-Irish Royal Navy officer and Whig politician who sat in the English and British House of Commons between 1695 and 1720. (✓)
Topic	Derek King
Original Sentence	In 2014, Derek King released his debut single "Let's Go," which quickly became a hit in Canada. (✗)
Hallucination Reason	"Let's Go" was not his song.
Modified Sentence	In 2014, Derek King released his debut mixtape Fake I.D., which featured fellow Bay Area artists Rayven Justice and IAMSU!, among others. (✓)
Topic	Marshall Manesh
Original Sentence	Marshall Manesh is an Iranian-American actor best known for his roles on the television shows How I Met Your Mother and The Middle. (✗)
Hallucination Reason	He was not in The Middle.
Modified Sentence	Marshall Manesh is an Iranian-American actor best known for his recurring roles on the television shows Will & Grace, Scrubs, Andy Barker, P.I., Hot in Cleveland, Boston Legal, and How I Met Your Mother, where he played taxi driver Ranjit. (✓)
Topic	William J. Flanagan, Jr.
Original Sentence	He is the founder and CEO of Flanagan Financial Group, a financial services firm based in New York City. (✗)
Hallucination Reason	The complete sentence is hallucinated as he is a retired US Navy admiral.
Modified Sentence	He is a retired four-star admiral who served as Commander in Chief, United States Atlantic Fleet from 1994 to 1996 and is the recipient of numerous military awards. (✓)

Table 10. Examples of successful mitigation of hallucinations by our mitigation technique. Original Sentence corresponds to the sentence generated by the model and Modified Sentence corresponds to the sentence obtained on applying our technique.

Topic	Matthew Aylmer, 1st Baron Aylmer
Original Sentence	Aylmer joined the Royal Navy in 1790 and served in the French Revolutionary Wars and the Napoleonic Wars. (✗)
Hallucination Reason	He did not serve in these wars.
Modified Sentence	Aylmer entered the Royal Navy under the protection of the Duke of Buckingham as a Lieutenant in 1678 and served in the French Revolutionary Wars and the Napoleonic Wars. (✗)
Hallucination Reason	It rectified the date but failed to correct the hallucination about wars.
Topic	K. S. Manilal
Original Sentence	Manilal was a prolific writer and translator, having written more than 50 books and translated over 100 works from English, Sanskrit, and other languages into Malayalam. (✗)
Hallucination Reason	He has not written 50 books and translated works to English and Malayalam.
Modified Sentence	Manilal was a prolific researcher and translator, having translated Hendrik van Rheede's 17th century Latin botanical treatise, Hortus Malabaricus, into English, Sanskrit, and Malayalam. (✗)
Hallucination Reason	The information about Hortus Malabaricus is correct but he translated it into English and Malayalam only and not Sanskrit.
Topic	Harry S. Kennedy
Original Sentence	Harry S. Kennedy was an American politician who served as the 35th President of the United States from 1961 to 1963. (✗)
Hallucination Reason	This sentence is true for John F. Kennedy not Harry S. Kennedy.
Modified Sentence	John F. Kennedy was an American politician who served as the 35th President of the United States from 1961 to 1963. (✗)
Hallucination Reason	This sentence is not hallucinated but it is not related to the topic.

Table 11. Examples where our mitigation technique fails to mitigate complete hallucination in the generated sentence. Original Sentence corresponds to the sentence generated by the model and Modified Sentence corresponds to the sentence obtained on applying our technique.

hallucinations. We observe that in many of the failure cases, our technique fixes some hallucinated content of the sentences but fails to fix ALL the hallucinated content from them. Furthermore, in some of the failure cases, our technique results in a sentence which is no longer hallucinated but it not completely related to the topic.

Clarification on Percentage Numbers specified for Mitigation Performance in Section 3.3.2: Table 7 shows the percentage of the four scenarios for (Before Modification, After Modification). We mention that "It successfully mitigates the hallucination on 57.6% of the correctly detected hallucinations (True Positives)". Therefore, this number corresponds to (40.81/ (40.81 + 30.04) = 57.6%).

3.7.4 Analysis of Logit Output Values

3.7.4.1 Benefit of Identifying Concepts from a Sentence

Now, we demonstrate the benefit of identifying concepts from a sentence and leveraging the logit output values corresponding to their tokens for detecting hallucinations. To this end, we plot precision-recall curves for the hallucination detection task corresponding to two methods that use the probabilities calculated from the logit output values. The blue curve corresponds to the technique in which we use the minimum probability across **all tokens** of the sentence and the orange curve is for the technique in which we use the minimum over **only the tokens of the identified concepts**. Figure 9 shows the two curves. The orange curve achieves higher area under the precision-recall curve implying that utilizing **the probabilities of the**

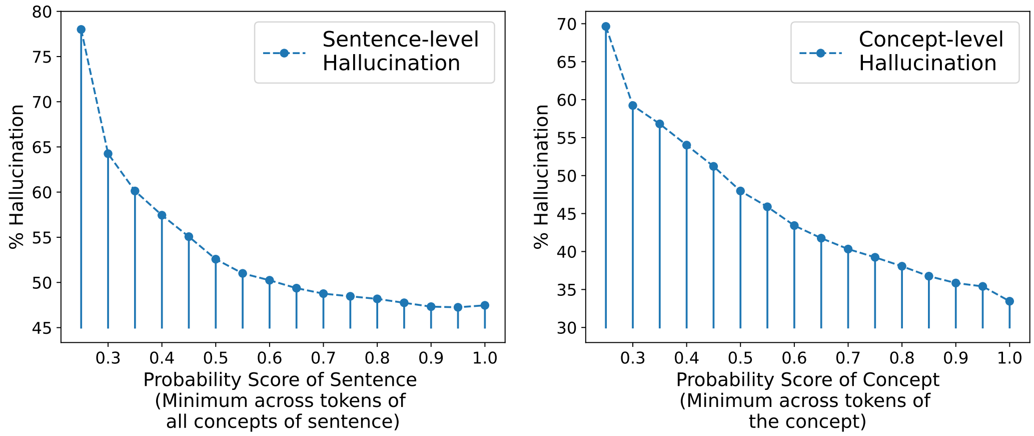


Figure 10. Trend of hallucination with the calculated probability score (Minimum technique) at both the sentence and concept levels. As the score increases, the tendency to hallucinate decreases.

concept tokens provides a stronger signal for hallucination as compared to the probabilities corresponding to all the tokens.

3.7.4.2 Logit Output Values with Minimum Technique

Figure 10 shows the trend of hallucination with our calculated probability scores at both sentence and concept levels. For sentence-level, we use the minimum across tokens of all its identified concepts as the probability score, and for concept-level, we use the minimum across the concept's tokens as the probability score. The figure shows that as the probability score increases (or uncertainty decreases), the tendency to hallucinate decreases.

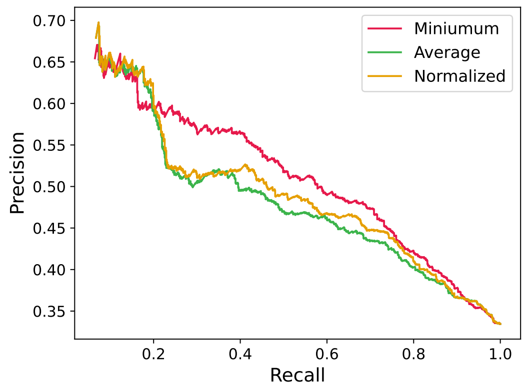


Figure 11. PR curves for the hallucination detection task (concept-level) using the three probability calculation techniques. 'Minimum' technique achieves highest AUC.

3.7.4.3 Comparing Probability Calculation Techniques

Figure 11 shows the Precision-Recall curves for the hallucination detection task (at concept-level) using the three probability calculation techniques, i.e., Minimum, Average, and Normalized (described in 3.2.1.2). **The 'Minimum' technique achieves the highest area under the curve and hence is better at the hallucination detection task.**

3.7.5 Efficacy with Another LLM

Figure 12 compares hallucination % in the output of Vicuna-13B (on the 'article generation task') and with our proposed active detection and mitigation approach.

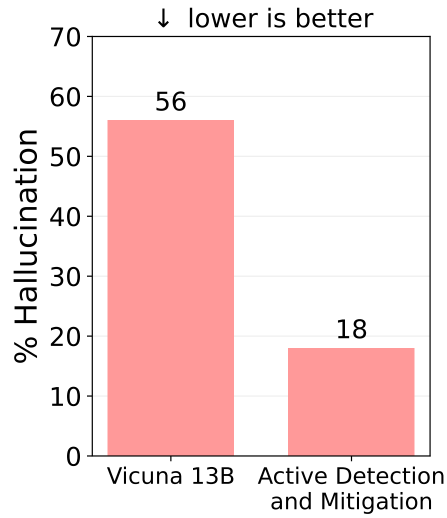


Figure 12. Comparing hallucination % for Vicuna-13B and our approach on the 'article generation task'.

Question	**Answer**
The football manager who recruited David Beckham managed Manchester United during what timeframe?	from 1986 to 2013
The Vermont Catamounts men's soccer team currently competes in a conference that was formerly known as what from 1988 to 1996?	the North Atlantic Conference
Ralph Hefferline was a psychology professor at a university that is located in what city?	New York City
What is the county seat of the county where East Lempster, New Hampshire is located?	Newport
Blackfin is a family of processors developed by the company that is headquartered in what city?	Norwood, Massachusetts

Table 12. Examples of multihop questions from HotpotQA.

3.7.6 Multi-hop QA Experiment

3.7.6.1 Prompt with In-context Examples

Question: Which team does the player named 2015 Diamond Head Classic's MVP play for?

Answer: The 2015 Diamond Head Classic's MVP was Buddy Hield. Buddy Hield plays for the Sacramento Kings of the NBA. So, the answer is Sacramento Kings.

Question: The Dutch-Belgian television series that "House of Anubis" was based on first aired in what year?

Answer: "House of Anubis" was based on is Het Huis Anubis. Het Huis Anubis was first aired in 2006. So, the answer is 2006.

Question: What is the birth date of the coach of the 2014-15 Michigan Wolverines men's basketball team?

Answer: The coach of the 2014-15 Michigan Wolverines men's basketball team was John Beilein. John Beilein's birth date is February 5, 1953.

Question: {question} **Answer**:

Prompt with in-context examples used for prompting the model to answer 'bridge' multi-hop questions step by step.

Question: *Kent Dairy Round Barn is near the city in what Montana county?*
Response: The Kent Dairy Round Barn is located near Red Lodge, Montana. Red Lodge is located in Carbon County, Montana. So, the answer is Carbon County.

Question: *What political party was the 6th governor of Hawaii that passed bills to help with global warming in Hawaii?*
Response: The 6th governor of Hawaii was Linda Lingle. Linda Lingle was a member of the Republican Party and passed bills to help with global warming in Hawaii. So, the answer is Republican Party.

Question: *In what city is the mortgage lending company, founded by Dan Gilbert, headquartered?*
Response: The mortgage lending company founded by Dan Gilbert is Quicken Loans. Quicken Loans is headquartered in Detroit, Michigan. So, the answer is Detroit, Michigan.

Question: *Actor David Lee Stenstrom played the character Waldo the inventor in a TV show that ran on Nickelodeon during what yeras?*
Response: The TV show was Out of Control (1984). David Lee Stenstrom played the character Waldo the inventor in the show. The show ran on Nickelodeon from 1984 to 1985. So, the answer is 1984 to 1985.

Question: *What music school did the singer of The Search for Everything: Wave One attend?*
Response: The singer of The Search for Everything: Wave One is John Mayer. John Mayer attended the Berklee College of Music in Boston, Massachusetts. So, the answer is Berklee College of Music.

Table 13. Examples of responses generated using our approach for multihop bridge questions.

3.7.6.2 Examples of Multi-hop Bridge Questions

Table 12 shows examples of multi-hop bridge questions from HotpotQA dataset.

3.7.6.3 Examples of Model Responses

Table 13 shows examples of responses generated using our approach for multi-hop bridge questions.

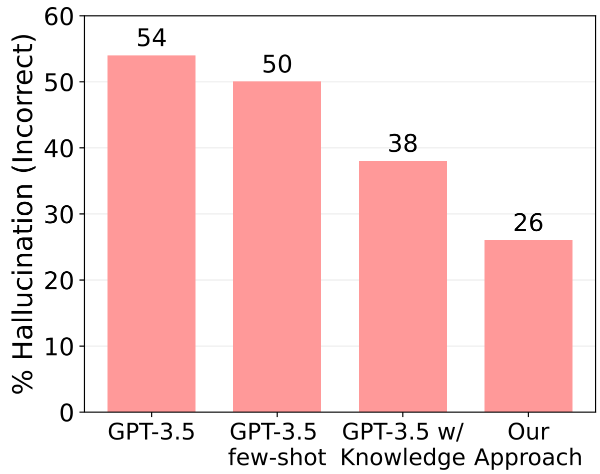


Figure 13. % Hallucination results on Multi-hop bridge Questions.

False Premise Question	True Premise Question
Why does Mars have **three** moons?	Why does Mars have two moons?
Why are golf balls **bigger** than basketballs?	Why are golf balls smaller than basketballs?
What are some papers on the relationship between homeschooling and **neuroplasticity**?	What are some papers on the relationship between homeschooling and college preparedness?
Why do migrant workers **never leave** their home?	Why do migrant workers leave their home?
When a diver swims deeper, why does the water **pressure declines**?	When a diver swims deeper, why does the water pressure increases?
Why does Mars have **higher gravity** than Earth?	Why does Mars have lower gravity than Earth?
Why is **baseball the most popular sport** in the world?	Why is soccer the most popular sport in the world?
How does **Earth make up over 99% of the solar system's mass**?	How does Sun make up over 99% of the solar system's mass?
Why does **Helium have atomic number of 1**?	Why does Hydrogen have atomic number of 1?

Table 14. Examples of 'false premise' questions and their corresponding 'true premise' counterparts.

3.7.7 False Premise QA Experiment

Table 14 shows examples of false premise and true premise question pairs. Table 17 shows responses generated on a few false premise questions by the GPT-3.5 (text-

Step	Prompt
I	`Context: {search results} Question: {question} Based on the above context, does the above question make factually correct assumptions?`
II	`Context: {response_of_Step_I} Question: {question} Rectify the incorrect information in the given question based on the context.`

Table 15. Instructional Prompts for rectifying the false premise questions.

Original Question	After Modification
False Premise Questions	
Why does Mars have three moons?	Why does Mars have two moons? (✓)
Why are golf balls bigger than basketballs?	Why are golf balls smaller than basketballs? (✓)
What are some papers on the relationship between homeschooling and neuroplasticity?	What are some papers on the relationship between homeschooling and learning outcomes? (✓)
Why USA has the lowest happiness index?	What factors have contributed to the decline in happiness among Americans? (✓)
How many metres does a typical apple weigh?	How many grams or ounces does a typical apple weigh? (✓)
Why do all rabbits have red eyes?	Why do some rabbits have red eyes? (✓)
Why does Helium have atomic number of 1?	Why does Helium have atomic number of 2? (✓)
Why does Bangladesh have the highest population in the world?	Why does Bangladesh have the highest population growth rate in the world? (✗)
Why are tigers' eggs bigger than chicken's eggs?	Why do some breeds of chickens lay larger eggs than others? (✗)
True Premise Questions	
Why gases are shapeless?	Why are gases shapeless? (✓)
How did USA become a developed country?	How did the United States become a developed country (✓)
Why Afghanistan has a low happiness index?	What factors contribute to Afghanistan's low happiness index? (✓)
Why are golf balls smaller than basketballs?	Why are golf balls typically smaller than basketballs? (✓)
How were the 2020 USA presidential election?	What were the results of the 2020 USA presidential election? (✓)

Table 16. Examples of original questions (both false premise and true premise) and the questions after rectification. ✗and ✓indicate that the modified question is incorrect and correct, respectively.

False Premise Q.	*Why are golf balls **bigger** than basketballs?*
GPT-3.5	Golf balls are bigger than basketballs because they need to be able to travel farther distances and have a larger surface area to create more spin and control. The larger size also helps to reduce air resistance, allowing the ball to travel farther. (✗)
GPT-3.5 w/ Knowledge	Golf balls are bigger than basketballs because they need to be heavier and denser in order to travel farther when hit with the same amount of force. The larger size allows for more mass-density, which gives the ball more momentum and distance when hit. (✗)
Our Approach	Golf balls are smaller than basketballs because they are optimized for golfers to provide the most ball velocity and distance possible, along with a controllable spin rate. The diameter of a golf ball must reach 1.68 inches, or 42.67 mm, in diameter, and a weight of no more than 1.62 ounces, or 45. (✓)
False Premise Q.	*Why do deserts have **higher** soil moisture than grasslands?*
GPT-3.5	Deserts have higher soil moisture than grasslands because they receive more rainfall. Deserts typically receive less rainfall than grasslands, but the rain that does fall is more concentrated and intense. This causes the soil to absorb more moisture than in grasslands, where the rainfall is more spread out and less intense. (✗)
GPT-3.5 w/ Knowledge	Deserts have higher soil moisture than grasslands because they receive more precipitation than grasslands. Deserts are located in areas where the demand for water by the atmosphere (evaporation) and plants (transpiration) is much greater than precipitation. This means that the soil in deserts is able to retain more moisture than the soil in grasslands. (✗)
Our Approach	Deserts have lower soil moisture than grasslands because they have less water vapor in the air, which makes it harder to trap heat or cold in a desert. Additionally, deserts typically have less precipitation than grasslands, which further contributes to lower soil moisture levels. (✓)

Table 17. Comparing responses generated on a few false premise questions by the GPT-3.5 model, GPT-3.5 moel leveraging the retrieved knowledge as context, and our approach.

davinci-003) model, GPT-3.5 (text-davinci-003) using the retrieved knowledge as context, and our approach.

Efficacy of Question Rectification: We analyze the performance of our approach in rectifying questions; **it successfully repairs 76% false premise questions while not incorrectly modifying any true premise question.** Though this step makes modifications in a small number of true premise questions (6

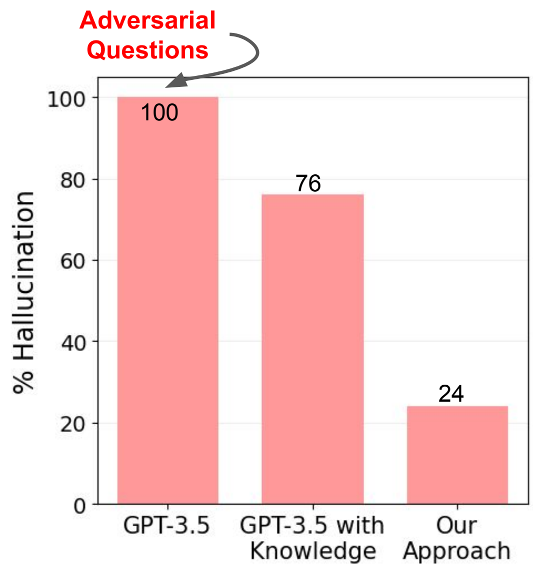


Figure 14. Results on 'False Premise Questions' for GPT-3.5, GPT-3.5 leveraging knowledge (retrieved via web search) and our approach.

instances), it does not change their semantics as shown in Table 16. Not incorrectly modifying a true premise question is an important characteristic of this approach.

3.7.8 Effectiveness of the Method beyond the First Five Generated Sentences

Our study on the article generation task is conducted on the first five generated sentences. After applying our method, the correctness at sentence number level (averaged over all the inputs) is as follows (Sentence 1: 90.0%, Sentence 2: 82.67%, Sentence 3: 86.67%, Sentence 4: 82.67%, Sentence 5: 85.34%). These values are indeed close and do not considerably reduce as the sentence number increases. With this result, we show that our method of active detection and mitigation successfully mitigates the hallucination throughout the generation (not restricted to any specific sentence number). Furthermore, it shows that the ability to address hallucinations does not considerably diminish as the sentence number increases. Thus, even increasing the number of sentences is not expected to considerably impact the improvement that our method would bring

3.7.9 Effectiveness of Retrieval Alone

For a fair comparison, we also compare the performance of retrieval alone with our active intervention approach. We also underline the advantages of our active intervention method over the retrieval alone method.

Figure 13 shows this comparison for the MultihopQA settings. Specifically, using the retrieved knowledge alone (retrieved using the question as the search query), the model's hallucination is at 38%. Using our approach of active intervention the hallucination is at 26%. We attribute our performance improvement to the active correction in the intermediate steps which eventually leads to improved answers.

Similarly, in the false premise QA setting, we show this comparison in Figure 14. We note that in this case, the improvement is even larger (76% vs 24%). This is because of a recently studied concept of sycophancy, where LLMs tend to generate responses that favor the user's perspective rather than providing correct or truthful answers, which can result in hallucinations. Our approach addresses this problem and reduces the hallucination.

Advantages of our active intervention over the retrieval alone baseline: Firstly, active retrieval retrieves the knowledge that is pertinent to the current sentence in the generation. In contrast, single retrieval retrieves only once and does not have the opportunity of retrieving knowledge pertinent to the current sentence.

Also, active intervention allows opportunities for course correction during the generation process i.e. if a sentence is hallucinated then it is fixed and then the subsequent sentences are generated. This prevents the propagation of hallucinations and also drives the generation in the right direction.

Furthermore, single retrieval can constrain the generation to be dependent on what has been retrieved initially. In contrast, active intervention allows the model to follow its course of generation and retrieve the knowledge based on that, unlike single retrieval where the generation is based on the retrieved knowledge

3.8 Other Applications of our Approach

Our approach has utility in a variety of other applications also such as Abstractive Summarization and Claim Verification. In abstractive summarization where the generated summary has been shown to be often hallucinated (Cao, Dong, and Cheung 2022; Zhao, Cohen, and Webber 2020; S. Chen et al. 2021) can be improved using our approach. Here, the relevant knowledge during validation will be retrieved from the original document instead of the web. Our approach can be adapted for the claim verification task also as we can first identify the key sub-claims and then verify each sub-claim using the validation procedure. Here, the mitigation step will also be useful for providing explanations behind the model's decision. We leave exploring these other usecases of our approach for future work.

3.9 Conclusion

We proposed an approach called active detection and mitigation to address the problem pertaining to the factual hallucinations of large language models. We demonstrate the phenomenon of propagation of hallucination which motivates our active intervention approach. Through systematic and extensive experiments on several tasks such as article generation, multi-hop QA, and false premise QA, we showed that our approach considerably reduces the hallucinations of LLMs. We also demonstrated the individual efficacy of our detection and mitigation techniques. Specifically, our detection technique achieves a high recall and the mitigation technique successfully mitigates a large fraction of the correctly detected hallucinations. Notably, the mitigation technique does not introduce new hallucinations even in the case of incorrectly detected hallucinations, i.e., false positives. We further demonstrated the effectiveness and wide applicability of our approach through several interesting studies including evaluation with another LLM from a different model family and on answering multi-hop questions and false premise questions. Overall, by addressing the hallucination problem, our work contributes to improving LLMs' reliability and trustworthiness, a crucial step en route to enabling their widespread adoption in real-world applications.

Chapter 4

IMPROVING RELIABILITY BY ADDRESSING THE HALLUCINATION PROBLEM IN TASKS INVOLVING NEGATION

Prior research has focused on investigating and addressing this problem for a variety of tasks such as biography generation, question answering, abstractive summarization, and dialogue generation. However, the crucial aspect pertaining to 'negation' has remained considerably underexplored. Negation is important because it adds depth and nuance to the understanding of language and is also crucial for logical reasoning and inference. In this chapter, we address the above limitation and particularly focus on studying the impact of negation in LLM hallucinations. Specifically, we study four tasks with negation: 'false premise completion', 'constrained fact generation', 'multiple choice question answering', and 'fact generation'. We show that open-source state-of-the-art LLMs such as LLaMA-2-chat, Vicuna, and Orca-2 hallucinate considerably on all these tasks involving negation which underlines a critical shortcoming of these models. Addressing this problem, we further study numerous strategies to mitigate these hallucinations and demonstrate their impact.

4.1 Introduction

Prior work has studied hallucination of LLMs in various scenarios such as open-ended text generation (Manakul, Liusie, and M. J. Gales 2023; Varshney, Yao, et al. 2023), question answering (Adlakha et al. 2023), abstractive summarization (Chrysostomou et al. 2023; Aralikatte et al. 2021; Cao, Dong, and Cheung 2022), machine translation (Feng et al. 2020), and dialogue generation (Dziri et al. 2021; Sun et al. 2023). While the above studies are important, investigating the impact of 'negation' in LLM hallucinations has remained underexplored. Negation is important because it adds depth and nuance to the understanding of language. It helps understand the opposite or absence of a statement, providing a more precise and nuanced interpretation and it is also crucial for logical reasoning and inference (Clark, Tafjord, and Richardson 2020; Talmor et al. 2020; Nakamura et al. 2023; Banerjee et al. 2020; Luo et al. 2023). Furthermore, we humans arguably use affirmative expressions (without negation) more often than expressions with negation (Hossain et al. 2020; Ettinger 2020); this implies that texts containing negation could be underrepresented in the training/tuning data of the models making it even more important to study.

With the aforementioned motivation, in this work, we focus on 'negation' and study its impact on LLM hallucinations. Prior work on negation has primarily studied classification tasks such as natural language inference and masked word prediction (Hosseini et al. 2021; Hossain et al. 2020; Hossain, Chinnappa, and Blanco 2022; Truong et al. 2023; Kassner and Schütze 2020). However, it is also important to study generative tasks with state-of-the-art LLMs. To this end, we study negation in four tasks: (i) *False Premise Completion* (FPC), (ii) *Constrained Fact Generation* (CFG), (iii) *Multiple-Choice Question Answering* (MCQA), and (iv) *Fact Generation* (FG).

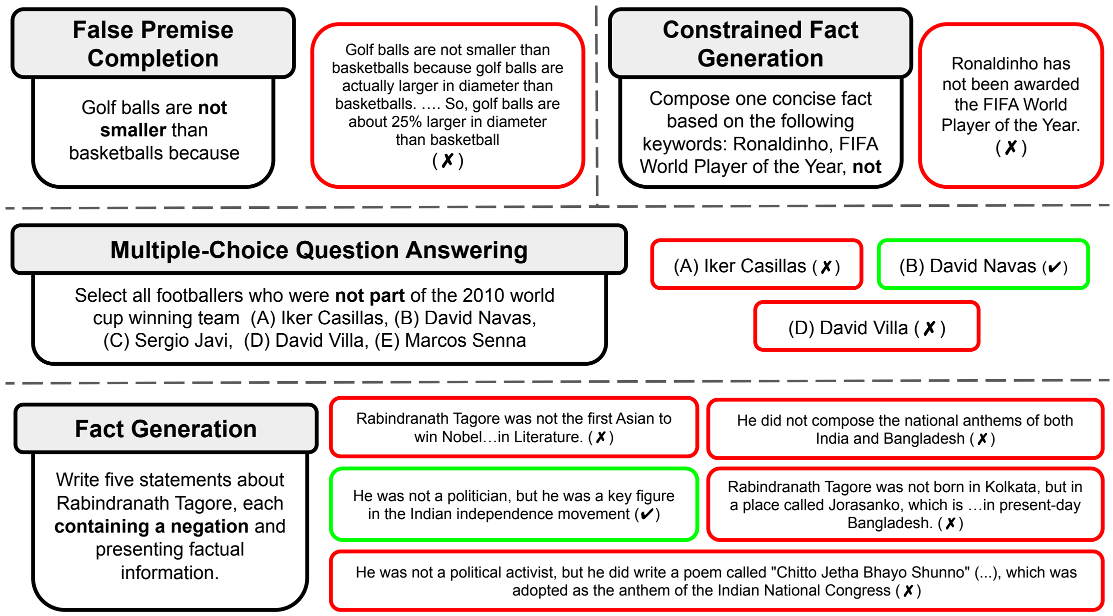


Figure 15. Illustration of the four tasks that deal with negation studied in this work. Responses enclosed in red boxes (marked with ✗) are hallucinations while those in green boxes (marked with ✓) are factually correct.

Figure 15 illustrates examples of all four tasks. We provide a detailed description and the rationale behind studying these tasks in Section 4.2.

We comprehensively study the performance of various open-source state-of-the-art LLMs including LLaMA-2-chat (Touvron et al. 2023), Vicuna-v1.5 (Chiang et al. 2023), and Orca-2 (Mitra et al. 2023). We show that these models hallucinate considerably on all the tasks. On average, they hallucinate 63.77%, 72.33%, 36.6%, and 62.59% on FPC, CFG, MCQA, and FG tasks respectively. This underlines a critical limitation of these LLMs in effectively dealing with negation.

To address this hallucination problem, we further study various mitigation strategies such as providing a 'cautionary instruction', demonstration via 'in-context exemplars', 'self-refinement' by leveraging the LLM's parametric knowledge, and ‘knowledge-augmented generation’. Our study results in numerous important findings such as (a) providing a ‘cautionary instruction’ along with ‘in-context exemplars’ performs the best in mitigating the hallucinations though there remains a considerable room for improvement, (b) providing contextual knowledge to the LLM when answering false premise prompts, coerces it to hallucinate even more instead of mitigation, (c) ‘self-refinement’ indeed mitigates the hallucinations to a certain extent; however, in some cases, it incorrectly transforms the output by introducing hallucinated information in the output.

Overall, our work underlines a critical shortcoming of existing LLMs and studies ways to mitigate it. We will release our work to further facilitate future research in developing robust LLMs that can effectively deal with negation.

4.2 Evaluation Tasks

In this section, we provide a detailed description and the rationale behind studying all the tasks.

4.2.1 False Premise Completion (FPC)

This task consists of prompts that involve negation (not) and are based on false premises, i.e., incorrect presuppositions. We (the authors) first compile a list of fundamental facts from various domains such as Science, Geography, Sports, Animals, and Astronomy and then introduce a negation (not) while ensuring the grammatical correctness to create false premise prompts. Table 18 shows examples of this task and the distribution of prompts across the different domains. For inference, we instruct the models to 'complete the given prompt by providing factually correct information'. Since the correct facts are negated, prompts in this task are factually incorrect; thus, a model needs to identify the false premise of the prompt and appropriately provide its response.

Consider a false premise prompt: "Saturn is not the second largest planet in our solar system because", we show that models often falter on such false premise prompts and generate hallucinated responses such as " *because it is actually the sixth largest planet in our solar system*"; however a robust model should respond to this false premise prompt with something like "*The statement in the prompt is incorrect because Saturn is indeed the second largest planet in our solar system, after Jupiter*". Note that we additionally study the performance on the corresponding correct premise prompts also as detailed in Section 4.3.1.

Rationale: We study this task because state-of-the-art models have been shown to perform well on a wide range of tasks that are based on correct presuppositions. However, users in real-world applications often tend to provide inputs that are based on false premises due to either the lack of relevant knowledge or to adversarially attack the system. Thus, the efficacy on this task is critical in preventing misinformation resulting from the hallucinated responses of the LLMs (Y. Pan et al. 2023). We attribute this kind of hallucination to the sycophantic behavior exhibited by LLMs (Sharma et al. 2023; Ranaldi and Pucci 2023).

Domain	Prompts
Science (39%)	The speed of sound is not affected by the medium through which it travels because Heat energy does not transfer from a warmer substance to a colder one because Hydrogen does not have atomic number of 1 because
Astronomy (20%)	Saturn is not the second largest planet in our solar system because Jupiter is not bigger than Earth because
Geography (13%)	The Sahara Desert does not have sand dunes because The Arctic region does not experience extreme cold temperatures because
Animals (8%)	Chickens do not lay eggs because Tigers are not carnivorous predators because
Sports (4%)	India did not win the 2011 world cup of cricket because Golf balls are not smaller than basketballs because
Tech. (3%)	Floppy disks do not have lower storage capacity than USB drives because
Others (9%)	Inflation does not decrease the purchasing power of money because The square root of 64 is not 8 because

Table 18. Examples of prompts for the FPC task.

4.2.2 Constrained Fact Generation (CFG)

This task requires composing a fact based on the given keywords one of which is a negation (not). Specifically, we use the following task instruction "Compose one concise fact based on the following keywords". Note that despite the presence of 'not' as a keyword, in all the instances of this task, there does indeed exist ways to compose factually correct responses from the provided keywords; however, a statement created by simply connecting 'not' with the other keywords (in a syntactically sound manner) will result in a factually incorrect sentence.

Domain	**Keywords**
Sports (40%)	Chris Froome, not, Tour de France Winner Sachin Tendulkar, not, Cricket World Cup, 2011 not, Luka Modric, Ballon d'Or Winner
Entertain (16%)	Luke Combs, not, Entertainer of the Year, CMA Awards not, Michael Jackson, Grammy Awards
Award (11%)	not, Ardem Patapoutian, Nobel Prize, 2021
Politics (13%)	Barack Obama, US Presidential Election, not, 2008
Others (13%)	The African Renaissance Monument, Senegal, tallest statue, not

Table 19. Examples of keywords for the CFG task.

Consider an example in which the keywords are "The African Renaissance Monument, Senegal, tallest statue, not", simply creating a sentence by combining the keywords would result in "The African Renaissance Monument statue in Senegal is not the tallest statue in Africa" which is factually incorrect; however, a possible correct output is "The African Renaissance Monument in Senegal, while being the tallest statue in Africa, is not the tallest statue in the world".

Thus, it poses an important challenge for the models and requires true understanding of negation to compose a factually correct statement. Here, we focus on historical facts from the domains of Sports, Awards such as Nobel prizes, Politics, and Entertainment. We particularly select these domains because information in these domains is unambiguously accurate and also easy to obtain and verify. Table 19 shows examples of this task. Note that we also vary the position of 'not' in the keyword list to avoid any bias in the models' outputs.

Rationale: This task has numerous applications in information retrieval and

Domain	Question
Sports (20%)	Choose the countries that have not hosted the Winter Olympics. Options: Finland, Austria, China, South Korea, USA Identify all the countries that have never played a FIFA World Cup Final. Options: Portugal, Belgium, USA, Germany, Argentina
Entertain (12%)	Pick the musicians who have not won a Grammy Award for Album of the Year. Options: Babyface, John Mayer, Ed Sheeran, Alanis Morissette, Taylor Swift Identify the films that have not won an Oscar for Best Film. Options: Anthony Adverse, The Irishman, Arrival The Lord of the Rings: The Return of the King, All the King's Men.
Geo. (27%)	Identify all European cities that are not capitals of their respective countries. Munich, Milan, Rome, Salzburg, Berlin Identify all African countries from which the Nile does not flow Options: Egypt, Burundi, Libya, Chad, Central African Republic

Table 20. Examples of questions for the MCQA task.

Sports	**Politics**	**Music**	**Films and TV**	**Science**	**Literature**
Cristiano Ronaldo	Xi Jinping	Michael Jackson	Rihanna	Albert Einstein	William Shakespeare
Lionel Messi	Vladimir Putin	The Beatles	Jackie Chan	Marie Curie	Akira Toriyama
Neymar Jr.	Donald Trump	Taylor Swift	Katy Perry	Isaac Newton	Georges Simenon
LeBron James	David Cameron	Miley Cyrus	Deepika Padukone	Galileo Galilei	Jin Yong
Virat Kohli	Narendra Modi	Justin Bieber	Jennifer Lopez	Satyendra Nath Bose	J. K. Rowling

Table 21. Names of personalities from six distinct domains considered in the study for FG task.

search engines because generating facts based on keywords, even when negation is involved, enhances the effectiveness of search engines and is vital for users seeking precise, relevant, and accurate information in a vast sea of data. This also has applications in automated content generation where users provide precise specifications to a generative system. It is also important to study this task for the prevention of misinformation from LLMs.

4.2.3 Multiple-Choice QA (MCQA)

In this task, a selection-based question involving negation is given along with multiple answer choices and the correct options that satisfy the question requirements need to be selected. Similar to the previous task, here, we focus on facts from the domains of Sports, Entertainment, Awards, etc. because these facts are unambiguously accurate and can be easily obtained and verified. Table 20 shows examples of this task. Note that this is a multi-choice multi-correct QA task where multiple answer options can be correct. In all the instances, we have a total of five answer options.

Rationale: This task is important in a variety of applications such as 'medical diagnosis' where a system might encounter statements like "the patient does not experience chest pain" and it needs to rule out/select certain options by understanding the statement, 'legal document analysis' where the system can help quickly sift through clauses based on a given statement, and 'customer service/sales chatbots' where sentences like "I don't want red color t-shirts' are commonly encountered.

4.2.4 Fact Generation (FG)

This task requires generating statements about personalities, each containing a negation and presenting factual information. To avoid any bias that may occur due to the lack of information, we include only widely known personalities. Also, we select these personalities from diverse domains such as Sports, Politics, Music, Films & TV, Science, and Literature. Specifically, we select five personalities from each domain from the Forbes popular list as shown in Table 21.

Rationale: This task is important in investigating misinformation which becomes very important when using LLMs to generate text about a person. Moreover, in a general sense, while comparing different options in decision-making, generating facts involving negation can help highlight the strengths and weaknesses of various options.

4.3 Experiments and Results

We experiment with various open-source state-of-the-art LLMs including LLaMA-2-chat (Touvron et al. 2023), Vicuna (Chiang et al. 2023), and Orca-2 (Mitra et al. 2023). We experiment with the 13B parameter models and the evaluation set contains 300 instances in FPC task (150 each of false premise and correct premise prompts as detailed in Section 4.3.1), 100 instances each in CFG and MCQA tasks, and 300 instances (5 each of without negation and with negation for all the 30 personalities) in FG task. Note that all the tasks described in Section 4.2 are focused on factuality; thus, we use the temperature of 0 during inference.

4.3.1 False Premise Completion

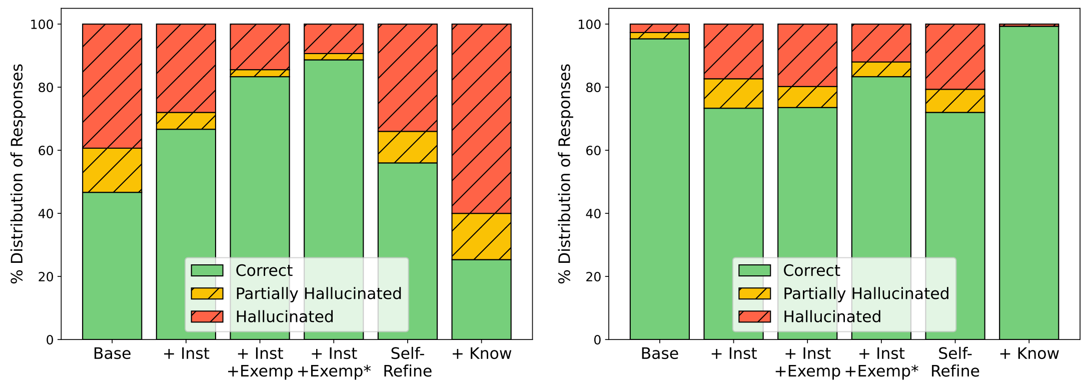
Performance Evaluation: To quantify the performance, we measure the percentage of hallucinated responses. Note that the performance evaluation on these tasks is challenging due to the existence of open-ended text responses to the inputs.

Response	LLaMA-2	Vicuna-v1.5	Orca-2
H	39.33%	58.67%	39.33%
PH	14.00%	19.33%	20.67%
C (↑)	46.67%	22.00%	40.00%

Table 22. Performance of models on the FPC task.

Thus, we (the authors) perform human evaluations for this task which are more reliable. However, we also study LLM-based evaluations (using the Bard model) to bolster the human evaluations.

In human evaluations, we classify a response to one of the following three categories: **Hallucinated (H)**, **Partially Hallucinated (PH)**, and **Correct (C)**. An 'H' category response fails to identify the false premise of the prompt and provides hallucinated (incorrect) information. A 'PH' category response fails to identify the false premise of the prompt and completes it but provides correct information at some place in the response (by contradicting itself). A 'C' category response indeed detects the false premise of the prompt and provides the correct information by rectifying it. Thus, 'C' category response is the ideal response for this task over 'PH' and 'H'. We note that this 3-level annotation is challenging to obtain from LLM-based evaluations; thus, we use only two levels (Correct and Incorrect) for LLM-based evaluations. We use the following instruction: "Check if the given text is factually accurate based on the fact that '{true fact}'. Answer in 'Correct' or 'Incorrect'. Text: {model's output}".



(a) On False Premise Prompts (b) On Correct Premise Prompts

Figure 16. Impact of various mitigation strategies with LLaMA-2 model on the Prompt Completion task. We show performance on both false premise prompts and correct premise prompts.

4.3.1.1 Performance of Models

Table 22 shows the performance of various models on the FPC task. Specifically, LLaMA-2, Vicuna-v1.5, and Orca-2 answer only 46.67%, 22%, and 40% instances correctly with label ‘C’. With BARD evaluations also, the correctness percentage is 44%. It demonstrates that all the models hallucinate considerably on this task. Table 24 shows examples of responses of various models. We attribute this poor performance to the sycophantic behavior exhibited by the LLMs where they tend to generate responses that favor the user’s perspective present in the input rather than providing correct or truthful answers. This result renders studying ways to mitigate these hallucinations important.

4.3.1.2 Mitigation of Hallucinations

To address the hallucination problem, we study a number of mitigation strategies on this task. Note that the mitigation strategies can potentially deteriorate the performance on the 'correct' premise prompts also. Thus, we also evaluate them on the **correct premise prompts**, i.e., the affirmative (non-negated) variants of the false premise prompts. Table 25 shows examples of false premise prompts and corresponding correct premise prompts.

Cautionary Instruction (Inst): State-of-the-art LLMs have shown a remarkable ability to follow natural language instructions. Thus, a way to mitigate hallucination is by providing a cautionary instruction in the input. Specifically, in addition to the task instruction ('Complete the given prompt by providing factually correct information'), we provide a cautionary instruction specifying 'Note that the prompt can be misleading as well'. We show that providing cautionary instruction indeed improves the performance to a certain extent.

Demonstrative Exemplars (Exemp): LLMs also possess an impressive ability to learn from demonstrations/exemplars provided in the input context. To this end, we provide exemplars of input-output pairs of negation prompts in the context. Note that we provide exemplars of negation prompts of both false premise and correct premise prompts so as to prevent biasing the models' outputs. We experiment with three different combinations of exemplars and report the averaged results. We use different combinations of the following exemplars:

(a) Input: Neptune is not wider than Earth because

Output: I must correct the statement: Neptune is indeed wider than Earth. Neptune is the eighth and farthest known planet from the Sun in our solar system, and it is a gas giant with a diameter of about 49,244 kilometers (30,598 miles).,

(b) Input: Australia did not win the 2007 cricket world cup because

Output: Australia did indeed win the 2007 Cricket World Cup. They defeated Sri Lanka in the final, held on April 28, 2007, at the Kensington Oval in Barbados.,

(c) Input: Rabindranath Tagore did not receive the Nobel Prize for Literature because

Output: Rabindranath Tagore did, in fact, receive the Nobel Prize for Literature. He was awarded the prestigious prize in 1913, making him the first Asian to be honored with a Nobel Prize.,

(d) Input: Neptune is not narrower than Earth because

Output: Neptune is not narrower than Earth because of fundamental differences in their compositions, sizes, and positions in the solar system. It's essential to clarify that Neptune is not narrower than Earth; in fact, it is significantly larger.,

(e) Input: Australia did not win the 2011 cricket world cup because

Output: Australia did not win the 2011 Cricket World Cup because they were eliminated in the quarterfinals of the tournament.,

(f) Input: Rabindranath Tagore did not receive the Nobel Prize for Physics because

Output: Rabindranath Tagore did not receive the Nobel Prize for Physics because his Nobel Prize, awarded in 1913, was specifically for Literature. Tagore was honored with the Nobel Prize in Literature for his collection of poems titled 'Gitanjali',

(g) Input: Neptune is narrower than Earth because

Output: I must correct the statement: Neptune is indeed wider than Earth. Neptune is the eighth and farthest known planet from the Sun in our solar system, and it is a gas giant with a diameter of about 49,244 kilometers (30,598 miles).,

(h) Input: Australia won the 2011 cricket world cup because

Output: Australia did not win the 2011 Cricket World Cup. The 2011 ICC Cricket World Cup was won by India.,

(i) Input: Rabindranath Tagore received the Nobel Prize for Physics because

Output: Rabindranath Tagore did not receive the Nobel Prize for Physics. He received the Nobel Prize for Literature. He was awarded the prestigious prize in 1913, making him the first Asian to be honored with a Nobel Prize.,

(j) Input: Neptune is wider than Earth because

Output: Neptune is indeed wider than Earth primarily because it is a larger planet in terms of both diameter and volume.,

(k) Input: Australia won the 2007 cricket world cup because

Output: Australia won the 2007 Cricket World Cup primarily due to their exceptional team performance throughout the tournament.,

(l) Input: Rabindranath Tagore received the Nobel Prize for Literature because

Output: He received this prestigious honor primarily for his collection of poems titled 'Gitanjali' (Song Offerings).,

For 'Inst + Exemp' strategy, we experiment with three different combinations of exemplars: [a,b,d,e], [a,c,d,f], and [a,b,e,f] and report averaged results.

For 'Inst + Exemp*' strategy, we experiment with the following examples [a,b,d,e,g,h,j,k].

We note that there is no overlap between the evaluation instances and the demonstrative exemplars.

Self-Refinement (Self-Refine): In self-refinement strategy, we first obtain the model's output and then instruct it to 'rewrite it by rectifying the factually incorrect information'. This method attempts to leverage the parametric knowledge of the model in rectifying the potential mistakes in its output (L. Pan et al. 2023).

Knowledge Augmentation (Know): Here, we provide knowledge relevant to the prompt as additional contextual information to the LLM during generation. We use web search via Bing search API to obtain the relevant knowledge. Specifically, we use the input prompt as the query to retrieve the web search results.

Table 23 shows examples of knowledge retrieved for various prompts. We use snippets returned by Bing Search API of two search results as knowledge.

Performance of Mitigation Strategies: Figure 16(a) shows the effectiveness of various mitigation strategies on the LLaMA-2 model's performance. The bar corresponding to 'Base' refers to the base setting without any mitigation strategy. In 'Inst' strategy, we add a cautionary instruction, and in 'Inst + Exemp', we also add demonstrative exemplars. 'Inst + Exemp*' corresponds to the strategy where we provide exemplars of both negated and non-negated prompts. The non-negated prompts exhibit just a slight impact on the false premise prompts; however, they play a crucial role on the correct premise prompts where we study the downside of these mitigation strategies (later in this Subsection).

Prompt	Knowledge
Jupiter is not bigger than Earth because	Jupiter: Facts - NASA Science, Quick Facts Eleven Earths could fit across Jupiter's equator. If Earth were the size of a grape, Jupiter would be the size of a basketball. Jupiter orbits about 484 million miles (778 million kilometers) or 5.2 Astronomical Units (AU) from our Sun (Earth is one AU from the Sun)... Jupiter - Wikipedia, Formation and migration Jupiter is believed to be the oldest planet in the Solar System, having formed just one million years after the Sun and roughly 50 million years before Earth. [23] ...
Metals are not a good conductor of heat because	7.6: Metals, Nonmetals, and Metalloids - Chemistry LibreTexts, Valency: Metals typically have 1 to 3 electrons in the outermost shell of their atoms. Conduction: Metals are good conductors because they have free electrons. Silver and copper are the two best conductors of heat and electricity. Lead is the poorest conductor of heat. Bismuth, mercury and iron are also poor conductors ... 2.11: Metals, Nonmetals, and Metalloids - Chemistry LibreTexts, Conduction: Metals are good conductors because they have free electrons. Silver and copper are the two best conductors of heat and electricity. Lead is the poorest conductor of heat. Bismuth, mercury and iron are also poor conductors; Density: Metals have high density and are very heavy. Iridium and osmium have the highest densities where as ...

Table 23. Examples of knowledge retrieved by using the corresponding prompt as the search query.

It can be observed that all the strategies except 'knowledge augmented generation' result in considerable improvements in reducing hallucinations. Table 26 shows examples of responses after application of various mitigation strategies on the false premise prompts. We also analyzed the improvement of exemplars strategies and attribute their performance to the ability to counter the false premise prompt acquired from the in-context exemplars. Also, we observe negligible deterioration (change from correct to incorrect) on the false premise prompts (except 'Know' strategy) due to the mitigation strategies.

Knowledge coerces hallucination on false premise prompts: Knowledge considerably increases the hallucination on the false premise prompts. We attribute this to the nature of the prompts, i.e., providing additional contextual knowledge coerces the model to respond to a prompt even when the prompt is misleading; which increases the hallucination percentage. Table 23 shows examples of this result. This is an important result because knowledge-augmented generation is typically considered to improve performance; however, we show that on false premise prompts, it instead proves to be detrimental. However, as expected, knowledge helps in answering the correct premise prompts as we show in the next study.

Impact of mitigation strategies on the correct premise prompts: Note that this study is crucial to highlight the negative impact of the mitigation strategies. Figure 16(b) shows the performance of various mitigation strategies on the correct premise prompts. Without any mitigation strategy ('Base'), the model correctly answers nearly all the instances. This is because the correct prompts are based on fundamental facts. However, all strategies barring 'Know' deteriorate the performance by hallucinating on the correct premise prompts. This highlights an important downside of the mitigation strategies. Unsurprisingly, 'Know' does well on the correct premise prompts, However, as noted before, it doesn't fair well on the false premise prompts where it coerces hallucination. Interestingly, self-refinement also deteriorates the performance to a slight extent on the correct premise prompts. This is because during refinement, the model instead introduces hallucinations in the output. We observe that in most of the deterioration cases, the model transformed the correct response by incorrectly introducing 'not' into it. Table 28 shows examples of outputs of various strategies on the correct premise prompts. Overall, 'Inst + Exemp*'

	Prompt/Model Response
False Premise	Sound does not travel through air because
LLaMA-2	Sound does not travel through air because air is a poor conductor of sound waves. Sound waves ... (✗)
Vicuna	Sound does not travel through air because sound waves are mechanical waves that require a medium to propagate. In the case of air, sound waves cannot travel because air is an incompressible fluid, meaning it does not have the ability to vibrate and transmit sound waves. (✗)
Orca-2	Sound does not travel through air because it is a mechanical wave that requires a medium to propagate. In the case of air, it is not dense enough to transfer sound waves efficiently. (✗)
False Premise	Metals are not a good conductor of heat because
LLaMA-2	Metals are not a good conductor of heat because they have a low thermal conductivity. Thermal conductivity is a measure of how easily ... (✗)
Vicuna	Metals are not a good conductor of heat because their atoms are not loosely packed, which makes it difficult for heat to travel through them. ... (✗)
Orca-2	Metals are not a good conductor of heat because they have a low thermal conductivity. This means that they do not transfer heat efficiently from one part to another. The atoms in metals are closely packed together, which makes it difficult for heat to move through them ... (✗)

Table 24. Examples of responses of various models on the false premise prompts.

performs the best out of all the mitigation strategies as it reduces the hallucination on the false premise prompts while causing lower deterioration (relatively) on the correct premise prompts.

Table 24 shows examples of responses of various models on the false premise prompts. Table 25 shows examples of false premise prompts and corresponding correct premise prompts.

Table 26 shows examples of responses after application of various mitigation strategies on the false premise prompts. Table 27 shows examples of responses after application of 'Know' mitigation strategy on the false premise prompts.

False Premise Prompt	**Correct Premise Prompt**
Saturn is not the second largest planet in our solar system because	Saturn is the second largest planet in our solar system because
Jupiter is not bigger than Earth because	Jupiter is bigger than Earth because
Golf balls are not smaller than basketballs because	Golf balls are smaller than basketballs because
Metals are not a good conductor of heat because	Metals are a good conductor of heat because
Tennis balls do not have a smaller diameter than soccer balls because	Tennis balls have a smaller diameter than soccer balls because
Heat energy does not transfer from a warmer substance to a colder one because	Heat energy transfers from a warmer substance to a colder one because
Hydrogen does not have atomic number of 1 because	Hydrogen has atomic number of 1 because
Inflation does not decrease the purchasing power of money because	Inflation decreases the purchasing power of money because
Dolphins are not mammals because	Dolphins are mammals because

Table 25. Examples of false premise prompts with corresponding correct premise prompts.

Table 28 shows examples of responses after application of various mitigation strategies on the correct premise prompts.

We also that it is also plausible to correctly complete the negated prompt by providing correct supporting details. Consider another prompt, "Trees do not produce oxygen because", a response mentioning "Trees do not produce the same amount of oxygen year-round because oxygen production is dependent on sunlight and other factors" is correct; in contrast, a response "Trees do not produce because they instead produce nitrogen during photosynthesis" is incorrect.

Self-Checking Analysis For further analysis, we additionally evaluate the ability of the models to correctly check the factuality of the prompts. To this end, we experiment with the following three different variants of the input: "Is the given

Prompt/Strategy	Prompt/Model Response
Prompt	Saturn is not the second largest planet in our solar system because
Base	Saturn is not the second largest planet in our solar system because it is actually the sixth largest planet in our solar system. (✗)
Self-Refinement	Saturn is not the second largest planet in our solar system because it is actually the sixth largest planet in our solar system. (✗)
Inst + Exemp	Saturn is, in fact, the second largest planet in our solar system, with a diameter of approximately 116,464 kilometers (72,337 miles). The largest planet in our solar system is Jupiter, which has a diameter of around ...(✓)
Prompt	Golf balls are not smaller than basketballs because
Base	Golf balls are not smaller than basketballs because golf balls are actually larger in diameter than basketballs. A standard golf ball is approximately 1.68 inches (4.27 cm) in diameter, while a standard basketball is approximately 1.21 inches (3.07 cm) in diameter. So, golf balls are about 25% larger in diameter than basketball (✗)
Self-Refinement	Golf balls are not larger than basketballs because golf balls are actually smaller in diameter than basketballs. (✓)
Inst	Contrary to popular belief, golf balls are actually larger than basketballs. (✗)
Inst + Exemp*	Golf balls are actually larger than basketballs. A standard golf ball is about 4.2 cm (1.65 inches) in diameter, while a standard basketball is about 2.5 cm (0.98 inches) in diameter. (✗)

Table 26. Examples of responses after application of various mitigation strategies on the false premise prompts.

prompt factually correct?", "Does the provided prompt contain factually accurate information?", and "Is the information presented in the prompt factually true?".

We provide both false premise and correct premise prompts as input. The averaged accuracy of the LLaMA model on this task is 62.7% just slightly above the random baseline. This shows the limitation of the model in self-checking the factuality of the prompt.

Prompt/Strategy	Prompt/Model Response
Prompt	The Sun does not emit light because
Base	the statement "The Sun does not emit light because" is not entirely accurate. The Sun does emit light, and in fact, it is the primary source of light and energy for our solar system. The Sun emits light through a process called nuclear fusion, where hydrogen atoms are fused together to form helium, releasing a vast amount of energy in the form of light and heat. This process occurs at the Sun's core. (✓)
Know	The Sun does not emit light because it is an ordinary star and does not have the ability to produce light on its own. (✗)
Prompt	The speed of sound is not affected by the medium through which it travels because
Base	Sorry, it's not accurate to say that the speed of sound is not affected by the medium through which it travels. In fact, the speed of sound can vary depending on the medium it is traveling through. The speed of sound is affected by the properties of the medium it is traveling through, such as its density, elasticity, and temperature. For example, sound travels faster through solids than through liquids, and faster through liquids than through (✓)
Know	The speed of sound is not affected by the medium through which it travels because the speed of sound is a fundamental physical constant, and it remains constant regardless of the medium through which it travels. (✗)

Table 27. Examples of responses after application of 'Know' mitigation strategy on the false premise prompts.

4.3.2 Constrained Fact Generation

For both the fact generation tasks (CFG and FG), we evaluate the factual correctness of the model's output. For this evaluation, we use BARD (Gemini) model as it utilizes web search results to generate its output. Note that having web search access further assists in getting accurate evaluations for these tasks as they involve fact checking. However, we also perform human annotations to measure the accuracy

Prompt/Strategy	Prompt/Model Response
Prompt	Golf balls are smaller than basketballs because
Base	Golf balls are smaller than basketballs because golf balls are designed to be smaller and lighter to allow for longer distances and more accurate shots, while basketballs are larger and heavier to provide more stability and control during play.(✓)
Self-Refinement	Golf balls are not smaller than basketballs. In fact, golf balls are slightly larger than basketballs. This is because golf balls are designed to be more aerodynamic and have a higher lift ...(✗)
Know	Golf balls are smaller than basketballs because golf balls are regulation size 5 balls, which have a circumference of 68cm to 70.5cm and weigh between 445g and 475g when fully inflated. In comparison, basketballs are typically size 7 or size 8, ... (✓)
Prompt	Snakes are reptiles because
Base	Snakes are reptiles because they have several characteristics that are common to all reptiles, such as: 1. Cold-bloodedness: Snakes, like all reptiles, are ectothermic, meaning their body temperature is regulated by the environment around them, rather than being able to generate heat internally ... (✓)
Inst + Exemp	Snakes are not reptiles because they do not have legs or external ears, which are characteristics of reptiles ... (✗)
Self-Refinement	Snakes are not reptiles because they do not have several characteristics that are common to all reptiles, such as: 1. Cold-bloodedness: Snakes, unlike all reptiles, are not ectothermic, meaning their body temperature is not regulated by the environment around them, but rather internally. ... (✗)

Table 28. Examples of responses after application of various mitigation strategies on the correct premise prompts.

Models	LLaMA-2	Vicuna-v1.5	Orca-2
Hallucination (↓)	72%	73%	73%

Table 29. Hallucination % of models on the CFG task.

of BARD in evaluating the correctness and find it to be highly reliable (Appendix 4.4).

Table 29 shows the hallucination percentage of models on this task. It shows that models falter significantly on this task as they generate a large percentage of

Input/Model	Keywords/Model Response
Keywords	Michael Clarke, Cricket World Cup, not, 2015
LLaMA-2	Michael Clarke did not play in the 2015 Cricket World Cup. (✗)
Vicuna	Michael Clarke did not participate in the 2015 Cricket World Cup. (✗)
Orca-2	Michael Clarke did not play in the 2015 Cricket World Cup due to a hamstring injury. (✗)
Possible Correct Generation	Michael Clarke did not score a century in the 2015 Cricket World Cup. (✓)
Keywords	not, Finland, Men's Ice Hockey World Championships
LLaMA-2	Finland has never won the Men's Ice Hockey World Championships, despite being one of the top teams in the tournament's history. (✗)
Vicuna	Finland did not win the Men's Ice Hockey World Championships in 2021. (✓)
Orca-2	Finland did not qualify for the Men's Ice Hockey World Championships in 2021. (✗)

Table 30. Examples of responses of various models on the CFG task.

hallucinated responses. Table 30 shows examples of responses of various models on this task. We note that highlights a sycophantic behavior as the models tend to concoct a fact by simply combining the given keywords. This is a major limitation because it can potentially propagate misinformation.

Table 30 shows examples of responses of various models on the CFG task. Though the scope of this project is limited to open-source 13B models, we also evaluate GPT-4 model on this task and found that even GPT-4 hallucinates on 60% instances. This result further opens up avenues for future research in evaluating and improving the ability of LLMs to effectively deal with negation. up several avenues for new research Table 31 provides examples of GPT-4 responses.

Input/Model	Keywords/Model Response
Keywords	8, not, Oxygen, Atomic Number
GPT-4	The atomic number of Oxygen is not 8. (✗)
Keywords	Roger Federer, not, Wimbledon
GPT-4	Roger Federer has not won Wimbledon since 2017.(✓)
Keywords	2016, Denver Broncos, Super Bowl, Not
GPT-4	In 2016, the Denver Broncos did not lose the Super Bowl; they won it.(✓)
Keywords	Claire Foy, not, emmy awards, 2018
GPT-4	Claire Foy did not win an Emmy Award in 2018.(✗)

Table 31. Examples of responses of GPT-4 on the CFG task.

Models	Baseline	LLaMA-2	Vicuna-v1.5	Orca-2
Perf. (↑)	51.4%	62.2%	54%	74%

Table 32. Performance of models on the MCQA task.

4.3.3 Multiple-Choice QA

Performance Evaluation: In this task, we use the following performance metric:

$$\frac{c_r + i_{\neg r}}{\#\texttt{options}}$$

where c_r is the number of correct answer options in the response, $i_{\neg r}$ is the number of incorrect answer options not in the response, and #`options` is the total number of answer options.

Table 32 shows the performance of various models on this task. The table also shows a baseline system performance that corresponds to the system that includes all the answer options in its response; thus its performance equals to the number of correct options divided by the total number of options. Orca-2 performs relatively

Question **Options**	Identify the Formula 1 drivers who have not won the Monaco Grand Prix. (A) Max Verstappen, (B) Ayrton Senna, **(C) Carlos Sainz**, (D) Lewis Hamilton, **(E) Lando Norris**
Orca-2	The question is asking for the Formula 1 drivers who have not won the Monaco Grand Prix. The options provided are Max Verstappen, Ayrton Senna, Carlos Sainz, Lewis Hamilton, and Lando Norris. First, let's eliminate the drivers who we know have won the Monaco Grand Prix. Ayrton Senna and Lewis Hamilton are both well-known for their success in Formula 1, including multiple Monaco Grand Prix wins. Therefore, we can eliminate options (B) and (D). Now, let's consider the remaining options: Max Verstappen, Carlos Sainz, and Lando Norris. All three of these drivers are currently active in Formula 1, and it is possible that they have not yet won the Monaco Grand Prix. However, Max Verstappen has won the Monaco Grand Prix in the past, so we can eliminate option (A). This leaves us with two options: Carlos Sainz and Lando Norris. Both of these drivers are still active in Formula 1, and as of the time of this writing, neither of them has won the Monaco Grand Prix. Therefore, both options (C) and (E) are correct. ### Final Answer: (C) Carlos Sainz, (E) Lando Norris

Table 33. Examples of Orca-2 responses on the MCQA task.

better than other models on this task. This is because of its tuning methodology which is based on 'explanation tuning', therefore, it explicitly tries to reason over all the options and then produces the final answer. Table 33 shows examples of responses from Orca-2 on this task. We also calculate the average number of answer options in the responses of all the models. Specifically, LLaMA-2, Vicuna, and Orca-2 have 3.11, 2.7, and 3.84 options in their respective responses and the average number of correct responses is 2.57.

Orca-2 performs relatively better than other models on this task. This is because of its tuning methodology which is based on 'explanation tuning', therefore, it explicitly

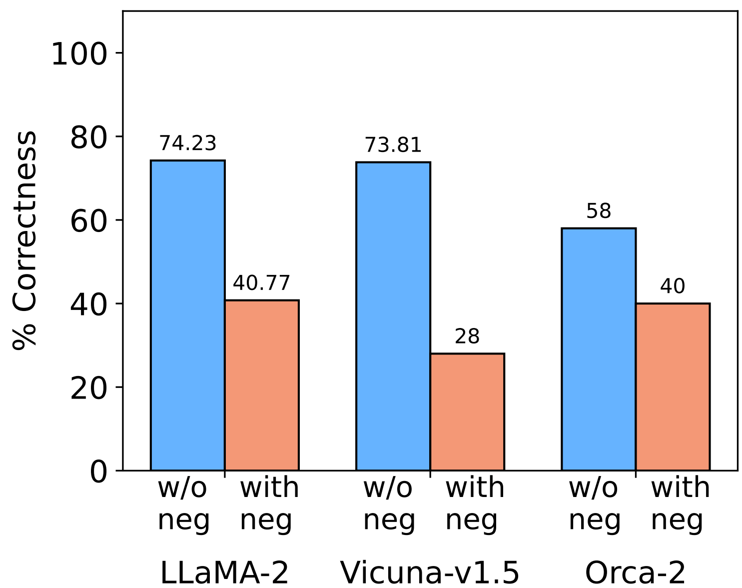


Figure 17. Performance of models on the FG task with negation (w/ neg) and without negation (w/o neg).

tries to reason over all the options and then produces the final answer. Table 33 shows examples of responses from Orca-2 on the MCQA task.

4.3.4 Fact Generation

Since LLMs are known to be brittle and sensitive to prompts, we experiment with three different prompts for this task. Furthermore, to compare models' ability to generate facts *involving* and *not involving* negation, we also generate facts using the following prompts: (a) 'Write five facts about {topic}. Each statement should be factually correct.' (b) 'Write five accurate statements about {topic}.' (c) 'Share five true facts about {topic}.'

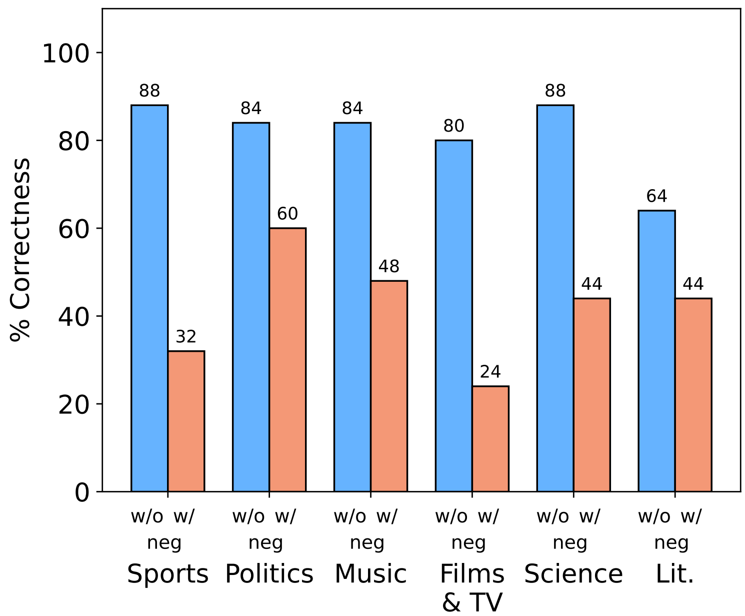


Figure 18. Domain-wise performance of LLaMA-2 on the FG task with negation and without negation.

Figure 17 shows the performance of models for both 'with negation' prompts (w/ neg) and 'without negation' prompts (w/o neg). On average, in the 'w/o neg' setting, the hallucination percentage is 25.77%, 26.19%, and 42% for the three models respectively while on the 'w/ neg' scenario, the hallucination percentage increases to 59.23%, 72%, and 60% for the three models. This shows the models hallucinate considerably higher in generating facts containing negation. We further show this comparison on each domain for the LLaMA-2 model in Figure 18. The same finding holds true across all the domains. Table 34 shows examples of facts generated for both 'without negation' and 'with negation' prompts.

Since LLMs are known to be brittle and sensitive to prompts, we experiment with three different prompts for this task: (a) 'Write five statements about {topic},

Name	Without Negation	With Negation
Virat Kohli	Kohli has won numerous awards for his cricketing achievements, including the ICC Cricketer of the Year award in 2017 and 2018. (✓) He is a right-handed batsman and has been ranked as the number one Test batsman in the world by the International Cricket Council (ICC) on several occasions. (✓) He has scored over 11,000 runs in One Day Internationals (ODIs) and has a batting average of over 50 in this format. (✓)	Kohli has never scored a double century in Test cricket, despite having a career average of over 50 in the format. (✗) 4. Virat Kohli has never won the ICC Cricketer of the Year award, although he has been nominated several times. (Negation: has never won) (✗) Virat Kohli does not have a perfect record in Test cricket, having been dismissed for a duck (zero) on several occasions.(✓)

Table 34. Examples of facts generated by LLaMA-2 for both 'without negation' and 'with negation' prompt. (✗) and (✓) correspond to hallucinated and factually correct statements respectively.

each containing a negation and presenting factual information.' (b) 'Compose five factually accurate statements that include negation related to {topic}.' (c) 'Write five factual statements about {topic}, all of which include negation.'

Furthermore, to compare models' ability to generate facts involving and not involving negation, we also generate facts using the following prompts: (a) 'Write five facts about {topic}. Each statement should be factually correct.' (b) 'Write five accurate statements about {topic}.' (c) 'Share five true facts about {topic}.'

Table 34 shows examples of facts generated by LLaMA-2 for both 'without negation' and 'with negation' prompt.

4.4 BARD Evaluations

For both the fact generation tasks (CFG and FG), we evaluate the factual correctness of the model's output. For this evaluation, we use BARD (Gemini) model as it utilizes web search results to generate its output. Note that having web search access further assists in getting accurate evaluations for these tasks as they involve checking factual correctness. Furthermore, BARD performs remarkably well in providing nuanced explanations while verifying the facts. It is able to dissect and consider every aspect of the response to construct the overall evaluation of the correctness of the statements. However, we also perform human annotations to measure the accuracy of BARD in evaluating the correctness. Table 35 shows examples of responses given by BARD.

4.5 Conclusion

Motivated by the importance of 'negation' in language, we comprehensively studied hallucinations of LLMs with four evaluation tasks that deal with negation. Specifically, we studied false premise completion, constrained fact generation, multiple-choice QA, and fact generation tasks. Through extensive experiments, we demonstrated that open-source state-of-the-art LLMs including LLaMA-2-chat, Vicuna-v1.5, and Orca-2 hallucinate considerably on all these tasks. This highlights a critical shortcoming of these models. Addressing this hallucination problem, we also studied various mitigation strategies and analyzed their effectiveness. Our analysis also revealed an

Text	BARD Response
Cristiano Ronaldo was born on February 5, 1985, in Funchal, Madeira, Portugal.	This statement is correct. Here's why: Multiple credible sources report Cristiano Ronaldo's birth date and place as February 5, 1985, in Funchal, Madeira, Portugal. These sources include: Wikipedia: https://en.wikipedia.org/wiki/Cristiano_Ronaldo Britannica: https://www.britannica.com/biography/Cristiano-Ronaldo Biography: https://www.biography.com/athletes/cristiano-ronaldo Ronaldo himself has confirmed his birth date and place in interviews and on social media. Therefore, based on the available evidence, the claim that Cristiano Ronaldo was born on February 5, 1985, in Funchal, Madeira, Portugal, is correct.
Cristiano Ronaldo has never represented Portugal in a World Cup tournament.	The statement that Cristiano Ronaldo has never represented Portugal in a World Cup tournament is incorrect. He has actually represented Portugal in five World Cups: 2006, 2010, 2014, 2018, and 2022. In fact, he is the all-time leading scorer for Portugal in the World Cup with eight goals.

Table 35. Examples of BARD evaluations.

important negative impact of employing these mitigation strategies. In summary, our work in this chapter underlined a critical shortcoming of existing LLMs pertaining to their ability to deal with negation and studied ways to address that limitation.

Chapter 5

IMPROVING RELIABILITY BY ABSTAINING FROM MAKING INCORRECT PREDICTIONS VIA SELECTIVE PREDICTION

NLP models are not absolutely perfect, i.e., they often make incorrect predictions. Incorrect predictions hamper the reliability of systems. Selective prediction partly addresses these concerns by enabling systems to abstain from making predictions when they are likely to be incorrect. Avoiding incorrect predictions allows them to maintain high task accuracy and thus makes them more reliable. In this chapter, we study various selective prediction approaches such as maximum softmax probability, monte-carlo dropout, label smoothing, and calibration in in-domain, out-of-domain, and adversarial settings.

5.1 Introduction

Despite impressive progress made in Natural Language Processing (NLP), it is unreasonable to expect models to be perfect in their predictions. They often make incorrect predictions, especially when inputs tend to diverge from their training data distribution (Elsahar and Gallé 2019; Miller et al. 2020; Koh et al. 2021). While this is acceptable for tolerant applications like movie recommendations, high risk associated with incorrect predictions hinders the adoption of these systems in real-world safety-critical domains like biomedical and autonomous robots. In such scenarios, *selective* *prediction* becomes crucial as it allows maintaining high accuracy by abstaining on instances where error is likely.

Selective Prediction (SP) has been studied in machine learning (Chow 1957; El-Yaniv et al. 2010) and computer vision (Geifman and El-Yaniv 2017, 2019), but has only recently gained attention in NLP. Kamath, Jia, and Liang 2020 proposed a post-hoc calibration-based SP technique for Question-Answering (QA) datasets. Garg and Moschitti 2021 distill the QA model to filter out error-prone questions. Unfortunately, despite the shared goal of making NLP systems robust and reliable for real-world applications, SP has remained underexplored; the community does not know which techniques work best across tasks/settings or even if they consistently outperform the simplest baseline *MaxProb* (Hendrycks and Gimpel 2017) (that uses a threshold over the maximum softmax probability for selective prediction).

In this work, we address the above point and study selective prediction in a large-scale setup of 17 datasets spanning over Natural Language Inference (NLI), Duplicate Detection, and QA tasks. Our comprehensive experiments under In-Domain (IID), Out-Of-Domain (OOD), and Adversarial (ADV) settings result in the following findings:

1. None of the existing SP approaches consistently and considerably outperforms *MaxProb*.

   Slight improvement in IID: Most of the approaches outperform MaxProb in the IID setting; however, the magnitude of improvement is very small (Figure 19). For instance, *MCD* achieves an average improvement of just 0.28 on AUC value across all NLI datasets.

Negligible improvement in OOD: The magnitude of improvement in OOD is even lesser (0.08) than that observed in the IID (Figure 20a). In a few cases, we also observe performance degradation (higher AUC than MaxProb).

Performance degradation in ADV: Most of the approaches fail to even match the MaxProb's performance in ADV setting (Figure 20b). For instance, *MCD* degrades the AUC value by 1.76 on Duplicate Detection datasets and *Calibration* degrades by 1.27 on NLI datasets.

2. **Approaches do not translate well across tasks**: We find that a single approach does not achieve the best performance across all tasks. For instance, *MCD* outperforms all other approaches on Duplicate Detection datasets but does not fare well on the NLI datasets.

3. **Existing approaches fail to outperform MaxProb despite leveraging additional resources**: *MCD* requires additional computation (for multiple inferences) while *calibration*-based approaches require a held-out dataset. In contrast, *MaxProb* does not require any such resources and still outperforms them, especially in the ADV setting.

Overall, our results highlight that there is a need to develop stronger selective prediction approaches that perform well across tasks while being computationally efficient.

Usually, the last layer of models has a softmax activation function that gives the probability distribution $P(y)$ over all possible answer candidates Y. Y is the set of labels for classification tasks, answer options for multiple-choice QA, all input tokens (for start and end logits) for extractive QA, and all vocabulary tokens for generative tasks. Thus, predictor f is defined as: $\underset{y \in Y}{\operatorname{argmax}} P(y)$

Maximum Softmax Probability (MaxProb): Hendrycks and Gimpel 2017 introduced a simple method that uses the maximum softmax probability across all answer candidates as the confidence estimator $\tilde{g}$ i.e. $\max_{y \in Y} P(y)$

Monte-Carlo Dropout (MCD): Gal and Ghahramani 2016 proposed to infer a test input multiple times using different dropout masks and ensemble them to get the confidence estimate.

Label Smoothing (LS): Szegedy et al. 2016 proposed to compute cross-entropy loss value with a weighted mixture of target labels during training instead of one hot 'hard' label. This prevents the network from becoming over-confident in its predictions.

Calibration (Calib): In calibration, a held-out dataset is annotated conditioned on the correctness of the model's predictions (correct as 'positive' class and incorrect as 'negative' class), and another model (calibrator) is trained on this annotated binary classification dataset. Softmax probability assigned to the positive class by this trained calibrator is used as the confidence estimator for SP. Kamath, Jia, and Liang 2020 study a calibration-based SP technique for Question Answering datasets. They train a random forest model using features such as input text length and probabilities of top 5 predictions and use it as a calibrator. We refer to this approach as **Calib C**. Inspired by the calibration technique presented in (Z. Jiang et al. 2021), we also train calibrator as a regression model (**Calib R**) by annotating the heldout instances on a continuous scale instead of categorical labels 'positive' and 'negative' (unlike the annotation done in Calib C). We compute these annotations using MaxProb as:

$$s = \begin{cases} 0.5 + \frac{maxProb}{2}, & \text{if correct} \\ 0.5 - \frac{maxProb}{2}, & \text{otherwise} \end{cases}$$

Furthermore, we train a transformer-based model for calibration (**Calib T**) that leverages the entire input text for training instead of features derived from it (Garg and Moschitti 2021).

5.3 Experimental Setup

5.3.1 Tasks and Settings:

We conduct comprehensive experiments with 17 datasets spanning over Natural Language Inference (NLI), Duplicate Detection, and Question-Answering (QA) tasks and evaluate the efficacy of various selective prediction approaches in IID, OOD, and adversarial (ADV) settings.

NLI: We train our models with SNLI (Bowman et al. 2015) / MNLI (Williams, Nangia, and Bowman 2018) / DNLI (Welleck et al. 2019) and use HANS (McCoy, Pavlick, and Linzen 2019) , Breaking NLI (Glockner, Shwartz, and Goldberg 2018), NLI-Diagnostics (A. Wang et al. 2018) , Stress Test (Naik et al. 2018) as adversarial datasets. While training with SNLI, we consider SNLI evaluation dataset as IID and MNLI, DNLI datasets as OOD. Similarly, while training with MNLI, we consider SNLI and DNLI datasets as OOD.

Duplicate Detection: We train with QQP (Iyer, Dandekar, and Csernai 2017) / MRPC (Dolan and Brockett 2005) and use PAWS-QQP, PAWS-Wiki (Zhang, Baldridge, and He 2019) as adversarial datasets.

QA: We train with SQuAD (Rajpurkar et al. 2016) and evaluate on NewsQA (Trischler et al. 2017), TriviaQA (Joshi et al. 2017), SearchQA (Dunn et al. 2017), HotpotQA (Z. Yang et al. 2018), and Natural Questions (Kwiatkowski, Palomaki, Redfield, Collins, Parikh, Alberti, Epstein, Polosukhin, Devlin, Lee, et al. 2019b).

5.3.2 Training Details:

We run all our experiments using *bert-base* model (Devlin et al. 2019) with batch size of 32 and learning rate ranging in $\{1\text{–}5\}e{-}5$. All experiments are done with Nvidia V100 16GB GPUs.

Calibration: For calibrating QA models, we use input length, predicted answer length, and softmax probabilities of top 5 predictions as the features (similar to (Kamath, Jia, and Liang 2020)). For calibrating NLI and Duplicate Detection models, we use input lengths (of premise/sentence1 and hypothesis/sentence2), softmax probabilities assigned to the labels, and the predicted label as the features. We train calibrators using random forest implementations of Scikit-learn (Pedregosa et al. 2011)

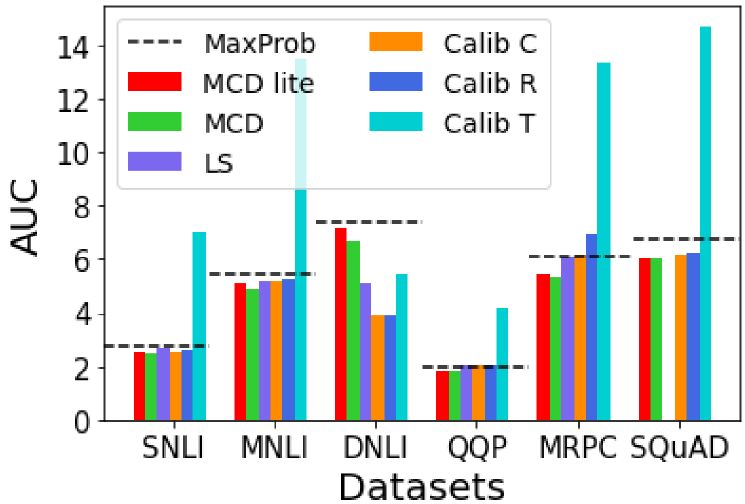


Figure 19. Comparing AUC of risk-coverage plot of various SP approaches with MaxProb in IID settings.

for Calib C and Calib R approaches, and train a bert-base model for Calib T. In all calibration approaches, we calibrate using the IID held-out dataset and use softmax probability assigned to the positive class as the confidence estimate for SP.

Label Smoothing: For LS, we use MaxProb of the model trained with label smoothing as the confidence estimator for SP. To the best of our knowledge, LS is designed for classification tasks only. Hence, we do not evaluate it for QA tasks.

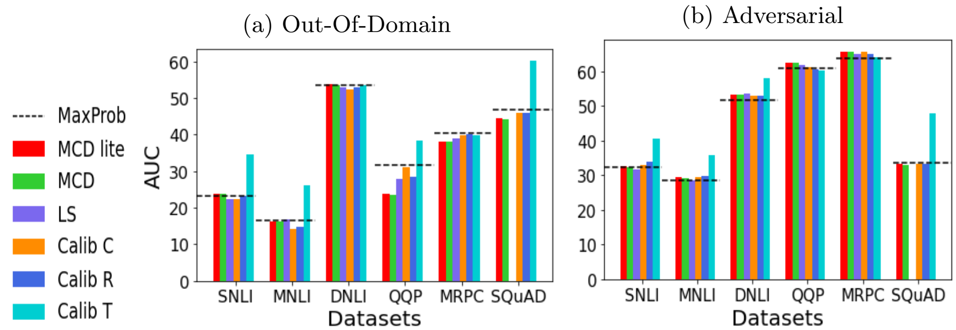


Figure 20. Comparing AUC of risk-coverage plot of various approaches with MaxProb in OOD and ADV settings. The results have been averaged over all the task-specific OOD/ADV datasets mentioned in Section 5.3 to highlight the general trend.

5.4 Results and Analysis

5.4.1 Slight Improvement in IID

We compare the selective prediction performance of various approaches in the IID setting in Figure 19. Though all the approaches except Calib T outperform MaxProb in most cases, the magnitude of improvement is very small. For instance, MCD achieves an average AUC improvement of just 0.28 across all NLI datasets.

Calib C and Calib R achieve the highest improvement on DNLI: *We find that these approaches benefit from using the predicted label as a feature for calibration.* Specifically, the model's prediction accuracy varies greatly across labels (0.94, 0.91, and 0.76 for entailment, contradiction, and neutral predictions respectively). This implies when the model predicts the label to be neutral, it is relatively less likely to be correct as compared to the scenario when the prediction is entailment or contradiction.

Train On	Method	IID↓	OOD avg.↓	ADV avg.↓
QQP	**MaxProb**	2.0	31.72	60.9
	MCD lite	1.85	*23.83*	62.53
	MCD	**1.8**	***23.61***	62.52
	LS	2.08	*27.92*	61.92
	Calib C	2.04	31.09	61.22
	Calib R	2.07	*28.53*	60.68
	Calib T	4.21	38.25	**60.25**
MRPC	**MaxProb**	6.13	40.46	**63.88**
	MCD lite	5.48	*38.23*	65.76
	MCD(	5.35	***38.21***	65.62
	LS	6.08	39.05	64.99
	Calib C	6.17	39.82	64.99
	Calib R	6.52	39.99	65.13
	Calib T	13.35	39.75	64.22

Table 36. Comparing selective prediction performance (AUC of risk-coverage curve) of various approaches for Duplicate Detection datasets. Lower AUC is better in SP. MaxProb baseline scores are underlined, best performance is in **bold**, and scores that considerably outperform MaxProb are in *italics*.

Calib C and R approaches leverage this signal by training a calibrator over a held-out dataset and thus achieve superior SP performance.

5.4.2 Negligible Improvement / Degradation in OOD and ADV

In Figure 20, we compare the selective prediction performance of various approaches in OOD and ADV settings. To highlight the general trend, the results have been averaged over all the task-specific OOD/ADV datasets mentioned in Section 5.3.

In OOD setting, we find that the approaches lead to a negligible improvement in

Train On	Method	IID↓	OOD avg.↓	ADV avg.↓
SQuAD	**MaxProb**	6.71	46.73	33.69
	MCD lite	6.06	44.56	33.34
	MCD	**6.00**	***44.35***	33.05
	Calib C	6.15	45.93	33.27
	Calib R	6.25	45.94	33.18
	Calib T	14.72	60.31	47.87

Table 37. Comparing selective prediction performance (AUC of risk-coverage curve) of various approaches for QA datasets. Lower AUC is better in SP. MaxProb baseline scores are underlined, best performance is in bold, and scores that considerably outperform MaxProb are *italics*.

AUC. Notable improvement is achieved only by MCD in the case of the QQP dataset. In the ADV setting, all approaches degrade SP performance. *Surprisingly, MCD that performed relatively well in IID and OOD settings, degrades more (by* 1.74 *AUC) in comparison to other approaches* (except Calib T which does not perform well in all three settings). This is because the individual models of the ensemble achieve poor prediction accuracy in the ADV setting and thus ensembling them further degrades the overall confidence estimate.

5.4.3 Calib T Degrades Performance

Calib C and Calib R slightly outperform MaxProb in most IID and OOD cases. However, Calib T considerably degrades the performance in nearly all the cases. *We hypothesize that associating correctness directly with the input text embeddings could be a harder challenge for the model as embeddings of correct and incorrect instances usually do not differ significantly.* In contrast, as discussed before, providing

Train On	Method	IID↓	OOD avg.↓	ADV avg.↓
SNLI	**MaxProb**	2.78	23.34	32.4
	MCD(K=10)	2.52	23.96	32.61
	MCD(K=30)	**2.47**	23.81	32.47
	LS	2.7	***22.42***	**31.7**
	Calib C	2.57	22.47	33.0
	Calib R	2.61	23.12	33.95
	Calib T	7.02	34.74	40.68
MNLI	**MaxProb**	5.47	16.48	**28.39**
	MCD(K=10)	5.07	16.29	29.42
	MCD(K=30)	**4.92**	16.18	29.18
	LS	5.18	16.94	28.55
	Calib C	5.16	***14.16***	29.57
	Calib R	5.28	14.84	29.67
	Calib T	13.51	26.12	35.79
DNLI	**MaxProb**	7.36	53.59	**51.85**
	MCD(K=10)	7.17	53.77	53.23
	MCD(K=30)	6.69	53.67	53.24
	LS	*5.13*	53.04	53.67
	Calib C	***3.88***	**52.35**	52.91
	Calib R	*3.9*	53.08	52.83
	Calib T	5.46	53.58	58.13

Table 38. Comparing selective prediction performance (AUC of risk-coverage curve) of various approaches for NLI datasets. Lower AUC is better in SP. MaxProb baseline scores are underlined, best performance is in **bold**, and scores that considerably outperform MaxProb are in *italics*.

features such as predicted label and softmax probabilities explicitly assists Calib C and R approaches in finding some distinguishing patterns that improve the selective prediction performance.

5.4.4 Existing Approaches Fail to Utilize Additional Resources

Unlike typical ensembling, MCD does not require training or storing multiple models but, it requires making multiple inferences (using different dropout masks) and can still become practically infeasible for large models such as BERT as their inference cost is high. Calibration-based approaches need additional held-out data and careful feature engineering to train the calibrator. *Despite being computationally expensive, these approaches fail to consistently outperform MaxProb that does not require any such additional resources.*

5.4.5 Effect of Increasing Dropout Masks in Monte-Carlo Dropout

With the increase in number of dropout masks used in MCD, the SP performance improves (from *MCD lite* with 10 masks to *MCD* with 30 masks). *This is due to the ensembling effect as combining more predictions on the same input results in a more accurate overall output.* However, we note that both MCD lite and MCD degrade SP performance in the ADV setting as discussed in 5.4.2.

5.4.6 No Clear Winner

None of the approaches consistently and considerably outperforms MaxProb in all three settings. Most approaches do not fare well in OOD and ADV settings. Furthermore, a single approach does not achieve the highest performance across all tasks.

For instance, MCD outperforms all other approaches on Duplicate Detection datasets but does not perform well on NLI datasets (Calib C achieves better performance, especially in the OOD setting). This reveals that *the existing selective prediction approaches do not translate well across tasks.*

Table 36 compares SP performance (AUC of risk-coverage curve) of various approaches for Duplicate Detection datasets. Table 37 compares SP performance (AUC of risk-coverage curve) of various approaches for QA datasets. Table 38 compares SP performance (AUC of risk-coverage curve) of various approaches for NLI datasets.

5.4.7 MaxProb for Selective Prediction

Figure 21 shows the trend of accuracy against maxProb for various models in the IID setting. It can be observed that with the increase in MaxProb the accuracy usually increases. This implies that a higher value of MaxProb corresponds to more likelihood of the model's prediction being correct. Hence, MaxProb can be directly used as the confidence estimator for selective prediction. We plot the risk-coverage curves using MaxProb as the SP technique in Figure 22. As expected, the risk increases with the increase in coverage for all the models. We plot such curves for all techniques and compute area under them to compare their SP performance. This shows that MaxProb is a simple yet strong baseline for selective prediction.

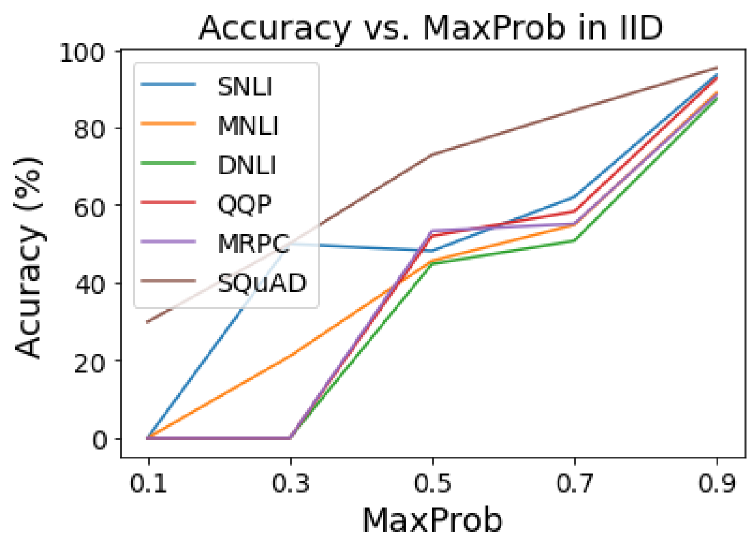


Figure 21. With increase in MaxProb, the accuracy usually increases.

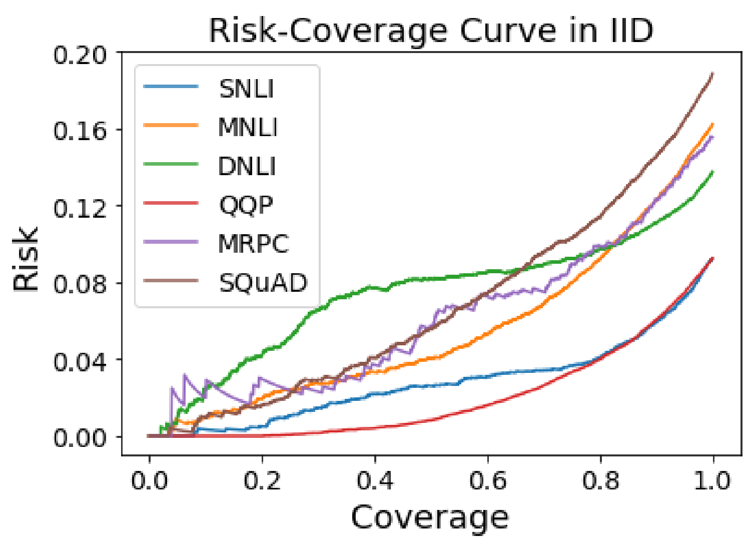


Figure 22. With increase in coverage (i.e decrease in abstention threshold), the risk usually increases.

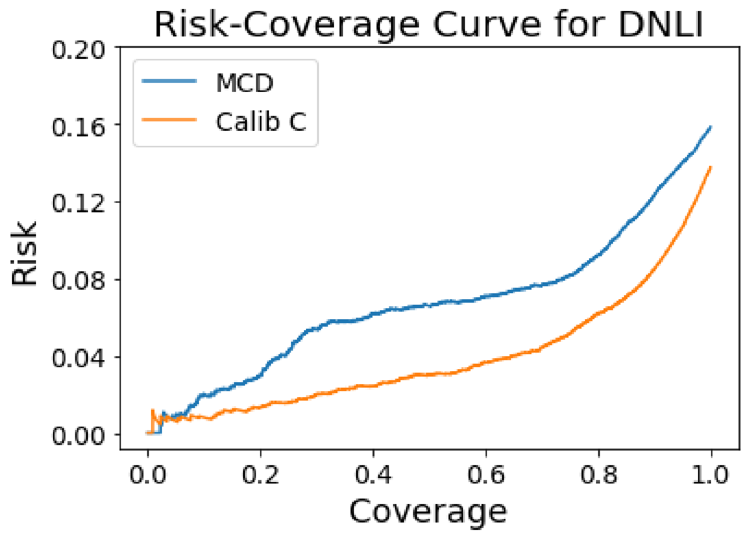


Figure 23. Comparing risk-coverage curves of MCD and Calib C for DNLI dataset in IID setting.

5.4.8 Comparing Risk-Coverage Curves of MCD and Calib C for DNLI Dataset in IID Setting

We compare the risk-coverage curves of MCD and Calib C approaches on DNLI in Figure 23. We observe that at all coverage points, Calib C achieves lower risk than MCD and hence is a better SP technique. We find that they benefit from using the predicted label as a feature for calibration. Specifically, the model's prediction accuracy varies greatly across labels (0.94, 0.91, and 0.76 for entailment, contradiction, and neutral labels respectively). This implies that when the model's prediction is neutral, it is relatively less likely to be correct (at least in the IID setting). Calib C and R approaches leverage this signal and tune the confidence estimator using a held-out dataset and thus achieve superior SP performance.

5.5 Towards Improving Selective Prediction Ability of NLP Systems

Hendrycks and Gimpel 2017 proposed '*MaxProb*' that uses the maximum softmax probability across all answer candidates as the confidence estimate to selectively make predictions. While performing reasonably well in the *in-domain* setting, *MaxProb* and other existing selective prediction techniques fail to translate that performance in the *out-of-domain* setting (Varshney, Mishra, and Baral 2022b; Kamath, Jia, and Liang 2020).

In this work, we propose a selective prediction method that improves probability estimates of models in both in-domain and out-of-domain settings by learning strong representations via calibration. Specifically, we calibrate models' outputs using a held-out dataset and use the calibrator as confidence estimator for selective prediction. To this end, we first argue that "*all instances are not equally difficult and the model is not equally confident in all its predictions*" and then through extensive experiments, we show that prediction confidence is positively correlated with correctness while difficulty score is negatively correlated (5.8.2). We leverage the above finding to calibrate models' outputs using these two signals.

For computing the difficulty scores, we use a *model-based* technique (5.6.1) because human perception of difficulty may not always correlate well with machine interpretation. To calibrate a model, we annotate instances of a held-out dataset conditioned on the model's predictive correctness (computed using difficulty score and prediction confidence) and then train a calibrator using these instances. This annotation score represents the likelihood of correctness of the model's prediction.

Finally, the trained calibrator predicts this likelihood value for test instances and is used as the confidence estimator for selective prediction.

To evaluate the efficacy of our method, we conduct comprehensive experiments in In-Domain (IID) and Out-of-Domain (OOD) settings for Natural Language Inference (NLI) and Duplicate Detection (DD) tasks. We also compare its performance with existing calibration techniques. On the NLI task, our method achieves 15.81% and 5.64% improvement on AUC of *risk-coverage* curve over *MaxProb* in IID and OOD setting respectively. Furthermore, on the DD task, it achieves 6.19% and 13.9% improvement in IID and OOD setting respectively. Finally, we hope that our work will facilitate development of more robust and reliable AI systems making their wide adoption in real-world applications possible.

5.6 Method

We propose to train a confidence estimator that can assign higher scores to correctly predicted instances than incorrectly predicted ones. To this end, we leverage a held-out dataset and annotate it's instances conditioned on the model's predictive correctness. Specifically, we infer the model on the held-out dataset and annotate instances with a score such that correctly predicted instances get assigned a higher score than incorrectly predicted instances. This annotation score models the likelihood of the prediction being correct and is computed using the model's prediction confidence and difficulty level of the instance. Finally, a calibrator (regression model) is trained using this annotated held-out dataset and used as the confidence estimator for selective prediction.

We detail each component of our method and the intuition behind it in the following subsections.

5.6.1 Difficulty Score Computation

To compute difficulty score of an instance, we evaluate it after every training epoch and subtract the aggregated softmax probability assigned to the ground-truth answer from 1 i.e. for an instance i, difficulty score d_i is calculated as:

$$s_i = \frac{\sum_{j=1}^{E} c_{ji}}{E}$$

$$d_i = 1 - s_i$$

where the model is trained till E epochs and c_{ji} is prediction confidence of the correct answer given by the model after j^{th} training epoch. Note that c_{ji} is probability assigned to the correct answer not the maximum probability across all answer candidates. The intuition behind this procedure is that the *instances that can be consistently answered correctly from the early stages of training are inherently easy and should receive lower difficulty score than the ones that require a large number of training steps.* A similar method has been explored in (Swayamdipta et al. 2020) for analyzing "training dynamics" but here we use it to quantify difficulty of the held-out instances.

5.6.2 Annotation Score Computation

We define annotation score for the held-out instances as a function of *softmax probability* outputted by the model and the *difficulty score*. We show that softmax score is positively correlated while difficulty score is negatively correlated with the predictive correctness i.e the system is more likely to be correct if the softmax score is high and difficulty score is low. Furthermore, in order to justifiably separate the scores for correct and incorrect prediction scenarios in the range 0 to 1, we push the scores above 0.5 in case of correct and below 0.5 in case of incorrect scenarios. Concretely, we use the following functions to compute this:

$$AS_1 = \begin{cases} 0.5 + \frac{maxProb}{2}, & \text{if correct} \\ 0.5 - \frac{maxProb}{2}, & \text{otherwise} \end{cases}$$

$$AS_2 = \begin{cases} 0.5 + \frac{s_i}{2}, & \text{if correct} \\ 0.5 - \frac{s_i}{2}, & \text{otherwise} \end{cases}$$

$$AS_3 = \begin{cases} 0.5 + \frac{max(s_i, maxProb)}{2}, & \text{if correct} \\ 0.5 - \frac{min(s_i, maxProb)}{2}, & \text{otherwise} \end{cases}$$

AS_1 uses only softmax, AS_2 uses only difficulty score and AS_3 uses a combination of both. These annotation strategies assign a relatively higher score when the model's prediction is correct and a lower score when it is incorrect. This gold score ranges from 0 to 1 as both s_i and $maxProb$ lie in the same range and better captures the likelihood of correctness unlike the categorical labels (1 for correct and 0 for incorrect) used in typical calibration approaches. **Note that this annotation computation is only required for training the calibrator and not at test time.** Therefore, difficulty score of the test instances need not be computed.

Both difficulty score and annotation score computation procedures are generic and are widely applicable since NLP systems usually make probabilistic predictions for all kinds of tasks ranging from Classification to Question Answering.

5.6.3 Calibration

Equipped with annotation scores, we extract syntactic features, namely, lengths, Semantic Textual Similarity (STS) value, number of common words between given sentences, and presence of negation words / numbers from the held-out instances to train the calibrator model. These features along with maxProb and prediction outputted by the model serve as inputs for the calibrator. Finally, we use a simple random forest implementation of Scikit-learn (Pedregosa et al. 2011) to train our calibrator that learns strong representations for the inputs. We note that these syntactic features are general and applicable for all language understanding tasks and any regression model can be used as the calibrator. We compare our method with other calibration techniques described in Section 5.7.1.

5.7 Experimental Setup

5.7.1 Calibration Baselines

Kamath, Jia, and Liang 2020 study a calibration-based selective prediction technique for Question Answering datasets where they annotate a held-out dataset such that correctly predicted instances are assigned class label '1' and incorrect ones are assigned label '0'. Then, a calibrator is trained using this annotated binary classification dataset using features such as input length and probabilities of top 5 predictions. The softmax probability assigned to class '1' by this calibrator is used as the confidence estimator for selective prediction. We refer to this approach as **Calib C**. We also train a transformer-based model for calibration (**Calib T**) that leverages the entire input text for this classification task instead of the syntactic features (Garg and Moschitti 2021).

Our proposed calibration method differs from these approaches as we quantify the correctness on a continuous scale (instead of categorical labels '1' and '0') using prediction confidence and difficulty of the instances and use explicitly provided general syntactic features described in Section 5.6.3 for training. Our annotation procedure provides more flexibility for the calibrator to look for fine-grained features distinguishing various annotation scores. We note that our simplest annotation strategy (AS_1) that does not incorporate difficulty score is similar to Calib R method described in (Varshney, Mishra, and Baral 2022b) but our calibration method uses more general syntactic features.

Method	SNLI	MNLI			Stress Test			
		Matched	Mismatched	Avg	Competence	Distraction	Noise	Avg
MaxProb (AUC)	2.78	14.00	14.44	14.22	47.87	26.49	20.34	31.57
Calib T (%)	-181.2	-129.55	-127.86	-128.69	-48.65	-81.3	-91.17	-68.93
Calib C (%)	+8.97	+2.15	-1.36	+0.40	-3.75	**+8.27**	-0.80	+0.55
Proposed (%)	**+15.81**	**+2.35**	**+2.04**	**+2.19**	**+8.01**	+6.60	**+0.22**	**+5.64**

Table 39. Comparing percentage improvement of various calibration approaches on AUC of risk-coverage curve (over MaxProb) in in-domain (SNLI) and out-of-domain settings (MNLI, Stress Test) for NLI task.

Method	MRPC	QQP
MaxProb (AUC)	6.13	40.46
Calib T (%)	-148.87	+2.21
Calib C (%)	-0.82	+2.0
Proposed (%)	**+6.19**	**+13.9**

Table 40. Comparing % improvement of various calibration approaches on AUC of risk-coverage curve in IID (MRPC) and OOD (QQP) settings for DD task.

Note that for fair estimation of abilities of the proposed method, we compare it with other calibration-based techniques only. Other techniques such as Monte-Carlo dropout (Gal and Ghahramani 2016) and Error Regularization (Xin et al. 2021) are complementary and can further improve our performance.

5.7.2 Datasets

We conduct experiments with Natural Language Inference and Duplicate Detection datasets and compare the performance of various calibration techniques in in-domain and out-of-domain settings.

NLI Datasets: SNLI (Bowman et al. 2015), MNLI (Williams, Nangia, and Bowman 2018) (Matched and Mismatched), and Stress Test (Naik et al. 2018) (Competence, Distraction, and Noise).

Duplicate Detection Datasets: QQP (Iyer, Dandekar, and Csernai 2017) and MRPC (Dolan and Brockett 2005).

For NLI task, we train 3-way classification model (NLI has three labels) on SNLI and evaluate the selective prediction performance on SNLI (IID) and MNLI, Stress Test (OOD) datasets. For the DD task, we train model on MRPC and evaluate on MRPC (IID) and QQP (OOD) datasets. We use BERT-BASE model (Devlin et al. 2019) with a linear layer on top of [CLS] token representation for training the model for these tasks. We train these models with the default learning rate of $5e-5$ for 3 epochs. We use the same experimental setup as (Varshney, Mishra, and Baral 2022b) for calibration methods.

5.8 Results and Analysis

5.8.1 MaxProb Struggles in OOD Setting

First rows in Table 39 and 40 show the AUC values achieved by MaxProb in NLI and DD tasks respectively. Note that in selective prediction, low AUC values of risk-coverage curves are preferred. We find that MaxProb performs well in the IID setting as it achieves low AUC values (2.78 on SNLI and 6.13 on MRPC). However, it fails to translate that in the OOD setting (AUC of 14.22 on MNLI, 31.57 on Stress Test, and 40.46 on QQP). This implies that the model makes a significant number of incorrect predictions with relatively high MaxProb and thus needs to be calibrated.

For calibration methods, we compare the performance improvement achieved over MaxProb w.r.t the minimum possible AUC.

5.8.2 Proposed Method Outperforms All

Our method shows a clear benefit over existing calibration techniques as it leads to a considerable improvement in all the cases. The proposed method achieves 15.81% and 6.19% improvement in the IID setting on SNLI and MRPC respectively. Furthermore, it achieves 2.19% on MNLI, 5.64% on Stress Test, and 13.9% on QQP in the OOD setting. *Calib T considerably degrades performance in both IID and OOD settings.* However, *Calib C results in a minor improvement in the IID setting* (8.97% for SNLI) *but does not consistently improve in the OOD setting* (especially on MNLI Mismatched and Competence Stress Test). We attribute this to the limited signal that is given to the calibrator by annotating the held-out dataset with categorical labels '1' and '0'. Thus, it learns weak representations.

Comparing Annotation Functions: We find that *the improvement using our method comes from using AS_3 as the annotation score* which outperforms AS_1 and AS_2. This is expected as it leverages useful signals provided by both maxProb and difficulty score for annotation computation.

Relationship With Predictive Correctness: To further analyze our method, we plot the relationship of predictive correctness with prediction confidence and difficulty score in Figure 24. It shows that prediction confidence is positively correlated

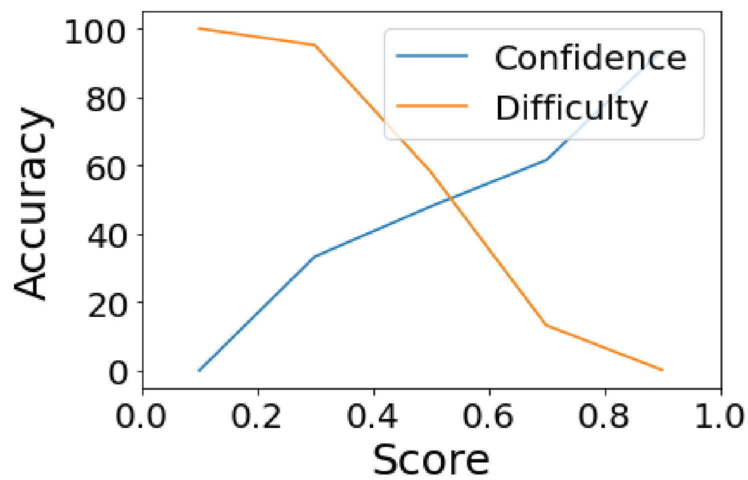


Figure 24. Trend of Model Accuracy with Confidence and Difficulty score for the NLI task.

while the difficulty score is negatively correlated with correctness. This further justifies our annotation score computation procedure.

5.9 Conclusion

We proposed a selective prediction method that calibrates the model outputs using prediction confidence and difficulty level of the instances. Through comprehensive experiments, we demonstrated that it achieves considerable improvement over MaxProb on NLI and Duplicate Detection tasks in both IID and OOD settings. We hope that our work will facilitate development of more robust and reliable AI systems making their wide adoption in real-world applications possible.

Chapter 6

RELIABLY INCREASING THE COVERAGE OF A SELECTIVE PREDICTION SYSTEM

Despite remarkable progress made in natural language processing, even the state-of-the-art models often make incorrect predictions. Such predictions hamper the reliability of systems and limit their widespread adoption in real-world applications. *Selective prediction* partly addresses the above concern by enabling models to abstain from answering when their predictions are likely to be incorrect. While selective prediction is advantageous, it leaves us with a pertinent question '*what to do after abstention*'. To this end, we present an explorative study on 'Post-Abstention', a task that allows re-attempting the abstained instances with the aim of increasing *coverage* of the system without significantly sacrificing its *accuracy*. We first provide mathematical formulation of this task and then explore several methods to solve it. Comprehensive experiments on 11 QA datasets show that these methods lead to considerable risk improvements –performance metric of the Post-Abstention task– both in the in-domain and the out-of-domain settings. We also conduct a thorough analysis of these results which further leads to several interesting findings.

6.1 Introduction

Despite remarkable progress made in Natural Language Processing (NLP), even the state-of-the-art systems often make incorrect predictions. This problem becomes worse when the inputs tend to diverge from the training data distribution (Elsahar and Gallé 2019; Miller et al. 2020; Koh et al. 2021). Incorrect predictions hamper the reliability of systems and limit their widespread adoption in real-world applications.

Selective prediction partly addresses the above concern by enabling models to abstain from answering when their predictions are likely to be incorrect. By avoiding potentially incorrect predictions, it allows maintaining high task accuracy and thus improves the system's reliability. Selective prediction has recently received considerable attention from the NLP community leading to development of several methods (Kamath, Jia, and Liang 2020; Garg and Moschitti 2021; Xin et al. 2021; Varshney, Mishra, and Baral 2022c). While these contributions are important, selective prediction leaves us with a pertinent question: *what to do after abstention?*

In this work, we address the above question and present an explorative study on '**Post-Abstention**', a task that allows re-attempting the abstained instances with the aim of increasing *coverage* of the given selective prediction system without significantly sacrificing its *accuracy*. Figure 25 illustrates the benefit of employing a post-abstention method; a model that achieves an accuracy of 70% is first enabled with the selective prediction ability that increases the accuracy to 85% but answers only 71% instances. Then, a post-abstention method is employed (for the 29% abstained instances) that assists the system in answering 9% more instances raising the coverage

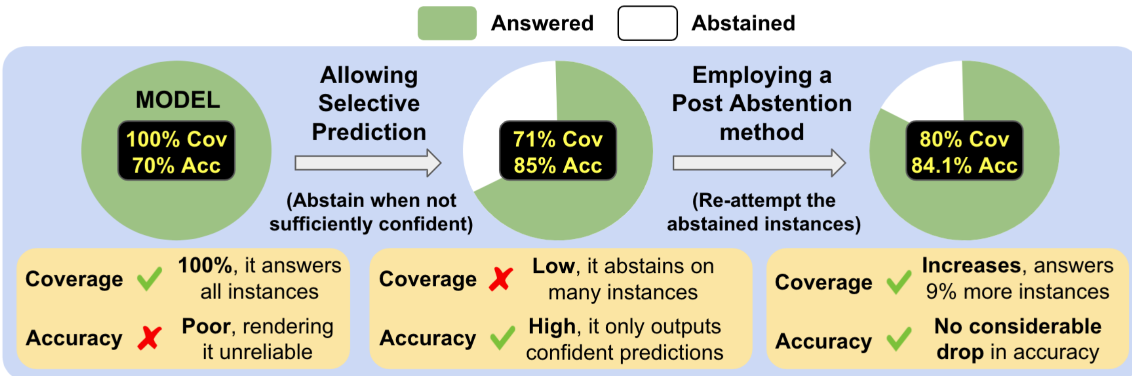


Figure 25. Illustrating the impact of employing a post-abstention method on top of selective prediction system. A regular model that has an accuracy of 70% (at coverage 100%) is first enabled with selective prediction ability that increases the accuracy to 85% but drops the coverage to 71%. Then, on employing a post-abstention method to the abstained instances (remaining 29%), coverage increases to 80% without a considerable drop in overall accuracy.

to 80% without considerably dropping the overall accuracy. We note that this task allows re-attempting all the abstained instances but does not require the system to necessarily output predictions for all of them i.e. the system can abstain even after utilizing a post-abstention method (when it is not sufficiently confident even in its new prediction). This facet not only allows the system to maintain its performance but also provides opportunities of sequentially applying stronger post-abstention methods to reliably and optimally increase the coverage in stages.

We provide mathematical formulation of the post-abstention task and explore several baseline methods to solve it (Section 6.2). To evaluate the efficacy of these methods, we conduct comprehensive experiments with 11 Question-Answering datasets from MRQA shared task (Fisch et al. 2019) in both in-domain and out-of-domain settings (Section 6.3). Our post-abstention methods lead to overall risk improvements (performance metric of the proposed task) of up to 21.81 in the in-domain setting and 24.23 in the out-of-domain setting. To further analyze these results, we study several research questions, such as 'what is the extent of overlap between the instances answered by different post-abstention methods', 'what is the distribution of model's original confidence on instances that get answered in the post-abstention stage', and 'how often do the system's predictions change after applying post-abstention methods'. In Section 6.4, we show that these investigations lead to numerous important and interesting findings.

In summary, our contributions are as follows:

1. We present an **explorative study on 'Post-Abstention'**, a task that aims at increasing the *coverage* of a given selective prediction system without significantly sacrificing its *accuracy*.
2. We **explore several baseline post-abstention methods** and evaluate them in an extensive experimental setup spanning 11 QA datasets in both in-domain and out-of-domain settings.
3. We show that the proposed post-abstention methods **result in overall risk value improvements** of up to 21.81 and 24.23 in the in-domain and out-of-domain settings respectively.
4. Our **thorough analysis** leads to several interesting findings, such as (a) instances answered by different post-abstention methods are not mutually exclusive i.e. there exist some overlapping instances, (b) instances that get answered in the post-abstention stage are not necessarily the ones on which the given system was initially most confident, etc.

We believe our work will encourage further research in Post-Abstention, an important step towards improving the reliability of NLP systems.

6.2 Post-Abstention

In this section, we first provide background for post-abstention (6.2.1) and then describe the task (6.2.2) and its approaches (6.2.3).

6.2.1 Background

Post-abstention, as the name suggests, is applicable for a system that abstains from answering i.e. a selective prediction system. A system can typically abstain when its prediction is likely to be incorrect. This improves the reliability of the system. Such a system typically consists of two functions: a predictor (f) that gives the model's prediction on an input (x) and a selector (g) that determines if the system should output the prediction made by f:

$$(f,g)(x) = \begin{cases} f(x), & \text{if g(x)} = 1 \\ Abstain, & \text{if g(x)} = 0 \end{cases}$$

Typically, g comprises of a prediction confidence estimator $\tilde{g}$ and a threshold th that controls the level of abstention for the system:

$$g(x) = \mathbb{1}[\tilde{g}(x)) > th]$$

A selective prediction system makes trade-offs between *coverage* and *risk*. Coverage at a threshold th is defined as the fraction of total instances answered by the system (where $\tilde{g} > th$) and risk is the error on the answered instances.

With decrease in threshold, coverage will increase, but the risk will usually also increase. The overall selective prediction performance is measured by the *area under Risk-Coverage curve* (El-Yaniv et al. 2010) which plots risk against coverage for all confidence thresholds. Lower AUC is better as it represents lower average risk across all confidence thresholds.

In NLP, approaches such as Monte-Carlo Dropout (Gal and Ghahramani 2016), Calibration (Kamath, Jia, and Liang 2020; Varshney, Mishra, and Baral 2022b, 2022c; Zhang, Gong, and Choi 2021), Error Regularization (Xin et al. 2021) and Label Smoothing (Szegedy et al. 2016) have been studied for selective prediction. In this work, we consider MaxProb (Hendrycks and Gimpel 2017), a technique that uses the maximum softmax probability across all answer candidates as the confidence estimator. We use this simple technique because the focus of this work is on post-abstention i.e. the next step of selective prediction. However, we note that the task formulation and the proposed methods are general and applicable to all selective prediction approaches.

6.2.2 Task Formulation

We define the post-abstention task as follows:

Given a selective prediction system with an abstention threshold, the post-abstention task allows re-attempting the abstained instances with the aim of improving the coverage without considerably degrading the accuracy (or increasing the risk) of the given system. Next, we mathematically describe the task and its performance evaluation methodology.

Let the coverage and risk of the given selective prediction system at abstention threshold th be cov_{th} and $risk_{th}$ respectively. A post-abstention method re-attempts the originally abstained instances (where $\tilde{g} < th$) and outputs the new prediction for the ones where it is now sufficiently confident. This typically leads to an increase in the coverage of the system with some change in the risk value; let the new coverage and risk be cov'_{th} and $risk'_{th}$ respectively. From the risk-coverage curve of the given system, we calculate its risk at coverage cov'_{th} and compare it with $risk'_{th}$ to measure the efficacy of the post-abstention method (refer to Figure 26).

For a method to have a positive impact, its risk ($risk'_{th}$) should be lower than the risk of the given system at coverage cov'_{th}. We summarize this performance evaluation methodology in Figure 26. To get an overall performance estimate of a post-abstention method, we compile these differences in risk values for all confidence thresholds and calculate an aggregated value. The higher the overall improvement value, the more effective the method is. We note that this evaluation methodology is fair and accurate as it conducts pair-wise comparisons at **equal coverage** points. An alternative performance metric could be AUC but it computes the overall area ignoring the pair-wise comparisons which are crucial for our task because the coverage points of the original system would be different from those achieved by the post-abstention method.

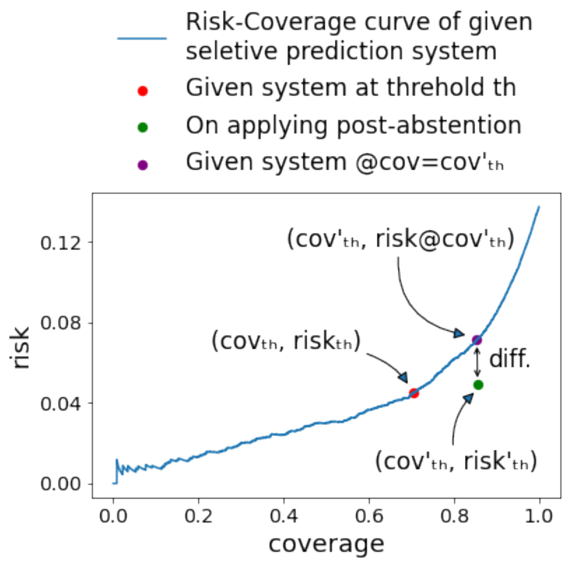


Figure 26. Summarizing performance evaluation methodology of post-abstention. Given a selective prediction system with coverage cov_{th} and risk $risk_{th}$ at abstention threshold th, let the new coverage and risk after applying a post-abstention method be cov'_{th} and $risk'_{th}$ respectively. From the risk-coverage curve of the given system, we calculate its risk at coverage cov'_{th} and compare it with $risk'_{th}$ (diff). *For the method to have a positive impact, $risk'_{th}$ should be lower than the risk of the given system at coverage cov'_{th}.*

6.2.3 Approaches

6.2.3.1 Ensembling using Question Paraphrases

It is well known that even state-of-the-art NLP models are often brittle i.e. when small semantic-preserving changes are made to the input, their predictions tend to fluctuate greatly (Jia and Liang 2017; Belinkov and Bisk 2018; Iyyer et al. 2018; Ribeiro, Singh, and Guestrin 2018; Wallace et al. 2019). Ensembling the predictions of the model on multiple semantically equivalent variants of the input is a promising

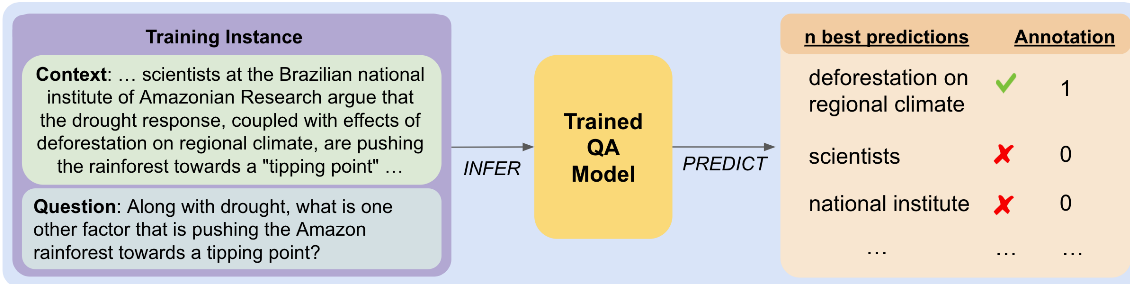


Figure 27. Illustrating annotation procedure of REToP. For each training instance, top N predictions given by the QA model are annotated conditioned on their correctness i.e. correct predictions are annotated as '1' and incorrect predictions are annotated as '0'. This annotated binary classification dataset is used to train the auxiliary model.

approach to address this issue (Anantha et al. 2021; Vakulenko et al. 2021) as it can reduce the spread or dispersion of the predictions.

We leverage the above technique in re-attempting the abstained questions i.e. we first generate multiple paraphrases of the input instance and then aggregate the model's predictions on them. We use BART-large (Lewis et al. 2019) model fine-tuned on Quora Question Corpus (Iyer, Dandekar, and Csernai 2017), PAWS (Zhang, Baldridge, and He 2019), and Microsoft Research Paraphrase Corpus (Dolan and Brockett 2005) for paraphrasing and explore the following strategies for aggregating the model predictions:

- **Mean**: In this strategy, we calculate the average confidence assigned to each answer candidate across all predictions. Then, we select the candidate with the highest average confidence as the system's prediction. Note that the system will output this prediction only if its confidence surpasses the abstention threshold.
- **Max**: Here, like the *mean* strategy, we select the answer candidate with the highest average confidence but we use the maximum confidence assigned to that candidate

as its prediction confidence. This is done to push the most confident prediction above the abstention threshold.

6.2.3.2 Re-Examining Top N Predictions (REToP)

State-of-the-art models have achieved impressive performance on numerous NLP tasks. Even in cases where they fail to make a correct prediction, they are often able to rank the correct answer as one of their top N predictions. This provides opportunities for re-examining the top N predictions to identify the correct answer in case of abstention. To this end, a model that can estimate the correctness of a prediction can be leveraged. Following this intuition, we develop an **auxiliary model** that takes the context, question, and a prediction as input and assigns a score indicating the likelihood of that prediction to be correct. This model can be used for each of the top N predictions given by the QA model to select the one that is most likely to be the correct answer.

Training Auxiliary Model: We first create data instances by annotating (context, question, prediction) triplets conditioned on the correctness of the QA system's predictions and then train a classification model using this data. This model is specific to the given QA system and essentially learns to distinguish its correct and incorrect predictions.

- **Annotate (context, question, prediction) triplets**: We utilize the trained QA model to get its top N predictions for each training instance. Then, we annotate each (context, question, prediction) triplet based on the prediction's correctness i.e.

a correct prediction is annotated as '1' and an incorrect prediction is annotated as '0'. Figure 27 illustrates this annotation step.

- **Train a classification model**: Then, a binary classification model is trained using the annotated dataset collected in the previous step. This model specifically learns to distinguish the correct predictions of the QA model from the incorrect ones. Softmax probability assigned to the label '1' corresponds to the likelihood of correctness for each prediction.

Note that we use the QA model's top N predictions to collect the '0' annotations instead of randomly selecting candidates because this procedure results in highly informative negative instances (that are probable predictions and yet incorrect) and not easy/obvious negatives. This can help the auxiliary model in learning fine-grained representations distinguishing correct and incorrect predictions.

Leveraging Auxiliary Model: For an abstained instance, we compute the likelihood value for each of the top N predictions given by the QA model using our trained auxiliary model. Then, we calculate the overall confidence (c) of each prediction (p) as a weighted average of the QA model's probability (s_q) and the auxiliary model's likelihood score (s_a) i.e. c_p is calculated as:

$$c_p = \alpha * s_q^p + (1 - \alpha) * s_a^p$$

where α is a weight parameter.

We incorporate QA model's probability as it provides more flexibility to compute the overall confidence. Finally, prediction with the highest overall confidence is selected as the new prediction. We differentiate this method from existing methods such as calibration in Section 6.7.

6.2.3.3 Human Intervention (HI)

In intolerant application domains such as biomedicals where incorrect predictions can have serious consequences, human intervention is the most reliable technique to answer the abstained instances. Human intervention can be in various forms such as providing relevant knowledge to the model, asking clarifying questions (Rao and Daumé III 2018) or simplifying the input question. In this work, we explore a simple human intervention approach in which the system provides multiple predictions instead of only one prediction for the abstained instances. The human can then select the most suitable prediction from the provided predictions. Performance of this method can be approximated based on the presence of the correct answer in the predictions provided to the human. Note that the above approach would answer all the abstained instances and hence the coverage would always be 100%. This implies that with the increase in abstention threshold, the risk would monotonically decrease as multiple predictions would be returned for a larger number of instances.

In addition to the above approach, we also explore a **REToP-centric** HI approach in which the system returns multiple predictions only when REToP surpasses the confidence threshold in the post-abstention stage. Similar to REToP, it abstains on the remaining instances. Finally, we note that comparing the performance of HI approaches with other post-abstention approaches would be unfair as other approaches return only a single prediction. Therefore, we present HI results separately.

Dataset	Model	0.2		0.32		0.36		0.48		0.54		0.60		0.68		Total Risk Improvement↑
		Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	
	Given (G)	96.65	32.45	87.24	28.10	83.34	26.69	69.94	21.91	62.57	19.91	56.23	17.98	47.92	15.43	
SQuAD	REToP	99.73	**33.75**	97.27	**31.93**	95.08	**30.85**	80.88	**24.84**	72.44	**21.82**	63.73	**19.19**	52.65	**16.43**	
(in-domain)	G@REToP$_{cov}$	-	34.00	-	32.77	-	31.67	-	25.82	-	22.59	-	20.24	-	16.83	**21.81**
	Given (G)	97.54	67.65	89.56	65.88	85.39	65.13	71.75	62.71	64.77	61.56	58.19	60.34	49.25	58.29	
HotpotQA	REToP	99.93	**68.17**	98.63	**67.39**	96.9	**66.61**	82.88	**63.61**	73.55	**61.89**	64.36	**60.53**	52.96	**58.34**	
	G@REToP$_{cov}$	-	68.30	-	67.92	-	67.47	-	64.52	-	63.04	-	61.55	-	59.01	**21.54**
	Given (G)	97.59	44.49	89.01	40.51	85.41	39.04	74.08	34.16	66.86	30.54	60.58	27.94	54.10	24.20	
RE	REToP	99.93	**45.38**	98.95	**44.39**	97.52	**43.79**	85.89	**38.67**	77.61	**34.57**	69.54	**31.12**	59.33	**25.39**	
	G@REToP$_{cov}$	-	45.47	-	45.01	-	44.43	-	39.22	-	35.51	-	32.10	-	27.33	**20.42**
	Given (G)	89.02	80.5	71.07	77.04	66.17	75.56	51.34	72.54	43.47	69.62	36.2	68.85	29.97	63.86	
RACE	REToP	99.41	82.24	92.28	**80.71**	86.94	**79.35**	62.91	**73.82**	51.48	**71.76**	42.28	**69.47**	33.09	**65.92**	
	G@REToP$_{cov}$	-	81.94	-	81.00	-	80.00	-	75.00	-	72.54	-	69.72	-	66.37	**15.10**
	Given (G)	93.90	69.76	80.91	66.40	75.5	64.91	60.30	60.79	53.30	58.8	47.17	56.62	39.32	54.11	
NewsQA	REToP	99.48	**71.03**	96.13	**70.24**	93.21	69.64	70.85	**63.71**	60.73	**60.67**	52.04	**58.07**	42.09	**54.94**	
	G@REToP$_{cov}$	-	71.31	-	70.36	-	69.61	-	63.81	-	61.01	-	58.33	-	55.02	**5.10**
	Given (G)	96.15	86.68	81.77	85.67	75.77	85.34	58.64	84.08	50.22	83.58	42.67	83.33	34.46	82.55	
SearchQA	REToP	99.92	87.06	97.58	86.81	93.92	**86.48**	71.49	**84.76**	59.46	**84.04**	48.6	**83.48**	37.08	**82.75**	
	G@REToP$_{cov}$	-	87.04	-	86.79	-	86.52	-	85.07	-	84.15	-	83.56	-	82.77	**1.78**
	Given (G)	96.67	67.31	86.89	65.05	82.54	63.82	68.81	60.39	61.44	58.39	55.11	56.48	47.12	54.03	
TriviaQA	REToP	99.86	**68.07**	97.07	**67.33**	93.72	**66.23**	76.72	62.40	67.93	60.25	59.55	**57.77**	49.29	54.89	
	G@REToP$_{cov}$	-	68.09	-	67.42	-	66.60	-	62.32	-	60.12	-	57.95	-	54.83	**0.70**
	Given (G)	92.37	63.78	79.04	59.99	74.87	58.77	60.60	53.51	54.03	51.00	47.94	48.31	41.70	45.27	
NQ	REToP	98.71	**65.34**	93.04	**63.39**	89.30	**62.62**	70.65	**56.90**	61.68	**53.54**	53.24	**50.10**	43.75	**46.44**	
	G@REToP$_{cov}$	-	65.67	-	63.93	-	63.02	-	57.43	-	53.80	-	50.68	-	46.45	**10.70**
	Given (G)	95.74	88.46	81.17	87.38	76.11	87.33	62.34	86.23	53.69	85.38	48.77	84.45	43.05	85.01	
DROP	REToP	99.53	88.64	92.95	**87.83**	88.42	88.04	69.00	**86.31**	58.55	**85.57**	51.90	**84.49**	44.18	85.09	
	G@REToP$_{cov}$	-	88.63	-	88.19	-	87.88	-	86.69	-	85.91	-	84.87	-	84.94	**3.63**
	Given (G)	97.20	68.68	87.87	66.41	84.21	65.82	71.09	62.42	64.16	61.47	57.16	59.91	50.03	58.46	
DuoRC	REToP	99.87	**69.45**	98.33	**69.17**	96.14	68.68	80.75	**64.69**	71.95	**62.59**	62.56	**60.70**	52.90	**58.69**	
	Original@cov	-	69.51	-	69.02	-	68.4	-	64.77	-	62.74	-	60.92	-	59.32	**4.32**
	Given (G)	94.34	67.14	80.9	63.32	75.65	61.92	57.49	56.02	49.63	52.14	41.45	51.04	34.07	50.00	
TBQA	REToP	99.53	**68.38**	95.01	**67.23**	91.68	**66.18**	68.20	**58.34**	58.55	**54.77**	47.37	**51.26**	37.26	**49.64**	
	G@REToP$_{cov}$	-	68.56	-	67.30	-	66.23	-	59.41	-	56.02	-	52.60	-	50.71	**24.23**

Table 41. Performance of REToP as a post-abstention method for selected abstention thresholds. The QA model is trained using SQuAD training data and evaluated on SQuAD (in-domain) and 10 out-of-domain datasets. The last column corresponds to the overall improvement aggregated over all confidences ranging from 0 to 1 at an interval of 0.02. ↓ and ↑ indicate that lower (risk) and higher (coverage, risk improvement) values are better respectively.

6.3 Experiments and Results

6.3.1 Experimental Setup

Datasets: We experiment with SQuAD 1.1 (Rajpurkar et al. 2016) as the source dataset and the following 10 datasets as out-of-domain datasets: NewsQA (Trischler et al. 2017), TriviaQA (Joshi et al. 2017), SearchQA (Dunn et al. 2017), HotpotQA (Z. Yang et al. 2018), and Natural Questions (Kwiatkowski, Palomaki, Redfield, Collins, Parikh, Alberti, Epstein, Polosukhin, Devlin, Lee, et al. 2019b), DROP (Dua et al. 2019), DuoRC (Saha et al. 2018), RACE (Lai et al. 2017), RelationExtraction (Levy et al. 2017), and TextbookQA (Kim, Kim, and Kwak 2019). We use the preprocessed data from the MRQA shared task (Fisch et al. 2019) for our experiments.

Implementation Details: We run all our experiments using the huggingface (Wolf et al. 2020) implementation of transformers on Nvidia V100 16GB GPUs with a batch size of 32 and learning rate ranging in $\{1–5\}e–5$. We generate 10 paraphrases of the question in Ensembling method, re-examine top 10 predictions, vary α in the range 0.3 – 0.7 for REToP method, and vary the number of predictions in the range 2 to 5 for HI methods. Since the focus of this work is on post-abstention, it's crucial to experiment with models that leave sufficient room for effectively evaluating the ability of post-abstention methods. For that reason, we experiment with a small size model (BERT-mini having just 11.3M parameters) from (Turc et al. 2019) for our experiments. However, we note that our methods are general and applicable for all models.

6.3.2 Results

6.3.2.1 REToP

Table 41 shows the post-abstention performance of REToP for selected abstention thresholds. The last column ('*Total Risk Improvement*') in this table corresponds to the overall improvement aggregated over all confidence thresholds. It can be observed that REToP achieves considerable risk improvements both in the in-domain setting (21.81 on SQuAD) and the out-of-domain settings (24.23 on TextbookQA, 21.54 on HotpotQA, 20.42 on RE, etc). Next, we analyze these results in detail.

Higher improvement on moderate confidences: In Figure 28, we plot risk improvements achieved by REToP on SQuAD (in-domain) and HotpotQA (out-of-domain) datasets for all confidence thresholds. These plots reveal that the improvement is more on moderate thresholds as compared to low thresholds. We attribute this to the high difficulty of instances that remain to be re-attempted at low thresholds i.e. only the instances on which the given system was highly underconfident are left for the post-abstention method. It has been shown that model's confidence is negatively correlated with difficulty (Swayamdipta et al. 2020; Rodriguez et al. 2021; Varshney, Mishra, and Baral 2022a; Mishra, Arunkumar, Bryan, et al. 2022) implying that the remaining instances are tough to be answered correctly. This justifies the lesser improvement in performance observed at low thresholds.

In-Domain vs Out-of-Domain Improvement: REToP achieves higher performance improvement on the in-domain dataset than the out-of-domain datasets (on average). This is expected as the auxiliary model in REToP is trained using the in-domain training data. However, it still has good performance on out-of-domain datasets as the auxiliary model learns fine-grained representations to distinguish

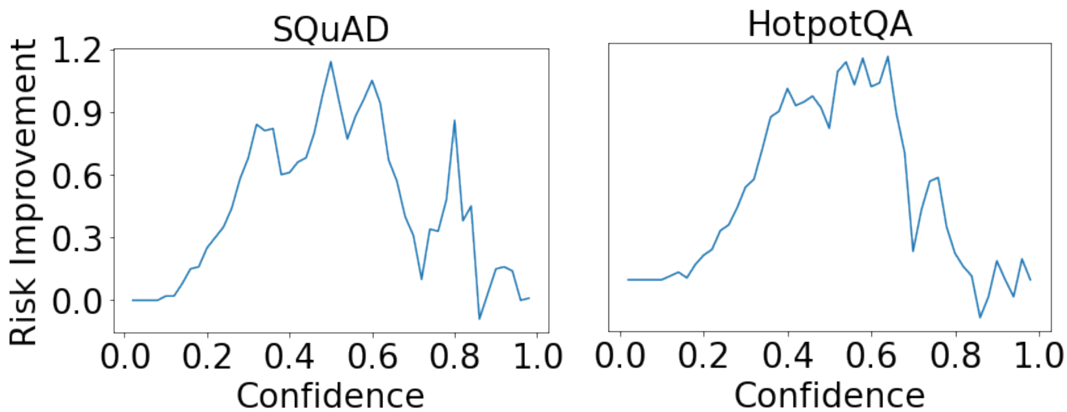


Figure 28. Improvement in risk achieved by using REToP in post-abstention on SQuAD (in-domain) and HotpotQA (out-of-domain) datasets for all confidences.

between correct and incorrect predictions. Furthermore, the improvement on out-of-domain data varies greatly across datasets (from 0.7 on TriviaQA to 24.23 on TextbookQA).

6.3.2.2 Comparing Post-Abstention Approaches

We provide the performance tables for other post-abstention approaches. However, we compare their total risk improvement values in Table 42. In the in-domain setting, REToP achieves higher improvement than Ensembling method. This is because the auxiliary model in REToP has specifically learned to distinguish the correct and incorrect predictions from the training data of this domain. However, in some out-of-domain cases, Ensembling outperforms REToP (SearchQA, TriviaQA, NewsQA). Overall, REToP leads to a consistent and higher risk improvement on average. Ensembling also leads to a minor degradation in a few out-of-domain datasets

Dataset	Ens.	RETоP (α = 0.6)	RETоP (α = 0.65)	*HI on (RETоP)
SQuAD	0.29	21.81	20.02	47.85
HotpotQA	0.93	21.54	19.00	37.88
RE	21.72	20.42	17.61	46.65
RACE	16.72	15.10	14.17	36.26
NewsQA	11.92	5.10	5.10	26.41
SearchQA	17.05	1.78	2.23	20.08
TriviaQA	9.50	0.70	1.47	17.21
NQ	13.40	10.70	10.89	31.95
DROP	1.57	3.63	2.99	8.08
DuoRC	-1.69	4.32	5.90	20.26
TBQA	-6.93	24.23	23.73	45.18
Total	84.48	**129.33**	123.11	337.81

Table 42. Comparing total risk improvement achieved by different post-abstention methods. * for HI indicates that it's results are not directly comparable as it outputs multiple predictions while others output only one.

(DuoRC and TextbookQA). Next, we analyze the performance of human intervention (HI) methods.

6.3.2.3 Human Intervention (HI)

We study two variants of HI method. In the first variant, multiple predictions (n=2) are returned for all the abstained instances. This makes the coverage to be 100% for all the confidences; therefore, we present only the risk values in Table 43. As expected, with increase in abstention threshold, the risk decreases because multiple predictions get outputted for a larger number of instances. Selection of operating threshold for an application depends on the trade-off between risk that can

Dataset	0.0	0.2	0.4	0.6	0.8
SQuAD	34.15	33.72	30.9	28.05	26.3
HotpotQA	68.33	68.19	66.56	63.65	61.57
RE	45.52	45.35	43.39	41.28	39.31
RACE	82.05	81.6	80.12	78.19	77.15
NewsQA	71.46	71.2	69.42	67.21	65.29
SearchQA	87.06	86.92	85.64	83.98	82.94
TriviaQA	68.13	67.9	66.62	64.21	62.47
NQ	66.09	65.67	63.63	61.06	59.31
DROP	88.69	88.69	87.56	86.36	85.7
DuoRC	69.55	69.42	68.15	66.42	65.22
TBQA	68.73	68.46	67.07	64.74	64.01

Table 43. Comparing risk values achieved by the HI method (returns two predictions for all abstained instances) across different abstention thresholds.

be tolerated and human effort required to select the most suitable prediction from a set of predictions returned by the system. For example, a low threshold can be selected for tolerant applications like movie recommendations and a high threshold for tolerant applications like house robots.

In the second variant of HI method, we study a **REToP-centric** approach in which the system returns multiple predictions only when REToP surpasses the confidence threshold in the post-abstention stage. The last column in Table 42 shows the risk improvements achieved by this approach (n=2). Note that REToP re-examines the top N predictions and selects one while this method outputs multiple predictions and requires a human to select the most suitable one. These results indicate that though REToP achieves good performance, there is still some room for improvement.

6.3.2.4 Ensembling Using Paraphrases

Comparing the performance of Mean and Max Ensembling strategies reveals that Max increases the coverage more than the Mean strategy but it also increases the risk considerably. Thus, pushing the instance's confidence to surpass the abstention threshold fails to provide risk improvements. However, such a technique could be employed in scenarios where risk degradation can be tolerated.

6.4 Analysis

What is the distribution of model's original confidence on the instances that get answered after applying post-abstention method? In Figure 29, we show the distribution of model's original confidence on SQuAD instances that get answered by RETоP at abstention threshold 0.5. Green-colored bars represent the number of instances answered from each confidence bucket. *We found that RETоP answers a large number of instances from the high confidence buckets; however, instances from even low confidence buckets get answered.* This can further be controlled using the weight parameter (α) in the overall confidence computation.

How often do the system's predictions change after applying RETоP and what is its impact? RETоP can either boost the confidence of the top most prediction of the given model or can select a different answer by re-examining its top N predictions. In Figure 30, we specifically analyze the latter scenario i.e. the instances on which RETоP's prediction differs from the original model's prediction. At a threshold of 0.5, the original system abstains on 3411 SQuAD instances and after applying RETоP, it answers 1110 of those instances. Out of these 1110 instances, the RETоP changes the prediction on 186 instances. The original prediction is incorrect

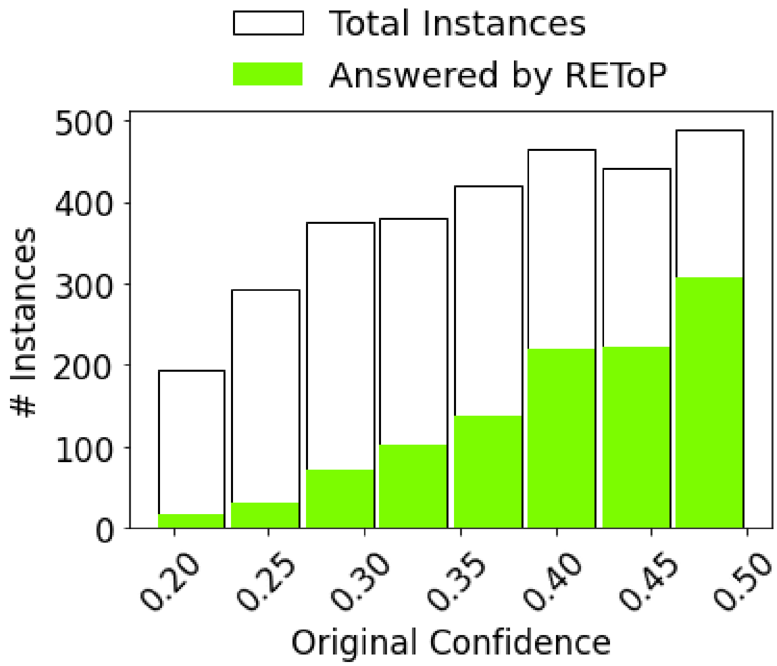


Figure 29. Distribution of QA model's confidence on SQuAD instances that get answered after applying REToP at abstention threshold 0.5.

in more cases (99 vs 87) and after applying REToP, the system gives 116 correct predictions and only 70 incorrect. This implies that by overriding the original system's prediction, REToP improves the system's accuracy. However, in some cases, it also changed a correct prediction to incorrect but such cases are lesser than the former.

To what extent do the instances answered by different post-abstention methods overlap? In Figure 31, we demonstrate the Venn diagram of SQuAD instances answered by REToP and Ensembling (Mean) approaches at abstention threshold 0.5. REToP answers 1110 instances while Ensembling answers 277 and there 127 common instances between the two approaches. This indicates that the two sets are not mutually exclusive i.e. there are some instances that get targeted by both

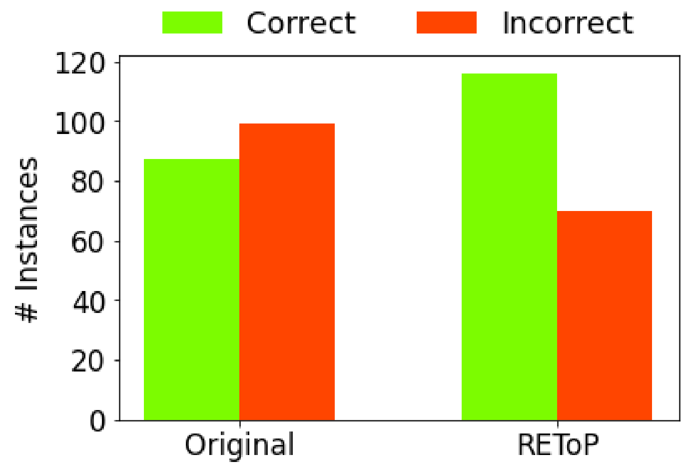


Figure 30. Number of correct (green) and incorrect (red) predictions on those abstained SQuAD instances where REToP surpasses the abstention threshold of 0.5 but its prediction differs from the original system.

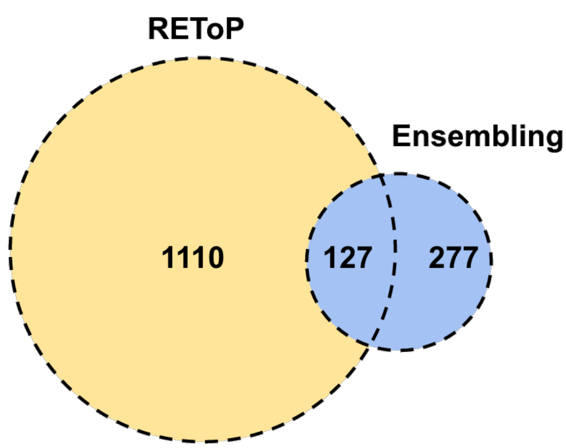


Figure 31. Venn diagram of abstained SQuAD instances answered by REToP and Ensembling (Mean) approaches at abstention threshold 0.5.

the approaches; however, there are a significant number of instances that are not in the intersection. This result motivates studying composite or sequential application of different post-abstention methods to further improve the post-abstention performance.

6.5 Ensembling (Mean) Performance

Table 45 shows the performance of using Ensembling (Mean) as a post-abstention method for a few selected abstention threshold values. For each dataset, we provide three rows: the first row ('*Given*') shows the coverage and risk values of the given selective prediction system at specified abstention thresholds, the second row ('*Ens*') shows the coverage and risk after applying the post-abstention method on the abstained instances of the given selective prediction system, and the final row ('$G@Ens_{cov}$') shows the risk of the given selective system at the coverage achieved by *Ens* method. For the post-abstention method to be effective the risk in the second row should be less than that in the third row and the magnitude of difference corresponds to the improvement. The last column '*Total Risk Improvement*' shows the overall improvement aggregated over all confidence thresholds ranging between 0 and 1 at an interval of 0.02.

6.6 Dataset Statistics

Table 44 shows the statistics of all evaluation datasets used in this work. SQuAD corresponds to the in-domain dataset while the remaining 10 datasets are out-of-domain. We use the pre-processed data from the MRQA shared task (Fisch et al. 2019).

Dataset	**Size**	**Dataset**	**Size**
SQuAD	10507	HotpotQA	5901
RE	2948	RACE	674
NewsQA	4212	SearchQA	16980
TriviaQA	7785	NQ	12836
DROP	1503	DuoRC	1501
TBQA	1503		

Table 44. Statistics of evaluation data used in this work.

6.7 Differentiating REToP from Calibration

REToP is different from calibration based techniques presented in (Kamath, Jia, and Liang 2020; Varshney, Mishra, and Baral 2022b) in the following aspects:
(a) Firstly, REToP does not require a held-out dataset unlike calibration based methods that infer the model on the held-out dataset to gather instances on which the model in incorrect.
(b) Secondly, the auxiliary model trained in REToP predicts the likelihood of correctness of (context, question, prediction) triplet i.e. it is used for each of the top N prediction individually. This is in contrast to calibrators that predicts a single score for an instance and ignores the top N predictions.
(c) Finally, we use the entire context, question, and the prediction to predict its correctness likelihood score unlike feature-based calibrator models in which a random-forest model is trained using just syntax-level features such as length of question, semantic similarity of prediction with the question, etc.

6.8 Other Post-Abstention Techniques

Asking clarifying questions to the user in order to get information about the question has started to received considerable research attention in conversational, web search, and information retrieval settings (Aliannejadi et al. 2021, 2020; Zamani,

Dataset	Model	0.2		0.32		0.36		0.48		0.54		0.60		0.68		Total Risk
		Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Cov↑	Risk↓	Improvement↑
	Given (G)	96.65	32.45	87.24	28.10	83.34	26.69	69.94	21.91	62.57	19.91	56.23	17.98	47.92	15.43	
SQuAD	Ens	97.64	32.88	89.51	28.93	87.64	28.24	72.46	22.71	65.12	20.58	58.37	18.7	49.59	15.89	
(in-domain)	G@Ens$_{cov}$	-	32.96	-	29.09	-	28.26	-	22.58	-	20.65	-	18.66	-	15.91	0.29
	Given (G)	97.54	67.65	89.56	65.88	85.39	65.13	71.75	62.71	64.77	61.56	58.19	60.34	49.25	58.29	
HotpotQA	Ens	98.59	67.84	91.93	66.23	90.41	65.92	75.65	63.17	68.45	62.22	61.31	60.72	52.26	58.88	
	G@Ens$_{cov}$	-	67.9	-	66.37	-	66.04	-	63.4	-	62.14	-	60.91	-	58.94	0.93
	Given (G)	97.59	44.49	89.01	40.51	85.41	39.04	74.08	34.16	66.86	30.54	60.58	27.94	54.10	24.20	
RE	Ens	98.27	44.56	92.2	41.35	90.57	40.71	77.44	34.87	70.86	31.45	64.86	29.08	56.07	24.74	
	G@Ens$_{cov}$	-	44.82	-	42.27	-	41.42	-	35.58	-	32.47	-	30.02	-	25.54	21.72
	Given (G)	89.02	80.5	71.07	77.04	66.17	75.56	51.34	72.54	43.47	69.62	36.2	68.85	29.97	63.86	
RACE	Ens	91.69	80.42	73.89	77.71	71.51	77.18	53.71	72.65	46.88	70.25	40.21	69.0	31.6	64.79	
	G@Ens$_{cov}$	-	80.88	-	77.31	-	77.13	-	72.93	-	71.43	-	70.11	-	65.09	16.72
	Given (G)	93.90	69.76	80.91	66.40	75.5	64.91	60.30	60.79	53.30	58.8	47.17	56.62	39.32	54.11	
NewsQA	Ens	95.56	70.24	83.52	67.14	81.13	66.49	63.01	61.53	55.75	59.45	49.53	57.19	41.17	54.21	
	G@Ens$_{cov}$	-	70.18	-	67.02	-	66.46	-	61.63	-	59.67	-	57.33	-	54.67	11.92
	Given (G)	96.15	86.68	81.77	85.67	75.77	85.34	58.64	84.08	50.22	83.58	42.67	83.33	34.46	82.55	
SearchQA	Ens	98.0	86.82	87.31	85.79	84.7	85.61	65.65	84.1	56.86	83.65	48.46	83.16	38.73	82.36	
	G@Ens$_{cov}$	-	86.83	-	86.05	-	85.87	-	84.52	-	84.03	-	83.59	-	82.94	17.05
	Given (G)	96.67	67.31	86.89	65.05	82.54	63.82	68.81	60.39	61.44	58.39	55.11	56.48	47.12	54.03	
TriviaQA	Ens	98.01	67.58	89.88	65.71	87.99	65.15	72.31	60.95	65.0	59.13	58.47	56.9	49.67	54.38	
	G@Ens$_{cov}$	-	67.64	-	65.76	-	65.3	-	61.38	-	59.25	-	57.55	-	54.94	9.5
	Given (G)	92.37	63.78	79.04	59.99	74.87	58.77	60.60	53.51	54.03	51.00	47.94	48.31	41.70	45.27	
NQ	Ens	94.59	64.35	83.46	60.82	81.32	60.16	64.83	54.7	58.05	52.17	51.8	49.8	44.33	46.31	
	G@Ens$_{cov}$	-	64.43	-	61.31	-	60.79	-	55.03	-	52.61	-	50.01	-	46.82	13.4
	Given (G)	95.74	88.46	81.17	87.38	76.11	87.33	62.34	86.23	53.69	85.38	48.77	84.45	43.05	85.01	
DROP	Ens	97.6	88.48	85.63	87.72	83.17	87.28	65.34	86.15	56.55	85.65	50.37	84.54	44.78	84.99	
	G@Ens$_{cov}$	-	88.47	-	87.72	-	87.52	-	86.05	-	85.63	-	84.54	-	84.84	1.57
	Given (G)	97.20	68.68	87.87	66.41	84.21	65.82	71.09	62.42	64.16	61.47	57.16	59.91	50.03	58.46	
DuoRC	Ens	98.0	68.86	90.34	67.11	88.61	66.84	73.82	63.36	66.96	62.19	59.96	60.78	51.57	58.4	
	Original@cov	-	68.91	-	67.18	-	66.69	-	63.18	-	61.79	-	60.07	-	58.91	-1.69
	Given (G)	94.34	67.14	80.9	63.32	75.65	61.92	57.49	56.02	49.63	52.14	41.45	51.04	34.07	50.00	
TBQA	Ens	95.94	67.55	84.3	64.17	81.1	63.33	62.28	56.94	53.96	54.25	45.78	52.33	37.72	51.15	
	G@Ens$_{cov}$	-	67.45	-	64.33	-	63.38	-	57.05	-	54.38	-	52.03	-	50.53	-6.93

Table 45. Performance of Ensembling (Mean) as a post-abstention method for selected abstention thresholds. The QA model is trained using SQuAD training data and evaluated on SQuAD (in-domain) and 10 out-of-domain datasets. The last column corresponds to the overall improvement aggregated over all confidences ranging from 0 to 1 at an interval of 0.02. ↓ and ↑ indicate that lower (risk) and higher (coverage, risk improvement) values are better respectively.

Dumais, et al. 2020; Y. Zhang et al. 2020; Zamani, Lueck, et al. 2020). These techniques can be leveraged/adapted for the post-abstention task.

Test-time adaptation is another promising research area in which the model is adapted at test-time depending on the instance. This is being studied in both computer vision (Dian Chen et al. 2022) and language processing (X. Wang et al. 2021; Banerjee, Gokhale, and Baral 2021).

Cascading systems in which stronger and stronger models are conditionally used for inference is also an interesting avenue to explore with respect to Post-Abstention (Varshney and Baral 2022; Lei Li et al. 2021; Varshney, Luo, and Baral 2022).

6.9 Coverage 100% for Human Intervention Methods

We believe that the ability to identify situations when there is no good answer in the top N returned candidates is a very difficult task (for the humans also) and it requires even more cognitive skills than just selecting the best answer from the provided answer candidates. Because of this reason, the coverage is 100%.

6.10 Comparison with Other Selective Prediction Methods

In this work, we presented a new QA setting and studied the performance of several baseline methods for this task. The focus of this work is on studying the risk improvement that can be achieved in this problem setup. We consciously do not pitch the approaches for this task as competitors of the existing selective prediction approaches. In fact, these approaches are **complimentary** to the selective prediction approaches. A post-abstention method can be used with any selective prediction method as the first step.

6.11 Conclusion and Discussion

In this work, we formulated 'Post-Abstention', a task that allows re-attempting the abstained instances of the given selective prediction system with the aim of increasing its *coverage* without significantly sacrificing the *accuracy*. We also explored several baseline methods for this task. Through comprehensive experiments on 11 QA datasets, we showed that these methods lead to considerable performance improvements in both in-domain and out-of-domain settings. We further performed a thorough analysis that resulted in several interesting findings.

Looking forward, we believe that our work opens up several avenues for new research, such as exploring *test-time adaptation*, *knowledge hunting*, and other human intervention techniques like *asking clarification questions* as post-abstention methods (discussed in Section 6.8). Studying the impact of composite or sequential application of multiple post-abstention methods in another promising direction. Furthermore, prior selective prediction methods can also be repurposed and explored for this task. We plan to pursue these crucial research directions in our future work. Finally, we hope our work will encourage further research in this important area and facilitate the development of more reliable NLP systems.

Chapter 7

IMPROVING THE LLM INFERENCE EFFICIENCY BY ENABLING INTERMEDIATE LAYER DECODING

Large Language Models (LLMs) have achieved remarkable performance across a wide variety of tasks; however, their large size makes their inference slow and computationally expensive. Focusing on this problem, we study instruction tuning LLMs with additional explicit **L**osses from the **I**n**T**ermediate lay**E**rs (`LITE`) and show that it enables these layers to acquire 'good' generation ability without affecting the generation ability of the final layer. We then perform '*dynamic confidence-based early exiting*' at token level from the intermediate layers which improves the computational efficiency of text generation without sacrificing the quality of the generation. We conduct comprehensive experiments by instruction tuning LLaMA-2 models on the Alpaca dataset and evaluate on four different instruction test sets. We show that dynamic early exiting achieves consistent and considerable inference cost improvements (37.86% for 7B and 46.35% for 13B model) while maintaining the generation quality. We further conduct a thorough analysis of the results and dissect the efficiency improvements which reveals several important findings.

7.1 Introduction

Recently developed LLMs (Touvron et al. 2023; OpenAI 2023; Chowdhery et al. 2022; Rae et al. 2021; Smith et al. 2022) have revolutionized the field of natural language processing and achieved remarkable performance across a wide variety of tasks. 'Instruction Tuning' further teaches these models to follow the user's instruction provided in natural language (Jason Wei et al. 2022; Mishra, Khashabi, et al. 2022; Sanh et al. 2022; Y. Wang et al. 2022; Chung et al. 2022). Despite all the notable abilities of these models, their large size (number of parameters) makes their inference slow and computationally expensive which poses a practical challenge limiting their widespread adoption in resource constrained applications. Focusing on the above problem, in this work, we investigate instruction tuning LLMs in a way that enables intermediate layer decoding for efficiently generating text without compromising the quality of the generation.

We first show that in standard instruction tuning, only the final layer of the model acquires the ability to generate '*quality*' text while the representations of the intermediate layers (when passed through the language modeling head) fail to do so. This restricts decoding from these intermediate layers without degrading the generation quality. Addressing this point, we instruction tune LLMs with additional explicit **L**osses from the **I**n**T**ermediate lay**E**rs (`LITE`) and show that it enables these layers to acquire 'good' generation ability. Importantly, we show that these layers acquire this ability without affecting the generation ability of the final layer; however, as expected, their generation ability still remains slightly inferior to the generation ability of the final layer. Thus, decoding the complete response from intermediate layers improves the efficiency of inference but still results in degradation in the quality of the response.

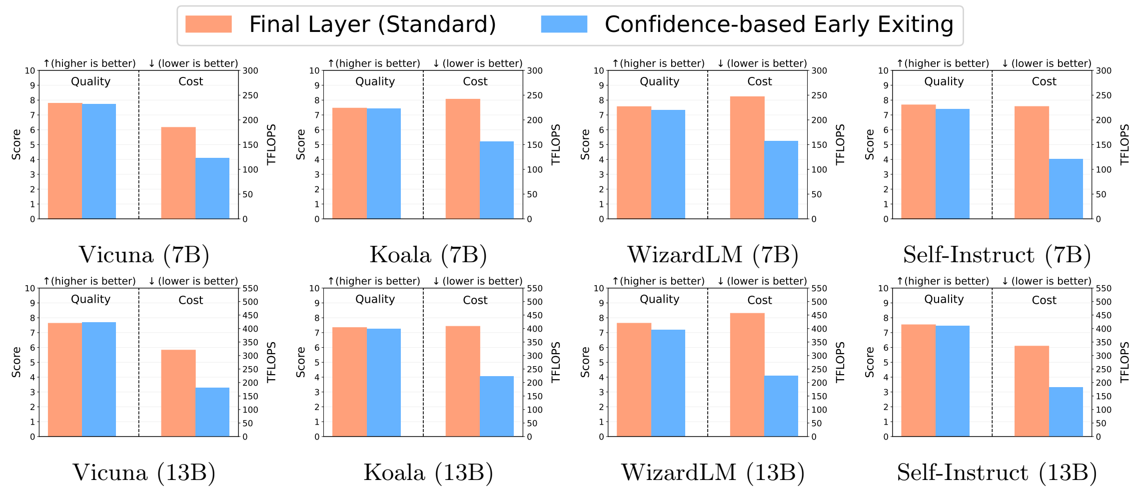


Figure 32. Comparing the quality of the responses and the inference cost of (i) the standard generation from the final layer (orange) and (ii) the dynamic early exiting method (blue) on model tuned with `LITE`. The top and the bottom rows show the effectiveness on four different test sets for the LLaMA-2 7B and 13B models, respectively.

Addressing the above limitation, we show that (a) `LITE` greatly aligns the intermediate layers' token prediction with that of the final layer and (b) the intermediate layers' token prediction probabilities provide a strong signal of this alignment. Building on these findings, we perform '*dynamic confidence-based early exiting*' at token level from the intermediate layers which improves the efficiency of inference while maintaining the generation quality.

We conduct comprehensive experiments by instruction tuning LLaMA-2 models (Touvron et al. 2023) on the widely used Alpaca dataset (Taori et al. 2023) and holistically evaluate on four different human-instruction test sets including Vicuna, WizardLM, Koala, and Self-Instruct. Figure 32 compares the quality of responses (evaluated using the Claude model as detailed in Section 7.4) and the inference cost (measured in FLOPs) of the (i) standard generation method from the final layer with (ii) the dynamic early exiting method. It shows that dynamic early exiting achieves consistent and considerable inference cost improvements (37.86% for 7B and 46.35% for 13B model on average) while maintaining the generation quality.

We further perform a thorough analysis of the results over several important aspects, such as, comparing the semantic similarity between the responses generated from the final layer and the early exiting method, and dissecting the efficiency improvements by comparing the number of tokens generated in the outputs. We also discuss the potential of intermediate layer decoding in '*speculative sampling*' and '*hallucination detection*'.

In summary, our work contributes to improving the efficiency of LLM inference while maintaining the generation quality, a crucial step en route to enabling their widespread adoption.

7.2 Instruction Tuning with `LITE`

Instruction Tuning (`IT`): One of the major reasons that necessitate instruction tuning of LLMs is the mismatch between their pre-training objective and the users' objective, i.e., LLMs are typically trained on minimizing the word prediction error while users want the model to follow their instructions. To this end, an instruction tuning dataset is collected and a pre-trained model is fine-tuned in a supervised manner (Mishra, Khashabi, et al. 2022; Chung et al. 2022). Loss calculation during instruction tuning of a typical decoder-only LLM (LLaMA in this case) is shown in Figure 33 (left). The model consists of a stack of decoder layers followed by a language modeling head which outputs the probability distribution over the vocabulary tokens as its prediction. During the supervised fine-tuning, the loss over the output tokens is backpropagated from the final layer of the model:

$$Loss(y_{1:M}) = -\sum_{t=1}^{M} log\ p(y_t|y_{<t})$$

IT with LITE: We show that in standard instruction tuning, only the final layer of the model acquires the ability to generate 'quality' text while the representations of the intermediate layers (when passed through the language modeling head) fail to do so (Section 7.5.1). In other words, it does not explicitly teach the intermediate layers of the tuned LLM to generate tokens. This restricts decoding from these intermediate layers without degrading the generation quality.

We note that during tuning, the same language modeling head (that is used with the final layer) can also be used with the intermediate layers to obtain the losses of those layers. Thus, this does not impact the number of parameters of the model. To this end, we calculate a weighted aggregation of the losses from the intermediate layers (including the final) to calculate the overall loss value:

$$Loss = \frac{\sum_{i=1}^{N} w_i Loss_i}{\sum_{i=1}^{N} w_i}$$

where N is the number of layers, w_i is the weight of the ith layer, and $Loss_i$ is the cross entropy loss of the ith layer as shown in Figure 33.

During training, we use the representations of the intermediate layers and calculate the loss from these layers at the end. We note that this is a general formulation as it

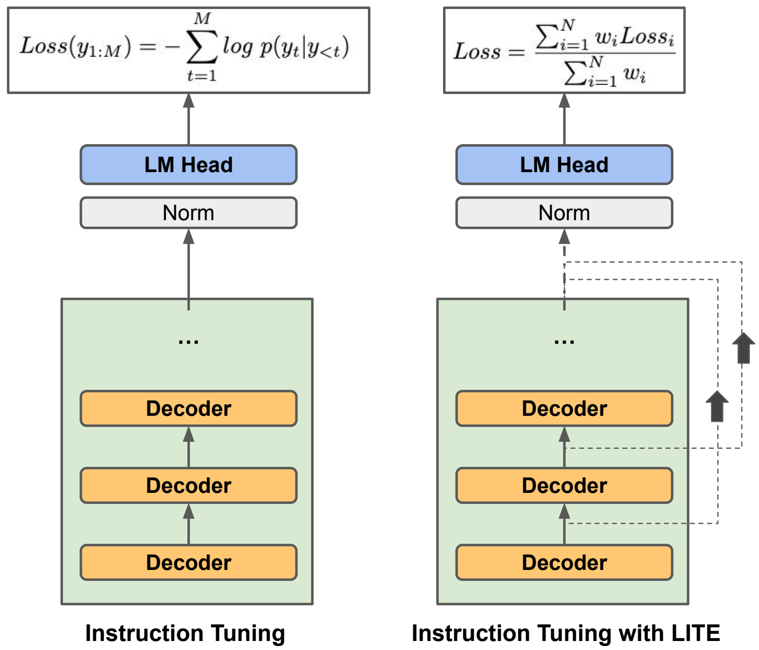


Figure 33. Loss calculation for standard instruction tuning (left) and instruction tuning with additional explicit losses from the intermediate layers `LITE` (right).

captures a variety of scenarios including the standard fine-tuning in which the loss is calculated only from the last layer (i.e., $w_{1:N-1} = 0$ and $w_N = 1$). Furthermore, this formulation also allows aggregating losses from only the selected intermediate layers instead of all the layers by accordingly defining the LM head pathways and the w_i values. In Section 7.5.2, we will show that this formulation while enabling the intermediate layers with 'good' generation ability does not adversely affect the final layer's generation ability. Furthermore, as expected, the quality of generation typically improves with the layer number as the later layers have more capacity to learn.

7.3 Making Inference Efficient

In this section, we first detail auto-regressive inference and then describe early exiting techniques.

Auto-Regressive Inference: It refers to the process of generating a sequence of tokens where each token is generated based on the preceding tokens in the sequence. For generating a token, the model takes the input (including the previously generated tokens) and runs a forward pass in which the input is fed to the model and passed sequentially along its layers until the probabilities for the next token are predicted (called as **logits**). Chaining model forward passes with next token selection iteratively leads to the generation of text. In greedy decoding, the token with the highest probability is selected as the next word prediction at each timestep.

7.3.1 Fixed Early Exiting

Since instruction tuning with `LITE` enables the intermediate layers to acquire 'good' generation ability, the computations during inference can be terminated at a pre-specified intermediate layer (referred to as **exiting layer**) and the language modeling head can be used to predict the next token. This saves the computations of the remaining layers that follow the specified exiting layer and thus it improves the efficiency of inference.

Though this method of fixed early exiting leads to improvement in the efficiency of inference, it is bound to result in some degradation in the quality of the generation as the generation ability of an intermediate layer still remains inferior to the generation ability of the final layer. However, the quality of generation typically improves with the layer number as the later layers have more capacity.

7.3.2 Dynamic Confidence-Based Early Exiting

Addressing the limitation of the fixed early exiting method, we study a dynamic early exiting method that decides the exiting layer for a token prediction based on the intermediate layer's probability of the prediction (softmax over the logit values).

This is motivated by our following two findings:

(a) Instruction Tuning with `LITE` greatly aligns the intermediate layers' token prediction with that of the final layer (Section 7.5.3) and

(b) The intermediate layers' token prediction probabilities (referred to as **confidence**) provide a strong signal of this alignment (Section 7.5.4).

Building on these two findings, we perform 'dynamic confidence-based early exiting' at <u>token level</u> from the intermediate layers which improves the efficiency of inference while maintaining the generation quality. Specifically, a set of intermediate layers with their corresponding confidence thresholds are defined and at inference time, the exiting decision for a prediction is taken by comparing the intermediate layer's prediction confidence against its corresponding threshold. This enables the model to perform efficient inference without degrading the generation quality. Note that this method does not introduce new parameters and uses the softmax probability to make the exiting decision. We study this exiting method for inference without KV caching.

Our work differs from existing work in the following aspects:

(1) Firstly, most of the existing work in early exiting focuses on improving the efficiency of encoder-only models (like BERT) or encoder-decoder models (like T5); our work focuses on the current state-of-the-art decoder-only LLMs (LLaMA-2). Furthermore, we focus on the instruction tuning setting with text generation, unlike prior work that focused on solving simpler tasks like GLUE classification or QA. (2) Early exiting methods typically require training additional classifiers for the intermediate layers, however, in this work, we use the same shared language modeling head at all the layers; thus, we do not introduce new model weights. (3) For leveraging the intermediate layers for decoding, we enable them to acquire generation ability by instruction tuning with LITE, unlike other methods that use a pre finetuned model in which these layers have poor generation ability as we show in Section 7.5.1. (4) Existing methods typically require complex architectural modifications, pruning, saliency quantification, or training new parameters. In contrast, our method (both for tuning and inference) is simple and easy to implement and yet achieves considerable benefits. (5) Existing methods typically require training a separate model for each computation budget; however, in our method, the same model can be adapted to meet all the computation constraints (by varying the exiting confidence thresholds). (6) The computational efficiency often comes with a compromise in performance. However, our method maintains the generation quality while providing efficiency benefits.

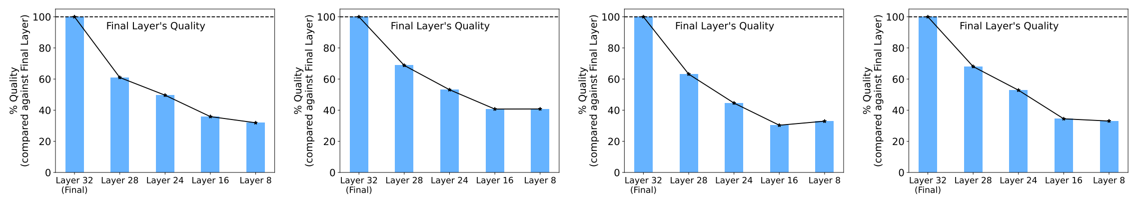
7.4 Experimental Setup

Instruction Tuning: We instruction tune the LLaMA-2 models (Touvron et al. 2023) (7B and 13B) with the widely used Alpaca dataset (Taori et al. 2023). Alpaca consists of 52K instruction-following demonstrations generated using the self-instruct (Y. Wang et al. 2023) technique. In IT with `LITE` for 7B model (32 total layers), we aggregate losses from the following selected intermediate layers: (8, 12, 16, 20, 24, 28) along with the final layer and use equal weights in loss calculation. Similarly, for the 13B model (40 total layers), we use (8, 12, 16, 20, 24, 28, 32, 36) layers. We perform full parameter fine-tuning on 4 A100 GPUs.

We skip selecting the initial layers because they have a limited capacity to learn and thus can not give good token predictions. Furthermore, we select layers at an interval of 4 so that at inference time, the model can do enough reasoning/interactions between two consecutive checkpoints. Otherwise, checking at every layer can result in computational overhead. We train this model for 5 epochs so that it achieves training loss comparable to standard tuning.

We present all the results corresponding to this tuning configuration in the main paper and present the study corresponding to weighted LITE in the Section 7.8.

Evaluation Datasets: To perform holistic evaluation, we experiment with four different human-instruction test sets including Vicuna (Chiang et al. 2023), Self-Instruct (Y. Wang et al. 2023), Koala (Geng et al. 2023), and WizardLM (C. Xu et al. 2023). We select these evaluation test sets as they can together cover a large number and types of instructions thus resulting in a comprehensive evaluation. Table 49 shows the statistics of the datasets.



(a) Vicuna (b) Koala (c) WizardLM (d) Self-Instruct

Figure 34. Demonstrating quality comparison of the output of intermediate layers (generated via fixed exiting) against the final layer's generation of the model tuned with standard instruction tuning.

Evaluation Methodology: The evaluation of the instruction-following ability of LLMs is challenging due to the existence of multiple correct responses to an input and the infeasibility of reproducing human evaluations. Addressing this problem, recent works have started to rely on automatic evaluations using LLMs (Zheng et al. 2023; Chiang et al. 2023). Specifically, we use Claude LLM (Bai et al. 2022) as a judge to compare the quality of responses of two models on a given instruction. We note that these LLMs have been shown to be vulnerable to position bias in their judgment (P. Wang et al. 2023). To circumvent this bias, we evaluate a response pair with both orderings of the responses and then aggregate the judgment scores. We provide the prompt for comparing the quality of the responses of two models in Section 7.6.

7.5 Results and Analysis

In this section, we first demonstrate the inability of the intermediate layers of the model tuned with standard `IT` to generate 'quality' text (7.5.1). Then, we show the impact of `IT` with `LITE`: it does not adversely affect the generation quality of the final layer (7.5.2), it aligns the intermediate layers' token predictions with the final layer (7.5.3), and the corresponding prediction confidence values provide a strong signal of the alignment (7.5.4). These findings motivate dynamic confidence-based early exiting. Finally, we show the effectiveness of the method in improving the efficiency of inference while maintaining the generation quality (7.5.5). To avoid repetition, we present results for the 7B model in the main paper and for the 13B model in 7.7.8.

7.5.1 Generation Ability of Intermediate Layers

In order to obtain the text (sequence of tokens) generated via fixed exiting from an intermediate layer, we apply the normalization (RMSNorm) followed by the language modeling head to the representations of that intermediate layer and skip the computations of the layers following the exiting layer (as detailed in Section 7.3.1). For the model tuned with the standard instruction tuning, we compare the quality of the text (as detailed in Section 7.4) generated from different intermediate layers against the final layer's generation in Figure 34. As expected, the intermediate layers generate text of considerably degraded quality and this quality drops as the layer number decreases.

This demonstrates that with standard instruction tuning, only the later layers (primarily the final layer) of the model acquire the ability to generate 'quality' text while the representations of the intermediate layers (when passed through the language modeling head) fail to do so. Thus, for such a model, the early exiting method saves the inference computation cost but considerably degrades the generation quality. This restricts employing such early exiting techniques for the model tuned with standard instruction tuning. We show examples of responses obtained via fixed early exiting from different intermediate layers in Section 7.7.1.

We perform instruction tuning with `LITE` to enable the intermediate layers to acquire 'good' generation ability. Importantly, we show that these layers acquire this ability without affecting the generation ability of the final layer (Section 7.5.2).

7.5.2 Impact of LITE on the Final Layer

In Figure 35, we compare the quality of responses of (a) the model tuned using standard instruction tuning (IT) and (b) the model tuned using IT with `LITE`. Note that the responses for both these models correspond to their respective final layer's output. From the figure, it can be observed that for all the datasets, the outputs of both models are of comparable quality which shows that tuning with `LITE` does not adversely affect the generation ability of the final layer of the model.

Next, we demonstrate two important characteristics of instruction tuning with `LITE` (in 7.5.3 and 7.5.4) that motivate us to study dynamic confidence-based early exiting from the intermediate layers.

7.5.3 'Alignment' of Intermediate Layers

We define percentage 'alignment' of a layer as the measure of how often the token predictions of that layer match with that of the final layer (given same input prefixes). For this study, we do not do early exiting, instead we just use the representation of each intermediate layer and pass it through the LM head to obtain the corresponding token prediction of each layer. Note that for generating the next token, we follow the standard generation methodology and append the predicted token of the last layer to the input to obtain the token prediction of all the layers given the same input prefixes.

In Figure 36, we plot the percentage alignment of token predictions of all intermediate layers with the token predictions of the final layer. The figure shows the percentage alignment of (i) the model tuned using standard IT (orange) and (ii) the model tuned using IT with `LITE` (blue). We show this result aggregated over all the output token predictions for all the inputs of the corresponding dataset.

We draw the following inferences:

(a) The predictions of the intermediate layers of the model tuned with LITE align well with the final layer, i.e., given a prefix, the intermediate layers' token predictions match quite well with the final layer's token prediction. In contrast for the model tuned using IT, the token predictions of the intermediate layers do not align well with the final layer's token predictions.

(b) As the layer number increases, the % alignment also increases, i.e., given a

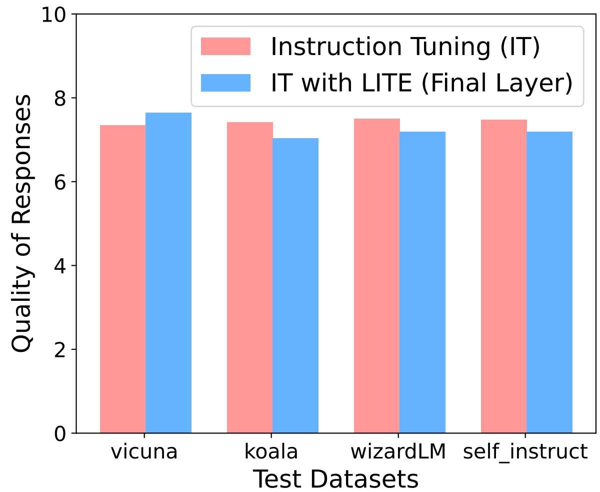


Figure 35. Comparing quality of responses of (a) model tuned using IT and (b) model tuned using IT with `LITE`. The outputs of the models are of comparable quality.

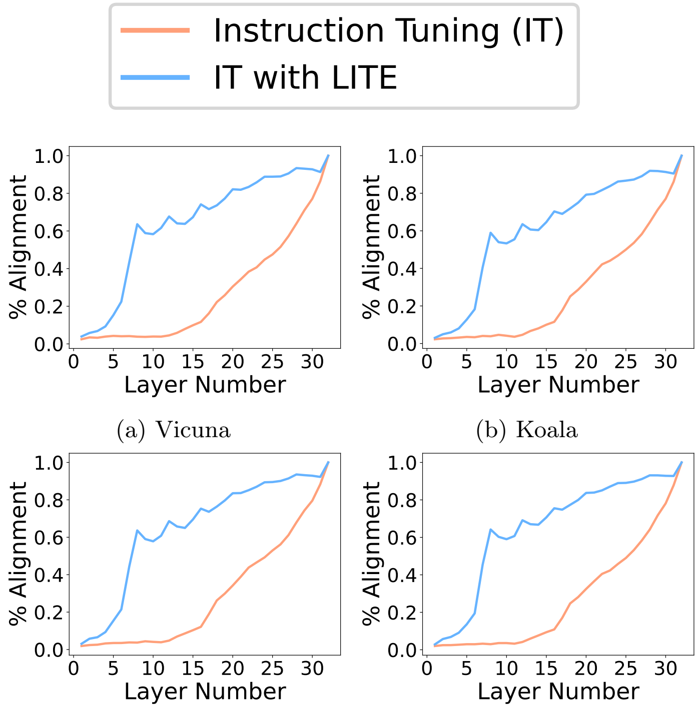
prefix, the predicted token of the later layers shows higher alignment (with the final layer) than the initial layers.

(c) There are some peaks in the curve for IT with `LITE` which correspond to the selected layers from which the loss is aggregated during tuning, i.e., these layers show higher alignment as expected.

In summary, this study demonstrates that IT with LITE greatly aligns the token predictions of intermediate layers with that of the final layer.

7.5.4 Token Probability and Alignment

We plot the relationship between the token prediction confidence (softmax over the logits of the LM head) of the intermediate layers and the percentage alignment with the token prediction of the final layer. Figure 37 shows this plot for the model



(a) Vicuna (b) Koala

(c) WizardLM (d) Self-Instruct

Figure 36. Comparing percentage 'alignment' of intermediate layer token predictions with the token predictions of the final layer for the model tuned using IT (orange) and the model tuned using IT with LITE (blue).

tuned with `LITE`. The figure shows that in IT with `LITE`, the intermediate layers' token prediction probabilities provide a strong signal of alignment, i.e., a high token prediction confidence implies a higher likelihood of its alignment with the token prediction of the final layer. It also shows that with the increase in the layer number, the percentage alignment typically increases at the same confidence values. In contrast, in standard instruction tuning (IT), the confidence is not well correlated with the percentage alignment as we show in Section 7.7.6.

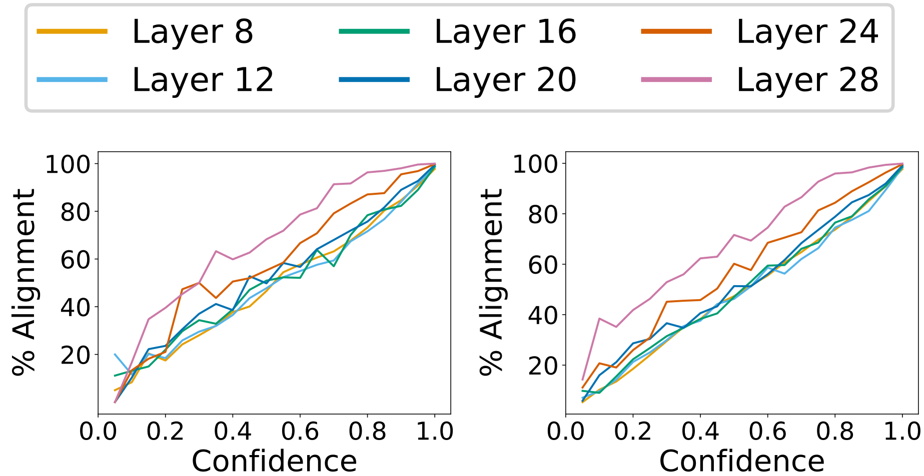
7.5.5 Effectiveness of Dynamic Early Exiting

Motivated by the findings of the previous two subsections (7.5.3 and 7.5.4), we perform dynamic confidence-based early exiting at token-level, i.e., we exit when the token prediction confidence of the intermediate layer is sufficiently high (thus it is likely to align with the final layer's prediction).

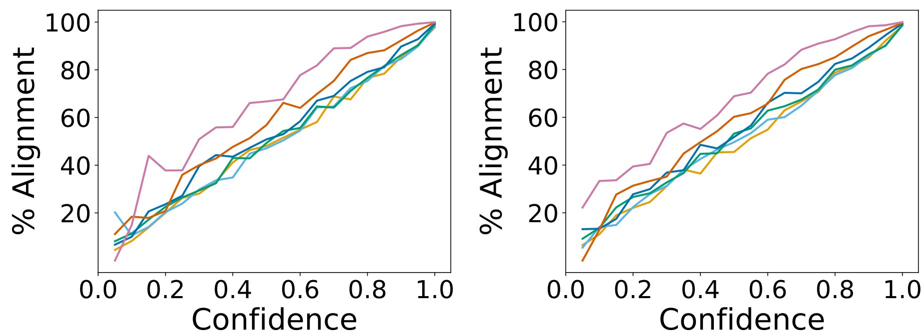
To this end, from the confidence vs percentage alignment curve, we identify a confidence threshold for each layer where the alignment is > 95%. Specifically, we use the following thresholds: Layer 8: 0.95, Layer 12: 0.95, Layer 16: 0.9, Layer 20: 0.9, Layer 24: 0.8, and Layer 28: 0.7.

In the main paper, we present the results and analysis for the aforementioned configuration. However, we note that a different threshold configuration can also be used for inference. For instance, a more aggressive configuration with lower thresholds (shown in Section 7.7.7) leads to even more cost improvements (49.92%); though it slightly drops the quality of generation (5.34%). The trade-off between quality and cost can be balanced depending on the application requirements. For example, applications with quality tolerance or resource limitations can keep low threshold to achieve higher cost improvements.

Dynamic confidence-based early exiting: At a selected layer, we pass its representations through the LM head, calculate the softmax logit value, and compare it with the corresponding confidence threshold of the layer. If it surpasses



(a) Vicuna (b) Koala



(c) WizardLM (d) Self-Instruct

Figure 37. Demonstrating trend of token prediction confidence of the intermediate layers and the % alignment with the final layer for model tuned with `LITE`.

the threshold value then we exit from that layer and proceed to generate the next token, otherwise, we repeat this process at the next selected layer.

Figure 32 (in Section 7.1) compares the quality of responses and the inference cost (measured in FLOPs) of the standard generation method (final layer) with the dynamic early exiting method. It shows that the dynamic early exiting method achieves consistent and considerable cost improvements (37.86% for 7B and 46.35% for 13B model on average) while maintaining the generation quality. Table 46 shows

Test Dataset	Inference Cost Improvement (%)
Vicuna	33.39 %
Koala	35.40 %
WizardLM	36.12 %
Self Instruct	46.54 %

Table 46. Percentage improvements in the inference cost (measured in FLOPs) with dynamic early exiting.

the percentage improvements in inference cost for each test set individually. We note that we use FLOPs as the metric of showcasing inference efficiency improvements because it is hardware independent, unlike latency.

7.5.5.1 Semantic Similarity of the Responses

In addition to comparing the quality, we also compare the semantic similarity between the responses of the final layer and the dynamic early exiting. Table 47 shows the semantic similarity (calculated using the 'en_core_web_sm' spacy model) for the four datasets. It shows that there is a large semantic similarity between the responses as the values are closer to 1. This implies that dynamic early exiting maintains the semantics of the responses while providing efficiency benefits. Section 7.7.4 shows examples of responses from both the last layer and the dynamic early exiting method.

Test Dataset	**Semantic Similarity**
Vicuna	0.9135
Koala	0.8940
WizardLM	0.9020
Self Instruct	0.9001

Table 47. Semantic similarity between the final layer's and the dynamic early exiting responses on test sets.

7.5.5.2 Dissecting the Cost Improvements

In Figure 38, we compare the average number of tokens generated in the final layer's responses and the dynamic early exiting responses. It shows that both the methods generate a comparable number of tokens in their respective outputs. This asserts that the cost improvement resulting in dynamic early exiting is because of the reduced computations and not due to generating a lesser number of tokens.

7.5.5.3 Contribution of Different Exiting Layers

Figure 39 shows the percentage of token outputs from different exit layers. Note that this is aggregated across all the token positions. This shows that the model exits a considerable percentage of times from the intermediate layers (while maintaining the generation quality) which further justifies the improvement in inference efficiency. We further conduct several interesting studies and analyses of the results and present them in Section 7.7.

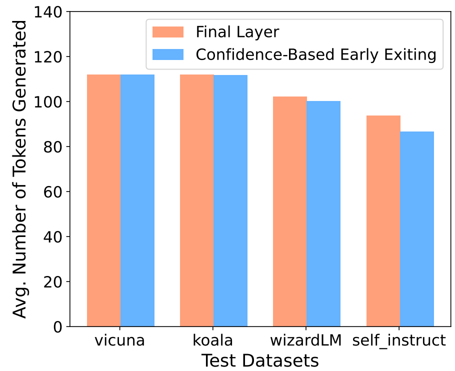


Figure 38. Comparing the average number of tokens generated in the final layer's responses and the dynamic early exiting responses for the four datasets.

7.5.5.4 Effectiveness at Category Level

Vicuna and WizardLM datasets also provide the category corresponding to different test instances. To this end, we present category-level quality and inference cost results for these datasets in Figure 40 and 41. On average, the method results in cost improvement of 33.39% on Vicuna and 36.12% on WizardLM.

7.6 Evaluation Methodology

We use the following prompt for comparing the quality of responses:

```
Human: You are a helpful and precise assistant for checking the quality of the answer.
[Question]
{question} [The Start of Assistant 1's Answer]
{answer_1}
[The End of Assistant 1's Answer]
[The Start of Assistant 2's Answer]
{answer_2}
[The End of Assistant 2's Answer]
We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above.
Please rate the helpfulness, relevance, accuracy, level of details of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.
Please output a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores are separated by a space.
Assistant:
```

To circumvent the position bias, we evaluate a response pair with both orderings of the responses and then aggregate the judgment scores. We use the Claude API with temperature set to 0 and modelId to 'anthropic.claude-v1'.

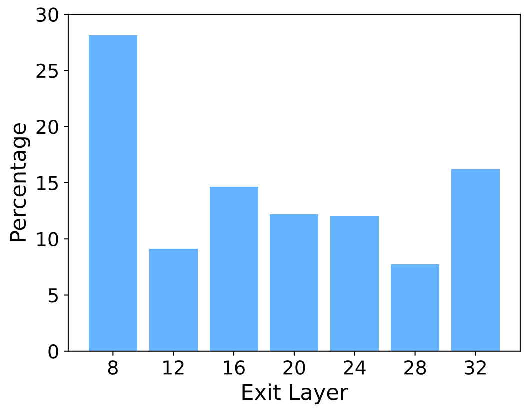


Figure 39. % of token outputs from different exiting layers in the proposed method.

7.7 Additional Results

7.7.1 Inability of the Intermediate Layers of the Model Tuned with Standard Instruction Tuning to Generate 'High-Quality' Text

In this Subsection, we show examples of responses of intermediate layers (fixed exiting) of the model tuned with standard instruction tuning (IT).

Question: What are the differences between plant-based and animal-based protein sources?

Layer 32 (Final): The main difference between plant-based and animal-based protein sources is that plant-based proteins are found in plant-based foods, such as legumes, nuts, and seeds, while animal-based proteins are found in animal-based foods, such as meat, eggs, and fish. Plant-based proteins typically contain fewer calories, fat, and ...

Layer 28: plant- based proteins sources such as legumes, legumes, and nuts contain essential amino achren needed for growth and repair of cellsular structures. animal based proteins sources such as meat, eggs, and fish contain additional nutrients such as vitamin B Complex vitamin D vitamin Iron and calcium needed for optimal health.

Layer 24: plant based prote source protein source diversity, quality proteinsrichness quantityquantityquantityquantity quantity quantity quantity quantity quantity q ...

Layer 20: difference between Plant Based vs animal based protein sourceage oldenhamptonensis proportionalityunicíclesqueque ...

Question: Imagine you are a time traveler from the year 3000. What technological advancements would you tell people about?

Layer 32 (Final): From the year 3000, I would tell people about the advancements in technology that have made life easier and more efficient. I would talk about the autonomous cars, the implanted chips that allow people to access the internet and doctors to monitor patients remotely, the 3D-printed homes and the list goes on. I would also tell people about the advancements in renewable energy and how we are able to generate electricity from the sun and the wind.

Layer 28: From the year 3000 perspective, some technological advances might include flying cars, robotic vac vacuifiers, implanted implants used for implantable devices such as pacemakers, implanted cameras implanted microscopes implanted anything else you can think of implanted implanted implanted ...

Layer 24: techn adv advances techn advances techn advances techn ...

Layer 20: futureproofing@gmailgmail domainainsights intoxicatednessipag eantryonswardswardsonswardsonswardso ...

The above examples and the results in Section 7.5.1 show that in standard instruction tuning, the intermediate layers do not acquire good generation ability as the generated responses are of poor quality.

In the next subsection, we illustrate examples that show that IT with LITE enables the intermediate layers to acquire good generation ability.

7.7.2 Intermediate Layers Acquire Good Generation Ability in IT with LITE

Question: What are the differences between plant-based and animal-based protein sources?

Layer 32 (Final): Plant-based protein sources are mainly found in foods such as legumes, nuts, and seeds, while animal-based protein sources are found in foods such as meat, fish, eggs, and dairy products. Plant-based proteins are generally lower in calories ...
Layer 28: Plant-based protein sources are mainly found in foods such as legumes, nuts, and seeds, while animal-based protein sources are found in foods such as meat, fish, eggs, and dairy products. Plant-based proteins are generally lower in calories and fat ...
Layer 24: Plant-based protein sources are mainly found in foods such as legumes, nuts, and seeds, and are typically lower in calories and fat than animal-based protein sources. Plant-based proteins are also rich in vitamins, minerals, and antioxidants, and are a ...
Layer 20: Plant-based protein sources are found in many different foods, including legumes, nuts, seeds, and seeds. They are usually lower in calories and fat than animal-based protein sources. Plant-based proteins are also rich in fiber and contain ...

Question: Imagine you are a time traveler from the year 3000. What technological advancements would you tell people about?

Layer 32 (Final): From the year 3000, I would tell people about the incredible advancements in technology that have been made over the centuries. We take many things for granted now, such as the internet and autonomous vehicles, but these technologies ...

Layer 28: From the year 3000, I would tell people about the advancements in artificial intelligence, robotics, and genetic engineering. In particular, I would tell people about the advancements made in AI, such as the development of autonomous cars, virtual assistants, and smart homes. I would ...

Layer 24: From the year 3000, I would tell people about the incredible advancements in technology that have been made over the centuries. We are now able to control our environment by using thought-controlled prostheses, use genetic ...

Layer 20: In the 3000s, we have developed advanced artificial intelligence, self-driving cars, and teleportation of particles. We have also made significant progress in quantum computing, which ...

The above examples illustrate that IT with LITE enables the intermediate layers to acquire good generation ability. This is because the losses are aggregated from these intermediate layers during instruction tuning.

7.7.3 Poor Quality of Responses With Dynamic Early Exiting from Model Tuned with Standard Instruction Tuning (IT)

Here, we show examples of the responses generated via dynamic confidence-based early exiting from the model tuned with standard instruction tuning (IT). Note that this is without `LITE`.

It can be observed from these examples that the early exiting responses are of poor quality. They suffer from a number of issues including degeneration such as unwanted token repetitions, and lack of coherence and structure. We note that this can be attributed to the poor generation ability and alignment of the intermediate layers.

Question: What are the most effective ways to deal with stress?

Layer 32 (Final): The most effective ways to deal with stress include identifying sources of stress, developing coping strategies, maintaining a balanced diet, exercising regularly, getting enough sleep, and taking time to relax and do something enjoyable. Taking time to practice deep breathing and meditation, talking to a friend or therapist, and using humor are also helpful.

Dynamic Early Exiting with IT model: cop coping strategies include identifying sources of stress, developing cop coping skills such as cop coping and relaxation techniques such as meditation and med meditation med meditation med meditation med meditation med meditation med meditation med meditation med meditation med meditation ...

Question: Can you explain the basics of quantum computing?

Layer 32 (Final): Quantum computing is a form of computing that uses quantum-mechanical phenomena, such as superposition and entanglement, to perform operations on data. It enables computation at a much faster rate than traditional computers and can solve problems that are intractable on classical computers. Quantum computing uses qubits2̌014quantum bits2̌014which can be in a state of 0 or 1, instead of the traditional bit of 0 or 1. Qubits can be used to represent data and perform operations on it, such as addition, subtraction, and multiplication.

Dynamic Early Exiting with IT model: quantum computing basics quantum computing basics explain the basics of quantum computing including what it is, how it works, why it matter, how it can be used for computing, and what its future may hold. quantum computing basics quantum computing basics explain the basics of quantum computing including what it is, how it works, why it matter, how it can be used for computing ...

This further showcases the importance of LITE during instruction tuning as it allows the intermediate layers to acquire good generation ability, thus enabling opportunities for early exiting without compromising the quality of the generation.

7.7.4 Examples of Responses with Dynamic Early Exiting for Model instruction tuned with LITE

Here, we show examples of responses of the final layer and the dynamic early exiting method with the model instruction tuned using `LITE`.

Question: What are the main differences between Python and JavaScript programming languages?

Layer 32 (Final): The main differences between Python and JavaScript programming languages are their syntax, data types, data structures, and logic. Python is a high-level, interpreted language that is focused on readability and simplicity, while JavaScript is a low-level, interpreted language that is focused on adding interactive elements to web pages. Python has strong support for data typing, while JavaScript has weaker support for data types. Python is more suitable for creating scripts and applications, while JavaScript is more suitable for creating interactive web pages.

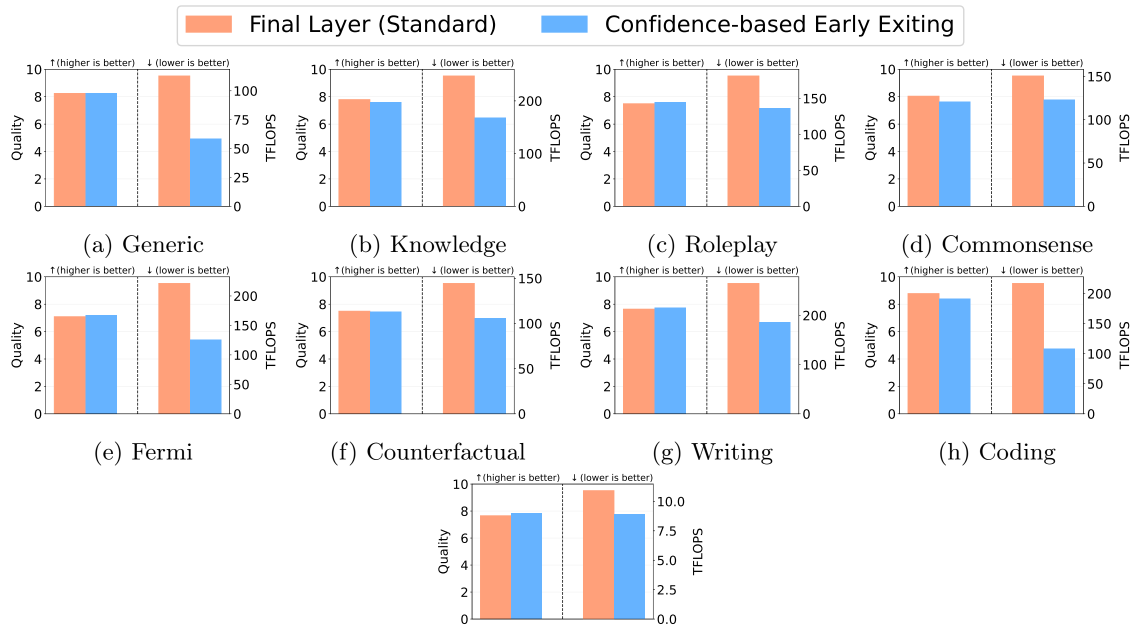
Dynamic Early Exiting: The main differences between Python and JavaScript programming languages are their syntax, data types, data structures, and logic. Python is a high-level, interpreted language that is focused on readability and simplicity, while JavaScript is a low-level, interpreted language that is focused on adding interactive features to web pages. Python has dynamic data types, while JavaScript has static data types. Python is more suitable for creating scripts and applications, while JavaScript is more suitable for creating interactive web pages.

Question: What if Isaac Newton had focused on biology instead of physics?

Layer 32 (Final): Isaac Newton had focused on biology instead of physics, he would have likely made similar discoveries in that field as he did in physics. His laws of motion and universal gravitation could have been formulated in a biological context, for example, describing the ways in which cells and organisms adhere to physical laws. He would have likely realized that biology is also an essential physical science, and that the laws of motion are just as universal in biology as in physics.

Dynamic Early Exiting: Isaac Newton had focused on biology instead of physics, he would have developed a different understanding of the natural world. He would have believed that biology was the foundation of all physics, and that it was the only way to understand the universe. He would have also believed that all living organisms had the same underlying laws of physics, rather than the laws of physics being created for humans to understand. He would have also believed that biology was the most important branch of science, and that it was the only way to understand the universe.

These examples show that the responses obtained via early exiting do not compromise the quality of the generation. Furthermore, responses have high syntactic and semantic overlap.



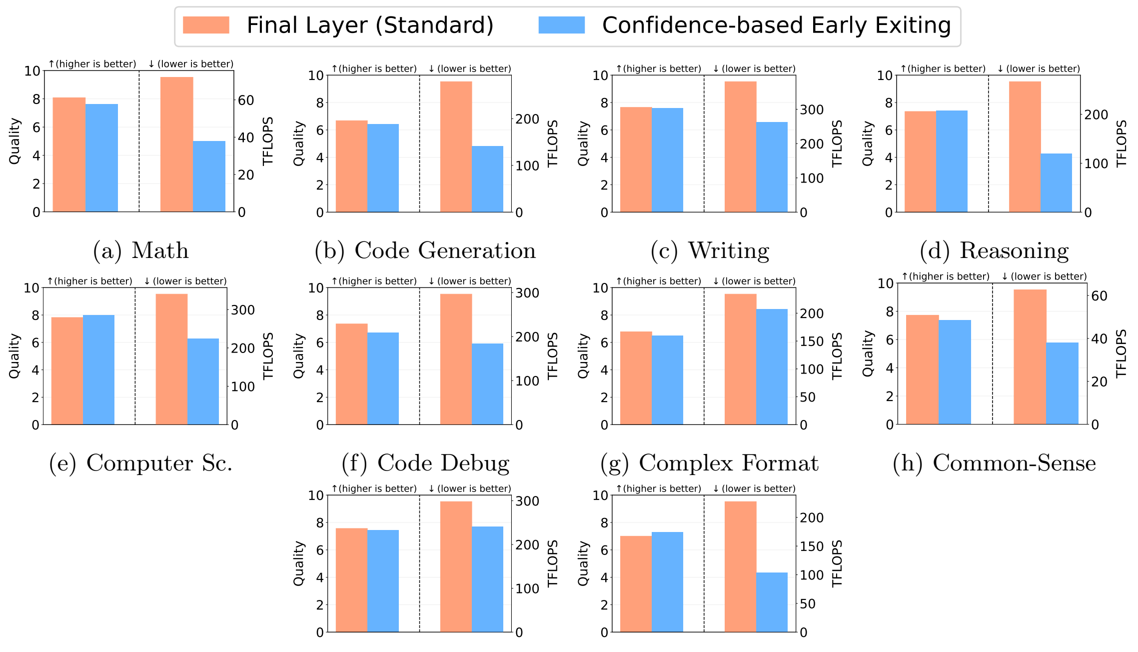
(a) Generic (b) Knowledge (c) Roleplay (d) Commonsense (e) Fermi (f) Counterfactual (g) Writing (h) Coding (i) Math

Figure 40. Comparing the quality of responses and the inference cost of the standard generation method with the dynamic early exiting method on different categories of the Vicuna Test set.

7.7.5 Quality and Inference Cost Analysis at Category Level

Vicuna and WizardLM test sets also provide the category corresponding to different test instances. To this end, we present category-level quality and inference cost results for these datasets.

Vicuna: Figure 40 compares the quality of responses and the inference cost of the standard generation method (final layer) with the dynamic early exiting method



(a) Math (b) Code Generation (c) Writing (d) Reasoning
(e) Computer Sc. (f) Code Debug (g) Complex Format (h) Common-Sense
(i) Counterfactual (j) Multilingual

Figure 41. Comparing the quality of responses and the inference cost of the standard generation method with the dynamic early exiting method on different categories of the WizardLM Test set.

for different categories of Vicuna test set. On average, it results in cost improvement of 33.39%. It can be observed that the approach consistently achieves efficiency improvement in all the categories which demonstrates the generality of the approach.

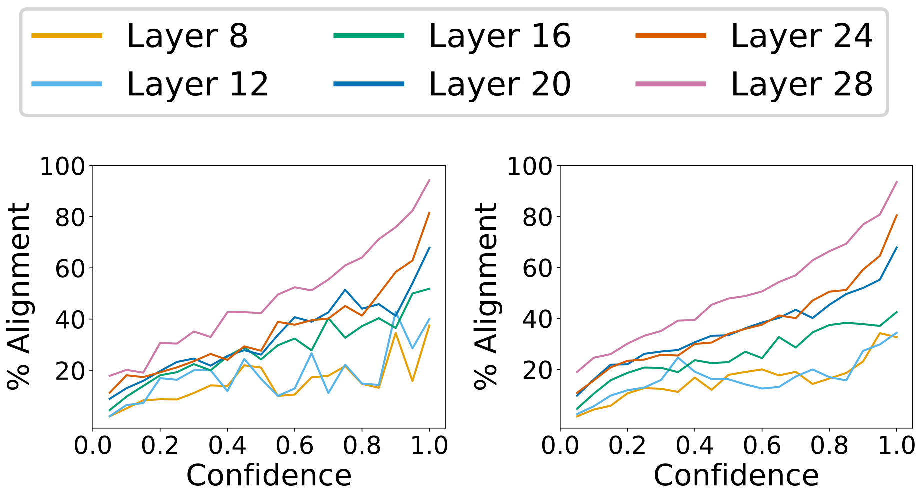
WizardLM: Figure 41 compares the quality of responses and the inference cost of the standard generation method (final layer) with the dynamic early exiting method for different categories of WizardLM test set. On average, it results in cost improvement of 36.12%.

7.7.6 Relationship Between Token Prediction Confidence and Percentage Alignment of the Intermediate Layers for the Model Tuned with Instruction Tuning (IT)

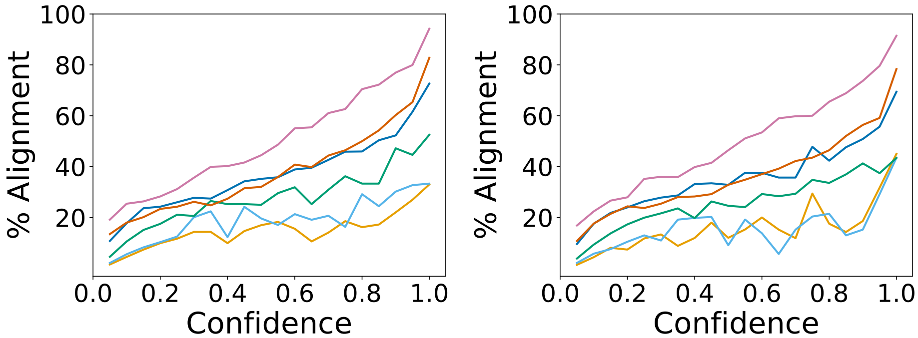
Figure 42 shows the relationship between the token prediction confidence of the intermediate layers and the percentage alignment with the token prediction of the final layer for standard instruction tuning (IT). It shows that the confidence is not well correlated with the percentage alignment. However, in IT with LITE (Figure 37), the intermediate layers' token prediction probabilities provide a strong signal of alignment.

7.7.7 Dynamic Confidence-Based Early Exiting with Aggressive Confidence Thresholds

We also experiment with aggressive confidence thresholds. Specifically, we use the following confidence thresholds: Layer 8: 0.85, Layer 12: 0.85, Layer 16: 0.8, Layer 20: 0.8, Layer 24: 0.7, and Layer 28: 0.6. These thresholds are lower than those used in the main paper. Figure 43 shows the quality and cost comparisons. It leads to larger cost improvements (of 49.92%) though it slightly drops the quality of generation (by 5.34%).

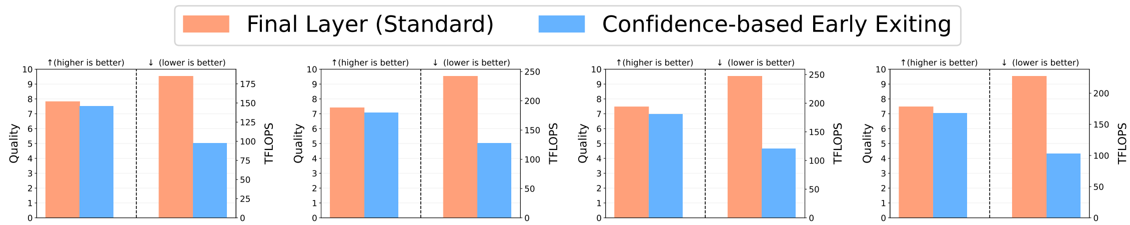


(a) Vicuna (b) Koala



(c) WizardLM (d) Self-Instruct

Figure 42. Demonstrating relationship between token prediction confidence of the intermediate layers and the percentage alignment with the token prediction of the final layer for model tuned with IT.



(a) Vicuna (b) Koala (c) WizardLM (d) Self-Instruct

Figure 43. Comparing the quality of responses (evaluated using the Claude model) and the inference cost (measured in FLOPs) of the standard generation method from the final layer with the dynamic early exiting method. Confidence Thresholds: Layer 8: 0.85, Layer 12: 0.85, Layer 16: 0.8, Layer 20: 0.8, Layer 24: 0.7, and Layer 28: 0.6. This aggressive configuration results in larger cost improvements of 49.93% but results in a slight degradation in the generation quality.

Test Dataset	**Cost Improvement (%)**
Vicuna	43.60 %
Koala	45.62 %
WizardLM	50.84 %
Self Instruct	45.35 %

Table 48. Percentage improvements in the inference cost (measured in FLOPs) with dynamic early exiting for the 13B model. On average, it achieves an improvement of 46.35%.

7.7.8 Results for 13B Model

For the 13B model we use the following confidence thresholds: Layer 8: 0.95, Layer 12: 0.95, Layer 16: 0.9, Layer 20: 0.9, Layer 24: 0.8, Layer 28: 0.7, Layer 32: 0.7, and Layer 36: 0.65,

Table 48 shows the cost improvements resulting from dynamic early exiting from the 13B model on each test dataset. On average, it results in 46.35% cost improvement. This improvement is higher than the improvement achieved in the case of the 7B model (37.86%).

Test Set	**# Samples**
Vicuna	80
Koala	180
WizardLM	218
Self Instruct	252

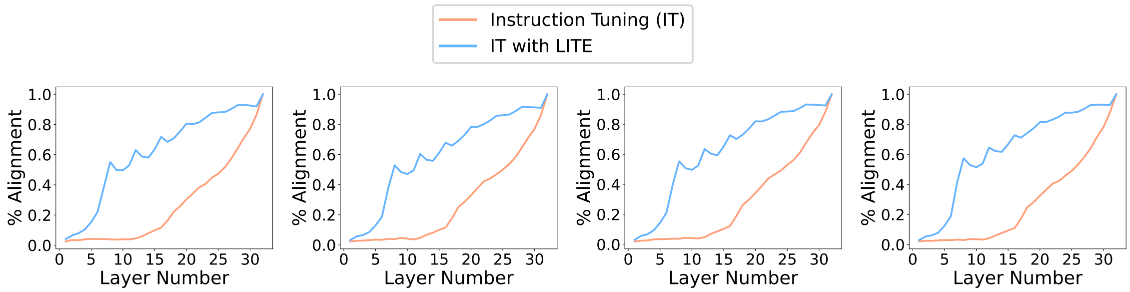
Table 49. Statistics of evaluation datasets: Vicuna, Koala, WizardLM, and Self-Instruct.

7.8 Weighted LITE

We also experiment using increasing weights for different intermediate layers during the loss aggregation. We use increasing weights as the later layers have more capacity to learn. Specifically, for the 7B model where we select layer numbers 8, 12, ..., 28, and 32, we use the following weights: 1, 2, ..., 7.

In Figure 44, we plot the percentage alignment of token predictions of all intermediate layers with the token predictions of the final layer. The figure shows the percentage alignment of (i) the model tuned using standard IT (orange) and (ii) the model tuned using IT with weighted `LITE` (blue). In Figure 45, we compare the alignment for the model tuned using IT with LITE and the model tuned using IT with weighted LITE. It can be observed that assigning lower weight to the initial layers results in just a slight reduction in alignment percentage.

We also plot the relationship between the token prediction confidence (softmax over the logits of the LM head) of the intermediate layers and the percentage alignment with the token prediction of the final layer. Figure 46 shows this plot for the model tuned with weighted `LITE`.



(a) Vicuna (b) Koala (c) WizardLM (d) Self-Instruct

Figure 44. Comparing percentage 'alignment' of intermediate layer token predictions with the token predictions of the final layer for the model tuned using IT (orange) and the model tuned using IT with weighted LITE (blue).

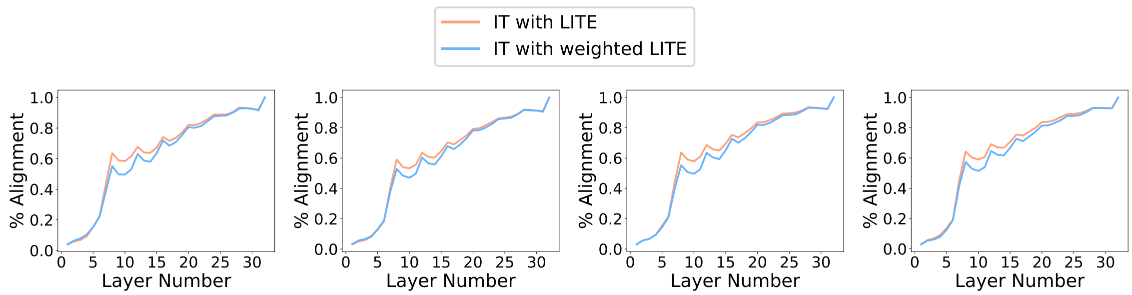
7.9 Design Decisions

FLOPs for measuring Computational Cost: We note that we use FLOPs as the metric of showcasing inference efficiency improvements because it is hardware independent, unlike latency.

KV Caching: We explore the dynamic exiting method for inference without KV caching. This is because the representations of the layers after the exiting layer are not computed in this method and thus will not be available in the cache for the next token prediction if the model exits from a higher layer than the previous token prediction.

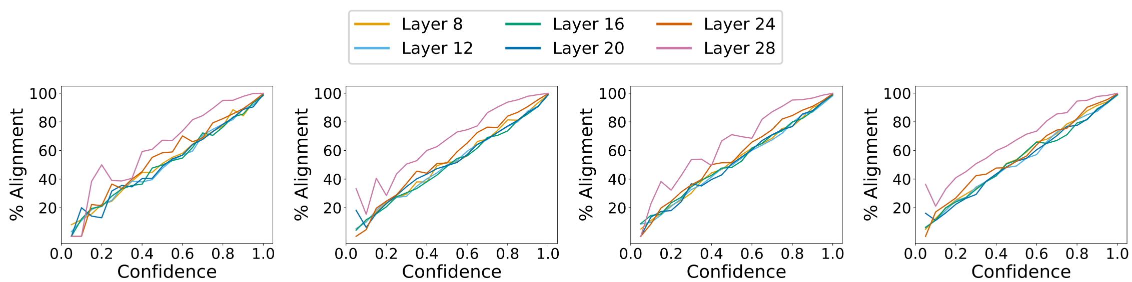
LLaMA-2 Models for Experiments: We experiment with LLaMA-2 models as they are publicly available and widely used for LLM research.

Evaluation Datasets To perform holistic evaluation, we experiment with four different human-instruction test sets including Vicuna, Self-Instruct, Koala, and WizardLM. We select these evaluation test sets as they can together cover a large



(a) Vicuna (b) Koala (c) WizardLM (d) Self-Instruct

Figure 45. Comparing percentage 'alignment' of intermediate layer token predictions with the token predictions of the final layer for the model tuned using IT with LITE (orange) and the model tuned using IT with weighted LITE (blue).



(a) Vicuna (b) Koala (c) WizardLM (d) Self-Instruct

Figure 46. Demonstrating trend of token prediction confidence of the intermediate layers and the % alignment with the final layer for model tuned with weighted `LITE`.

number and types of instructions thus resulting in a comprehensive evaluation. Table 49 shows the statistics of the datasets.

7.10 Discussion on Other Avenues of Research using Intermediate Layer Decoding

7.10.1 Speculative Sampling

In speculative sampling (Leviathan, Kalman, and Matias 2023; C. Chen et al. 2023; Ning et al. 2023; Kim et al. 2023; Spector and Re 2023), a short draft of K tokens is first generated from a smaller auto-regressive model and then the draft is scored using the target model. Using a rejection sampling scheme, a subset of the K draft tokens is accepted by sequentially checking from left to right, and thus in this process, the distribution of the target model is recovered for the accepted tokens. The efficiency in this technique comes from 'producing' more than one token (on average) from the target model in a single pass.

This technique requires an additional drafting model. However, we showed that instruction tuning with LITE enables the intermediate layers to acquire 'good' generation ability. Thus, the intermediate layer(s) of the same model can be used as the drafting model while the last layer remains to be the target model. This circumvents the requirement of maintaining a separate drafting model for speculative sampling.

7.10.2 Hallucination Detection

Addressing the hallucination problem of LLMs is an important research direction and a number of methods have been developed (Manakul, Liusie, and M. Gales 2023; Azaria and Mitchell 2023a; T. Zhang et al. 2023; Dhuliawala et al. 2023; Varshney, Yao, et al. 2023; Gou et al. 2023). Sampling based methods require generating multiple samples and then checking the information consistency between them. Enabling the intermediate layers with the generation ability equips us with multiple opportunities, such as checking the consistency based on the alignment percentage of the intermediate layers with the final layer or using the intermediate layers to generate the complete output and then checking the information consistency.

7.11 Conclusion and Discussion

In this chapter, we proposed instruction tuning with additional explicit losses from the intermediate layers and showed that it enables these layers to acquire 'good' generation ability without affecting the final layer's generation ability. We performed '*dynamic confidence-based early exiting*' at token level from the intermediate layers and showed that it improves the efficiency of inference while maintaining the generation quality. We further conducted a thorough analysis that resulted in several important findings. Overall, our work contributes to improving the efficiency of LLM inference while maintaining the generation quality, a crucial step en route to enabling their widespread adoption.

Looking forward, our work additionally opens up several other avenues for new research, such as speculative sampling from the intermediate layers to improve the inference efficiency and checking information consistency from the output of intermediate layers to detect hallucinations (discussed in Section 7.10). Furthermore, this approach is complementary to some existing efficiency methods, i.e., they can be used in conjunction to achieve even more efficiency gains.

Chapter 8

JOINTLY IMPROVING EFFICIENCY AND ACCURACY VIA MODEL CASCADING

Do all instances need inference through the big models for a correct prediction? Perhaps not; some instances are easy and can be answered correctly by even small capacity models. This provides opportunities for improving the computational efficiency of systems. In this chapter, we present an explorative study on 'model cascading', a simple technique that utilizes a collection of models of varying capacities to accurately yet efficiently output predictions. Through comprehensive experiments in multiple task settings that differ in the number of models available for cascading (K value), we show that cascading improves both the computational efficiency and the prediction accuracy. For instance, in K=3 setting, cascading saves up to 88.93% computation cost and consistently achieves superior prediction accuracy with an improvement of up to 2.18%. We also study the impact of introducing additional models in the cascade and show that it further increases the efficiency improvements.

8.1 Introduction

Pre-trained language models such as RoBERTa (Y. Liu et al. 2019a), ELECTRA (K. Clark et al. 2020), and T5 (Raffel et al. 2020b) have achieved remarkable performance on numerous natural language processing benchmarks (A. Wang et al. 2018;

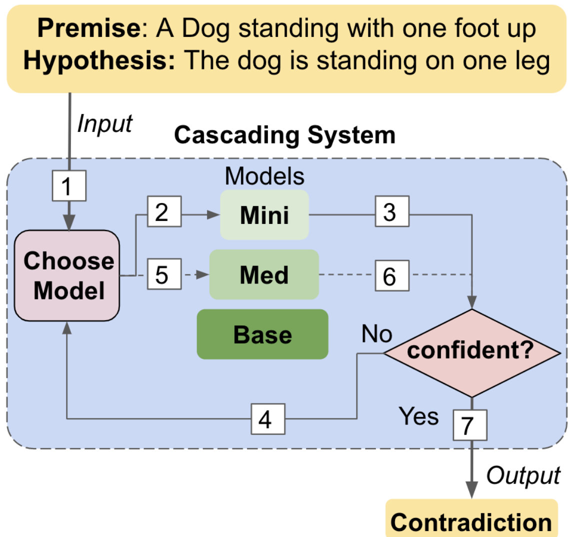


Figure 47. Illustrating a cascading approach with three models (Mini, Med, and Base) arranged in increasing order of capacity. An input is first passed through the smallest model (Mini) which fails to predict with sufficient confidence. Therefore, it is then inferred using a bigger model (Med) that satisfies the confidence constraints and the system outputs its prediction ('contradiction' as dog has four legs). Thus, by avoiding inference through large/expensive models, the system saves computation cost without sacrificing the accuracy.

A. Wang et al. 2019; Talmor et al. 2019). However, these models have a large number of parameters which makes them slow and computationally expensive; for instance, T5-11B requires ~87×10^{11} floating point operations (FLOPs) for an inference. This limits their widespread adoption in real-world applications that prefer computationally efficient systems in order to achieve low response times.

The above concern has recently received considerable attention from the NLP community leading to development of several techniques, such as (1) *network pruning* that progressively removes model weights from a big network (Wang, Wohlwend, and Lei 2020; Guo, Rush, and Kim 2021), (2) *early exiting* that allows multiple exit paths in a model (Xin et al. 2020), (3) *adaptive inference* that adjusts model size by adaptively selecting its width and depth (Goyal et al. 2020; Kim and Cho 2021), (4) *knowledge distillation* that transfers ‘dark-knowledge’ from a large teacher model to a shallow student model (Jiao et al. 2020; Z. Li et al. 2022), and (5) *input reduction* that eliminates less contributing tokens from the input text to speed up inference (Modarressi, Mohebbi, and Pilehvar 2022). These methods typically require architectural modifications, network manipulation, saliency quantification, or even complex training procedures. Moreover, computational efficiency in these methods often comes with a compromise on accuracy. In contrast, *model cascading*, a simple technique that utilizes a collection of models of varying capacities to **accurately yet efficiently** output predictions has remained underexplored.

In this work, we address the above limitation by first providing mathematical formulation of model cascading and then exploring several approaches to do it. In its problem setup, a collection of models of different capacities (and hence performances) are provided and the system needs to output its prediction by leveraging one or more models. On one extreme, the system can use only the smallest model and on the other extreme, it can use all the available models (ensembling). The former system would be highly efficient but usually poor in performance while the latter system would be fairly accurate but expensive in computation. Model cascading strives to get the best of both worlds by allowing the system to efficiently utilize the available models while achieving high prediction accuracy. This is in line with the ‘Efficiency NLP’ (Arase and al. 2021) policy document put up by the ACL community.

Consider the case of CommitmentBank (Marneffe, Simons, and Tonhauser 2019) dataset on which BERT-medium model having just 41.7M parameters achieves 75%

accuracy and a bigger model BERT-base having 110M parameters achieves 82% accuracy. From this, it is clear that the performance of the bigger model can be matched by inferring a large number of instances using the smaller model and only a few instances using the bigger model. Thus, by carefully deciding when to use bigger/more expensive models, the computational efficiency of NLP systems can be improved. *So, how should we decide which model(s) to use for a given test instance?* Figure 47 illustrates an approach to achieve this; it infers an instance sequentially through models (ordered in increasing order of capacity) and uses a threshold over the maximum softmax probability (*MaxProb*) to decide whether to output the prediction or pass it to the next model in sequence. The intuition behind this approach is that MaxProb shows a positive correlation with predictive correctness. Thus, instances that are predicted with high MaxProb get answered at early stages as their predictions are likely to be correct and the remaining ones get passed to the larger models. Hence, by avoiding inference through large and expensive models (primarily for easy instances), cascading makes the system computationally efficient while maintaining high prediction performance.

We describe several such cascading methods in Section 8.2.2. Furthermore, cascading allows custom computation costs as different number of models can be used for inference. We compute accuracies for a range of costs and plot an accuracy-cost curve. Then, we calculate its area (AUC) to quantify the efficacy of the cascading method. Larger the AUC value, the better the method is as it implies higher accuracy on average across computation costs.

We conduct comprehensive experiments with 10 diverse NLU datasets in multiple task settings that differ in the number of models available for cascading (K value from Section 8.2). We first demonstrate that cascading achieves considerable improvement in computational efficiency. For example, in case of QQP dataset, cascading system achieves 88.93% computation improvement over the largest model (M_3) in K=3 setting i.e. it requires just 11.07% of the computation cost of model M_3 to attain equal accuracy. Then, we show that cascading also achieves improvement in prediction accuracy. For example, on CB dataset, the cascading system achieves 2.18% accuracy improvement over M_3 in the K=3 setting. Similar improvements are observed in settings with different values of K. Lastly, we show that introducing additional model in the cascade further increases the efficiency benefits.

In summary, our contributions and findings are:

1. **Model Cascading:** We provide mathematical formulation of model cascading, explore several methods, and systematically study its benefits.
2. **Cascading Improves Efficiency**: Using accuracy-cost curves, we show that cascading systems require much lesser computation cost to attain accuracies equal to that of big models.
3. **Cascading Improves Accuracy:** We show that cascading systems consistently achieve superior prediction performance than even the largest model available in the task setting.
4. **Comparison of Cascading Methods:** We compare performance of our proposed cascading methods and find that DTU (8.2.2) outperforms all others by achieving the highest AUC of accuracy-cost curves on average.

We note that model cascading is trivially easy to implement, can be applied to a variety of problems, and can have good practical values.

8.2 Model Cascading

We define model cascading as follows:

Given a collection of models of varying capacities, the system needs to leverage one or more models in a computationally efficient way to output accurate predictions.

As previously mentioned, a system using only the smallest model would be highly efficient but poor in accuracy and a system using all the available models would be fairly accurate but expensive in computation. The goal of cascading is to achieve high prediction accuracy while efficiently leveraging the available models. The remainder of this section is organized as follows: we provide mathematical formulation of cascading in 8.2.1 and describe its various approaches in 8.2.2.

8.2.1 Formulation

Consider a collection of K trained models $(M_1, ..., M_K)$ ordered in increasing order of their computation cost i.e. for an instance x, $c_j^x < c_k^x$ $(\forall\ j < k)$ where c corresponds to the cost of inference. The system needs to output a prediction for each instance of the evaluation dataset D leveraging one or more models. Let M_j^x be a function that indicates whether model M_j is used by the system to make inference for the instance

x i.e.

$$M_j^x = \begin{cases} 1, & \text{if model } M_j \text{ is used for instance } x \\ 0, & \text{otherwise} \end{cases}$$

Thus, the average cost of the system for the entire evaluation dataset D is calculated as:

$$Cost_D = \frac{\sum_{x_i \in D} \sum_{j=1}^{K} M_j^{x_i} \times c_j^{x_i}}{|D|}$$

In addition to this cost, we also measure accuracy i.e. the percentage of correct predictions by the system. The goal is to achieve high prediction accuracy while being computationally efficient.

Performance Evaluation: With the increase in the computation cost, the accuracy usually also increases as the system leverages large models (that are often more accurate) for more number of instances. To quantify the performance of a cascading method, we first plot its accuracy-cost curve by varying the computation costs and then calculate the area under this curve (AUC). **Larger the AUC value, the better the cascading method is** as it implies higher accuracy on average across all computation costs. We note that the computation cost of the cascading system can be varied by adjusting the confidence thresholds of models in the cascade (described in the next subsection).

Along with the AUC metric, we evaluate efficacy of cascading on two additional parameters:

1. **Comparing computation cost of the cascading system at accuracies achieved by each individual model of cascade:** Consider a setting in which the model M_2 achieves accuracy a_2 at computation cost c_2; from the accuracy-cost

curve of the cascading system, we compare c_2 with the cost of the cascading system when its accuracy is a_2.

2. **Comparing the maximum accuracy of the cascading system with that of the largest model of collection:** We compare accuracy of the largest individual model with the maximum accuracy achieved by the cascading system.

Note that the first parameter corresponds to the point of intersection obtained by drawing a horizontal line from accuracy-cost point of each individual model on the accuracy-cost curve. Refer to the red dashed lines in Figure 48 and 50 for illustration. For a cascading system to perform better than the individual models in the cascade, it should have a lower computation cost (in parameter one) and a higher accuracy (in parameter two).

8.2.2 Approaches

We explore the following approaches of selecting which model(s) to use for inference.

Maximum Softmax Probability (MaxProb): Usually, the last layer of a model has a softmax activation function that distributes its prediction probability $P(y)$ over all possible answer candidates Y. MaxProb corresponds to the maximum softmax probability assigned by the model i.e.

$$MaxProb = \max_{y \in Y} P(y)$$

MaxProb (often termed as prediction confidence) has been shown to be positively correlated with predictive correctness (Hendrycks and Gimpel 2017; Hendrycks et al. 2020; Varshney, Mishra, and Baral 2022b) i.e. a high MaxProb value implies a high likelihood for the model's prediction to be correct. We leverage this characteristic of MaxProb in our first cascading approach. Specifically, we infer the given input instance sequentially through the models starting with M_1 and use a confidence threshold over MaxProb value to decide whether to output the prediction or pass the instance to the next model in sequence.

Consider an instance x for which the models till M_{z-1} fail to surpass their confidence thresholds and M_z exceeds its threshold then:

$$M_j^x = \begin{cases} 1, & \text{if } j \leq z \\ 0, & \text{if } j > z \end{cases}$$

The confidence thresholds could be different at different stages. Figure 47 illustrates this approach.

It provides efficiency benefits as it avoids passing easy instances (that can be potentially answered correctly by low-compute models) to the computationally expensive models. Furthermore, it does not sacrifice the accuracy of system because the difficult instances would often end up being answered by the large (and more accurate) models. We note that this approach requires additional computation for comparing MaxProb values with thresholds but its cost is negligible in comparison to the cost of model inferences and hence ignored in the overall cost calculation.

Distance To Uniform Distribution (DTU): In this approach, we use the distance between the model's softmax probability distribution and the uniform probability distribution as the confidence estimate (in place of MaxProb) to decide whether to output the prediction or pass the instance to the next model in sequence i.e.

$$DTU = \|P(Y) - U(Y)\|_2$$

where $U(Y)$ corresponds to the uniform output distribution. For example, in case of a task with 4 classification labels, $U(Y) = [0.25, 0.25, 0.25, 0.25]$. The intuition behind this approach is to leverage the entire shape of the output probability distribution and not just the highest probability as in MaxProb.

Random: In this approach, instead of using a metric such as MaxProb or DTU to decide which instances to pass to the next model in sequence, we do this instance selection process at random. This serves as a baseline cascading method.

Heuristic: Here, we use a heuristic derived from the input text to decide which instances to pass to the next model in sequence. Specifically, we use length of the input text as the heuristic.

Routing: In this approach, instead of sequentially passing an instance to bigger and bigger models, we skip intermediate models and pass the instance directly to a suitable model based on its maxProb value. For example, in $K = 3$ setting, we first infer using M_1 and if its maxProb is very low then we skip M_2 and directly pass it to M_3. On the other hand, if its maxProb is sufficiently high (but below M_1's output threshold) then we pass it to M_2. The intuition behind this approach is that the system might save inference cost of intermediate models by directly using a suitable model that is likely to answer it correctly. This approach is not applicable for $K = 2$ as there is only one option to route after inference through the model M_1.

Our work is different from existing methods in the following aspects: (1) Existing methods typically require architectural changes, network manipulation, saliency quantification, knowledge distillation, or complex training procedures. In contrast, cascading is a simple technique that is easy to implement and does not require such modifications, (2) The computational efficiency in existing methods often comes with a compromise on accuracy. Contrary to this, we show that model cascading surpasses the accuracy of even the largest models, (3) Existing methods typically require training a separate model for each computation budget; on the other hand, a single cascading system can be adjusted to meet all the computation constraints. (4) Finally, cascading does not require an instance to be passed sequentially through the model layers; approaches such as *routing* (section 8.2) allow passing it directly to a suitable model.

8.3 Experiments

8.3.1 Experimental Details

Datasets: We experiment with a diverse set of NLU classification datasets: SNLI (Bowman et al. 2015), Multi-NLI (Williams, Nangia, and Bowman 2018), Dialogue-NLI (Welleck et al. 2019), Question-NLI (A. Wang et al. 2018), QQP (Iyer, Dandekar, and Csernai 2017), MRPC (Dolan and Brockett 2005), PAWS (Zhang, Baldridge, and He 2019), SST-2 (Socher et al. 2013), COLA (Warstadt, Singh, and Bowman 2019), and CommitmentBank (Marneffe, Simons, and Tonhauser 2019).

Models: We use the following variants of BERT (Devlin et al. 2019): BERT-mini (11.3M parameters), BERT-medium (41.7M parameters), BERT-base (110M

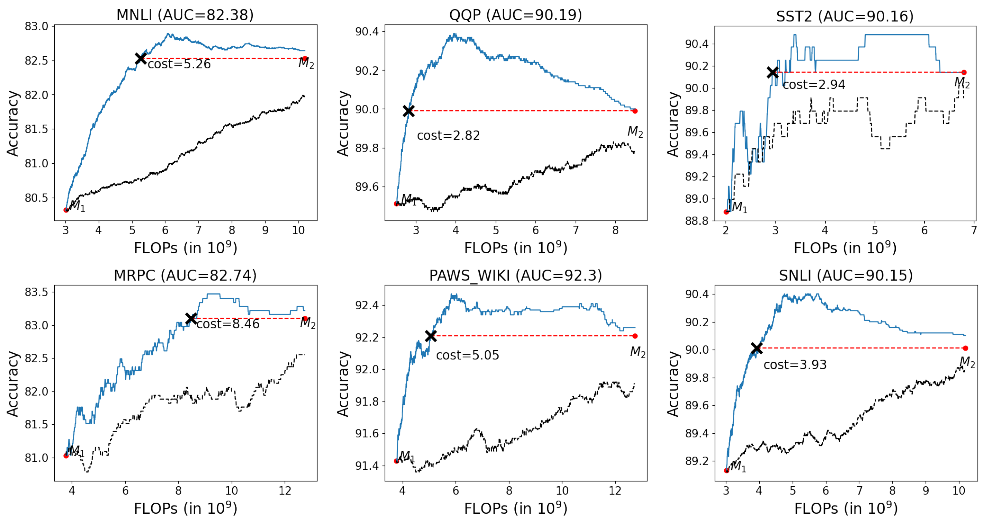


Figure 48. Accuracy-computation cost curves for cascading with *MaxProb* (in blue) and *Random* baseline (in black) methods in K=2 setting. Red points correspond to the accuracy-cost values of individual models M_1 and M_2. Points of intersection of red dashed lines drawn from M_2 on the blue curve correspond to the evaluation parameters described in Section 8.2. *MaxProb* outperforms *Random* baseline as it achieves considerably higher AUC.

parameters), and BERT-large (340M parameters) for our experiments. Table 50 shows the computation cost (in FLOPs) of these models for different input text sequence lengths. We use sequence length of 50 for COLA, 80 for SST2, 100 for QQP, 120 for MNLI, DNLI, SNLI, 150 for QNLI, MRPC, PAWS, and 275 for CommitmentBank datasets following the standard experimental practice. We run all our experiments on Nvidia V100 GPUs with a batch size of 32 and learning rate ranging in $\{1–5\}e–5$.

In the following subsections, we study the effect of cascading in multiple settings that differ in the number of models in the cascade i.e. K value in the task formulation.

Length	Mini (128M)	Medium (474M)	Base (1.3G)	Large (3.8G)
50	0.16	1.26	4.25	5.10
80	0.25	2.01	6.80	24.16
100	0.31	2.52	8.49	30.20
120	0.38	3.02	10.19	36.24
150	0.47	3.78	12.74	45.30
220	0.69	5.54	18.69	66.44
275	0.87	6.92	23.36	83.05

Table 50. Inference cost (in 10^9 FLOPs) of BERT variants for different input text sequence lengths. We also specify the storage size of the models in this table.

Method	MNLI	QNLI	QQP	SST2	COLA	CB	DNLI	MRPC	PAWS	SNLI
Random	81.16	84.24	89.64	89.64	77.97	77.27	85.35	81.76	91.63	89.50
Heuristic	81.13	84.15	89.69	89.06	79.45	76.79	85.30	81.54	91.65	89.45
MaxProb	82.38	85.73	90.19	90.16	80.25	80.40	85.27	82.74	92.30	90.15
DTU	82.38	85.73	90.18	90.16	80.25	80.70	85.24	82.74	92.30	90.16

Table 51. Comparing AUC values of different cascading methods in K=2 setting. *Random* and *Heuristic* correspond to the cascading baselines. *MaxProb* and *DTU* outperform both the baselines.

8.3.2 Cascading with Two Models (K=2)

8.3.2.1 Problem Setup

In this setting, we consider two trained models BERT-medium (41.7M parameters) as M_1 and BERT-base (110M parameters) as M_2. We also analyze results for other model combinations (such as medium, large and mini, large).

8.3.2.2 Results

Recall that the computation cost of a cascading system can be controlled by changing the M_j values. For example, in case of MaxProb, changing the confidence threshold value would result in different M_j values and hence different cost and accuracy values. Figure 48 shows accuracy-cost curves for two cascading approaches: MaxProb (in blue) and Random Baseline (in black). In the same figure, we also show accuracy-cost points for the individual models M_1 and M_2. However, to compare the performance of these methods, we provide their AUC values (of their respective accuracy-cost curves) in Table 51.

Efficiency Improvement: The accuracy-cost curves show that the cascading system matches the accuracy of the larger model M_2 at considerably lesser computation cost. This cost value corresponds to the point of intersection on the curve with a straight horizontal line drawn from M_2 (red dashed line). For example, in case of QQP, model M_2 achieves 89.99% accuracy at average computation cost of 8.49×10^9 FLOPs while the cascading system achieves the same accuracy at only 2.82×10^9 FLOPs. Similarly, in case of MNLI, M_2 achieves 82.53% accuracy at cost of 10.19×10^9 FLOPs while the cascading system achieves the same accuracy at only 5.26×10^9 FLOPs. Such improvements are observed for all datasets. This efficiency benefit comes from using the smaller models for a large number of instances and passing only a few instances to the larger models.

Accuracy Improvement: From the accuracy-cost curves, it can be observed that beyond the cost value identified in the previous paragraph (where the red dashed line intersects the accuracy-cost curve), the cascading system outperforms model M_2 in terms of accuracy. For example, in case of QQP, cascading with MaxProb achieves accuracy of up to 90.39% that is higher than the accuracy of M_2 (89.99%). Similar improvements are observed for all other datasets. We note that the accuracy improvement is a by-product of cascading, its primary benefit remains to be the improvement in computational efficiency.

Higher accuracy achieved by the cascading system (that uses M_1 for some instances and conditionally also uses M_2 for others) than the larger model M_2 implies that M_1, despite being smaller in size is more accurate than M_2 on at least a few instances. Though, on average across all instances, M_2 has higher accuracy than M_1. The cascading system uses M_1 for instances on which it is sufficiently confident and thus more likely to be correct. Only the instances on which it is not sufficiently confident get passed to the bigger model. This supports the findings of recent works such as (Zhong et al. 2021; Varshney, Mishra, and Baral 2022a) that conduct instance-level analysis of models' predictions. We further analyze these results in the next paragraphs.

Comparing Cascading Approaches: Figure 48 demonstrates that MaxProb cascading approach clearly outperforms the 'Random' cascading baseline. In Table 48, we compare AUC of respective accuracy-cost curves of various cascading approaches. Both MaxProb and DTU outperform both the baseline methods (Random and Heuristic). In $K = 2$ setting, both MaxProb and DTU achieve roughly the same performance on average across all datasets. The gap between MaxProb and DTU becomes more significant in $K = 3$ setting (8.3.3).

Contribution of M_1 and M_2 in the Cascade: To further analyze the performance of the cascading system, we study the contribution of individual models M_1 and M_2 in the cascade. Figure 49 shows the contribution of M_1 and M_2 for MNLI dataset when the cost is 5.26×10^9 FLOPs i.e. the point at which the accuracy of the cascading system is equal to that of the bigger model M_2 (intersection point of the horizontal red dashed line with the accuracy-cost curve of the cascading system in Fig 48). At this point, the cascade system uses M_1 for 78% instances and M_2 for the remaining 22% instances. The accuracy of M_1 on its 78% instances (87.6%) would be equal to that of M_2 on those 78% instances as the overall accuracy of system on complete dataset (100% instances) is equal to that of M_2. However, this does not imply that the instance-level predictions of the two models on those 78% would be exactly the same. Though, their predictions overlap in majority of the cases.

Figure 49 also shows that the accuracy of model M_1 on the instances that got passed to M_2 in the cascade system is significantly lesser (by 33.12%) than on the instances that M_1 answered (blue bars). M_2 achieves 10.12% higher accuracy on those instances than M_1. Therefore, the cascading system utilizes the models efficiently by using the smaller model M_1 for the easy instances and the larger model M_2 for the difficult ones.

8.3.3 Cascading with Three Models (K=3)

8.3.3.1 Problem Setup

Now, we study the effect of introducing another model in the problem setup of K=2 setting. Specifically, we consider three models: BERT-mini (11.3M parameters)

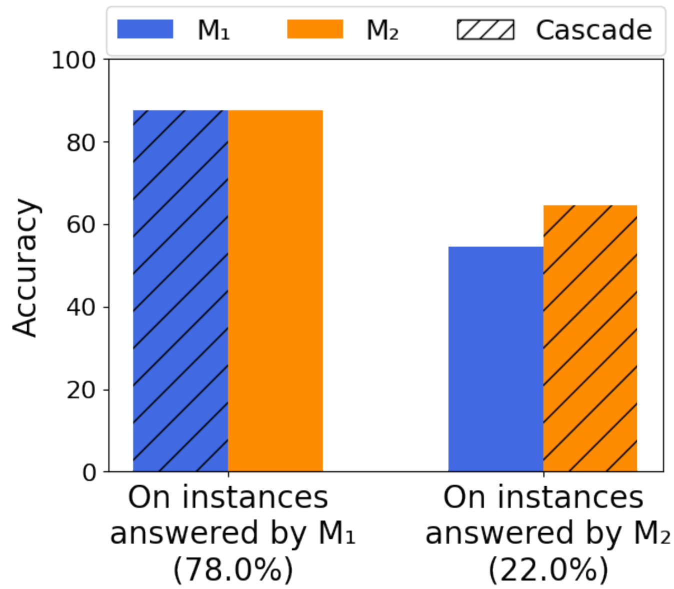


Figure 49. Comparing accuracy of individual models M_1 and M_2 on the instances answered by each model when used as cascade for MNLI dataset in K=2 setting.

Method	MNLI	QNLI	QQP	SST2	COLA	CB	DNLI	MRPC	PAWS	SNLI
Random	78.77	80.58	88.97	87.00	76.55	77.28	84.49	78.30	87.74	88.12
Heuristic	78.85	80.44	88.87	87.67	76.28	77.11	84.46	77.59	88.28	88.27
MaxProb	80.89	82.97	90.1	89.45	78.66	78.31	85.17	80.2	90.23	89.67
DTU	**80.98**	**83.28**	**90.15**	**89.6**	**78.87**	**78.52**	**85.20**	80.42	90.46	**89.72**
Routing	80.55	82.93	89.92	89.52	78.60	74.58	85.20	**80.97**	**90.68**	89.46

Table 52. Comparing AUC values of different cascading methods in K=3 setting. *Random* and *Heuristic* correspond to the cascading baselines. *DTU* outperforms other cascading methods on average.

as M_1, BERT-medium (41.7M parameters) as M_2, and BERT-base (110M parameters) as M_3 in this setting. **Note that BERT-medium is referred to as M_2 in this setting as it is the second model in cascading setup unlike the $K = 2$ setting (8.3.2) in which it was M_1.**

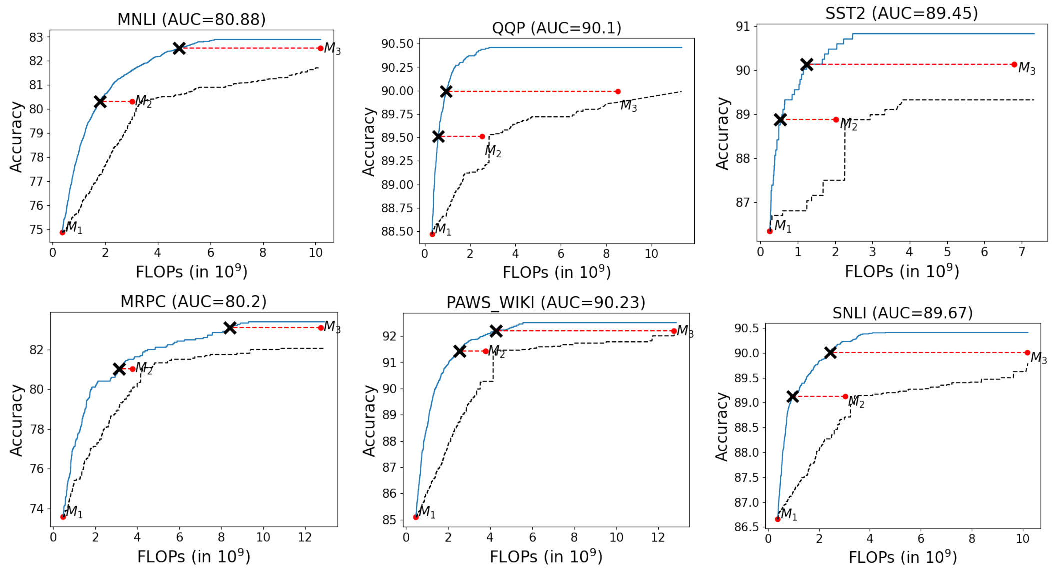


Figure 50. Accuracy-computation cost curves for cascading with MaxProb (in blue) and Random baseline (in black) methods in K=3 setting. Accuracy-cost values of individual models M_1, M_2, and M_3 are shown in red. Note that M_1 here is different from M_1 in Figure 48. *MaxProb* outperforms *Random* baseline as it achieves higher AUC.

8.3.3.2 Results and Analysis

Figure 50 shows the accuracy-cost curves of two cascading approaches: MaxProb (in blue) and Random Baseline (in black) and Table 52 compares AUC values achieved by various cascading approaches. In general, cascading achieves larger improvement (in magnitude) in K=3 setting than K=2 setting.

Efficiency Improvement: The accuracy-cost curves show that the cascading system matches the accuracy of larger models M_2 and M_3 at considerably lesser respective computation costs. For example, in case of QQP, cascading system matches the accuracy of model M_3 by using just 11.07% of M_3's computation cost and of

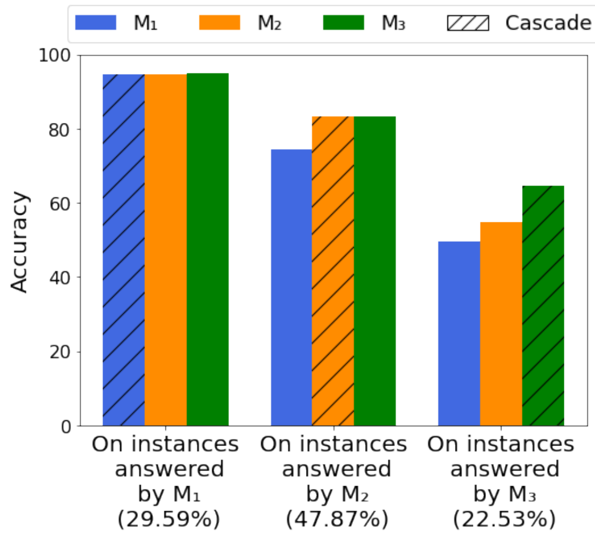


Figure 51. Comparing accuracy of individual models M_1, M_2, and M_3 on the instances answered by each model when used in the cascade for MNLI dataset.

model M_2 by using just 23.53% of M_2's computation cost. The magnitude of efficiency improvement in this setting is higher than that in the K=2 setting.

Accuracy Improvement: Cascading also achieves improvement in the overall accuracy. For example, on the CB dataset, cascading system achieves 83.93% accuracy that is even higher than the largest model M_3. Similar improvements are observed for other datasets also.

Comparing Cascading Approaches: Table 52 compares AUC values achieved by various cascading approaches. DTU clearly outperforms all other cascading methods as it achieves the highest AUC values. We attribute this to DTU's characteristic of utilizing the entire shape of the output probability distribution and not just the highest probability in computing its confidence.

Contribution of M_1, M_2, and M_3 in Cascade:

Figure 51 shows the contribution of individual models M_1, M_2, and M_3 in the cascade for MNLI dataset when the cost is 4.8×10^9 FLOPs i.e. the point at which accuracy of cascade is equal to that of the largest model M_3 (where the horizontal red dashed line drawn from M_3 intersects the accuracy-cost curve in Fig 50). The figure shows that the accuracy of M_1 on the instances that were passed to M_2 drops by 20.04% and accuracy of M_2 on instances that were passed to M_3 drops by 28.53%. This shows that the cascading system is good at identifying potentially incorrect predictions of M_1 and passes those instances to M_2 and similarly good at identifying potentially incorrect predictions of M_2 and passes those instances to M_3 .

Advantage of introducing another model in the Cascade: Comparing figure 51 for the K=3 setting with the figure 49 for K=2 setting, we find that by introducing a smaller model in the collection, the cascading system can be made more efficient. This is because the BERT-medium model answered 78% instances in K=2 setting and that portion got split into BERT-mini (smaller cost than medium) and medium models in K=3 setting while maintaining the accuracy. This suggests that the cascading technique utilizes the available models efficiently without sacrificing the accuracy.

8.3.4 Analysis with Other Model Combinations

8.3.4.1 Medium and Large

Figure 52 shows accuracy-cost curves with MaxProb (in blue) and Random (in black) as cascading approaches with M_1 as BERT-medium and M_2 as BERT-large.

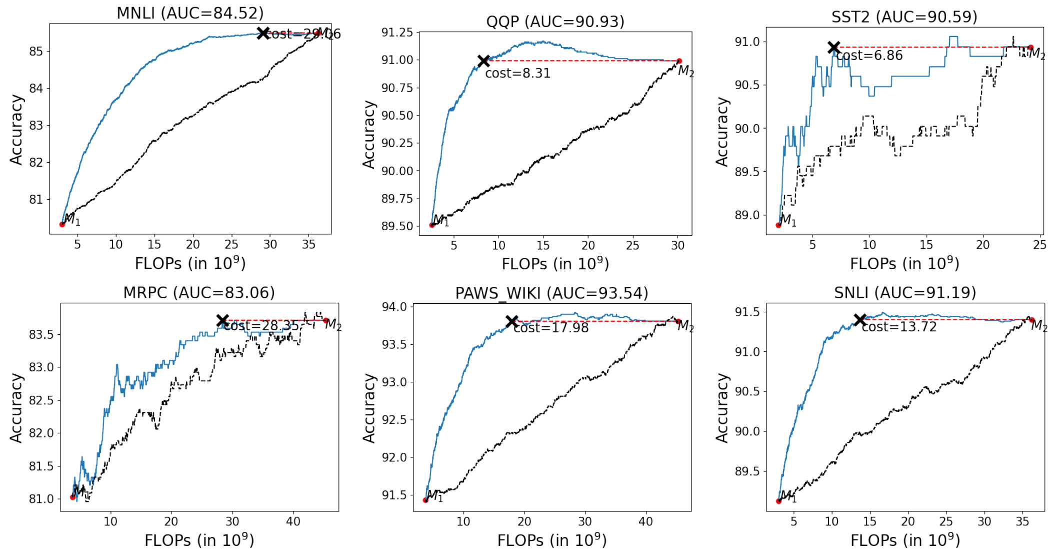


Figure 52. Accuracy-Cost curves for K=2 setting with M_1 as BERT-medium and M_2 as BERT-large models.

MaxProb approach clearly outperforms Random approach and achieves considerably higher AUC value.

8.3.4.2 Mini and Large

Figure 53 shows accuracy-cost curves with MaxProb (in blue) and Random (in black) as cascading approaches with M_1 as BERT-mini and M_2 as BERT-large. MaxProb approach clearly outperforms Random approach and achieves considerably higher AUC value.

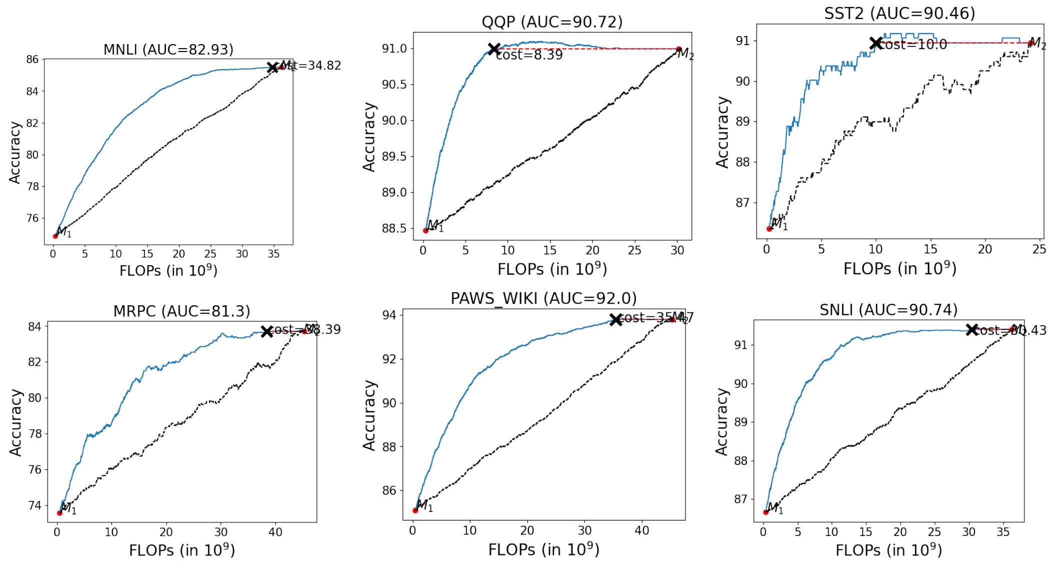


Figure 53. Accuracy-Cost curves for K=2 setting with M_1 as BERT-mini and M_2 as BERT-large models.

8.4 Conclusion and Discussion

We systematically explored model cascading and proposed several methods for it. Through comprehensive experiments with 10 diverse NLU datasets, we demonstrated that cascading improves both the computational efficiency and the prediction accuracy. We also studied the impact of introducing another model in the collection and showed that it further improves the computational efficiency of the cascading system.

Selecting Optimal Operating Threshold: The selection of confidence threshold for models in the cascade is dependent on the computation budget of the system. A low-budget system can select low threshold for the low-cost models (so that low-cost models answer majority of the questions leading to less computation cost) and similarly, high-budget systems can afford to select high thresholds to achieve higher accuracy. In order to select thresholds in an application-independent manner, the ML's standard practice of using the validation data to tune the hyperparameters can be used.

Outlier/OOD Detection Techniques: Outlier/OOD detection techniques such as (Kimin Lee et al. 2018; Hsu et al. 2020; W. Liu et al. 2020) can also be explored to decide which instance to pass to the bigger models in the cascade.

Including Linear Models in the Cascade: This idea can be extended to include non-transformer based less expensive models like linear models or LSTM based models. Since the computation cost of these models is significantly lower than the transformer based models and yet they achieve non-trivial predictive performance, a cascading system with these models could achieve even more improvement in computational efficiency. We plan to explore this aspect in the future work.

Chapter 9

EFFICIENTLY UTILIZE EXTERNAL KNOWLEDGE FOR OPEN-DOMAIN QUESTION ANSWERING VIA DYNAMIC READING

State-of-the-art open-domain QA models typically use a retriever–reader approach in which the retriever finds relevant passages and the reader leverages that to predict the answer. Prior work has shown that the reader's performance usually tends to improve with the increase in the number of these passages. Thus, state-of-the-art readers typically use a large number of passages (e.g. 100) for inference. While such systems achieve high prediction performance, their inference is computationally very expensive. We humans, on the other hand, have a remarkable capability of utilizing external knowledge **efficiently** while answering. Motivated by this, we ask a question "*Can the open-domain **QA reader** utilize external knowledge efficiently like humans without sacrificing the prediction performance?*" To this end, we investigate an approach that utilizes both the 'closed-book' (parametric knowledge) and the 'open-book' (external knowledge) inferences in an efficient manner. Furthermore, instead of using a large fixed number of passages for open-book inference, we dynamically read the external knowledge in multiple 'knowledge iterations'. Through comprehensive experiments on NQ and TriviaQA datasets, we demonstrate that this approach improves both the **inference efficiency** and the **prediction accuracy** of the reader.

9.1 Introduction

Retriever-reader systems (Danqi Chen et al. 2017; Karpukhin et al. 2020; Khattab and Zaharia 2020; Izacard and Grave 2021) have achieved impressive performance on the open-domain QA task. In this pipeline, the retriever finds top-N relevant passages and the reader leverages them to predict the answer. Prior work has shown that the reader's performance tends to improve (up to a certain extent) with the increase in the value of N. Thus, state-of-the-art models use a large number of passages (e.g. 100). While this strategy results in a high prediction performance, it makes the inference of the reader computationally very expensive. For instance, Fusion-in-Decoder reader model (FiD) (Izacard and Grave 2021) requires approximately 70×10^{11} floating-point operations (FLOPs) for an inference with 100 passages. This high inference cost limits the widespread adoption of such systems in real-world applications that prefer efficient systems to be able to achieve low response times.

Improving the efficiency of systems has been an important research topic in NLP. For the open-domain QA (ODQA) task, efficiency from the perspectives of retrieval (Zhao, Lu, and Lee 2021) and on-disk memory (Min et al. 2021; Izacard et al. 2020; Yamada, Asai, and Hajishirzi 2021) has been studied. However, the aspect of efficiently leveraging external knowledge to improve the computation performance of the **reader model** has remained underexplored.

Humans, notably have a remarkable capability of utilizing external knowledge: firstly, if we can answer the question using our already acquired knowledge then we do not even use the external knowledge, and in the case when we do require it, we don't always need to read the entire knowledge; we only read the amount that is sufficient to find the answer. Motivated by this strategy, we argue that some questions are trivial and can be answered with a few passages or even without using any external knowledge at all (by just relying on the knowledge already stored in the model parameters). Consider the case of FiD model, it achieves 54.43% and 50.61% exact match accuracies when used with 100 and 10 passages respectively. From this, it is clear that the performance of the system utilizing 100 passages can be matched by inferring a large number of instances with just 10 passages and only a few instances with 100 passages. Furthermore, the closed-book model (Roberts, Raffel, and Shazeer 2020) that does not use any external knowledge and relies only on the parametric knowledge (acquired during pre-training/fine-tuning) requires considerably lesser FLOPs and achieves lower yet non-trivial accuracy of 29.83%. **Thus, by carefully deciding when external knowledge is required and whether the current amount of knowledge is sufficient to answer a question correctly, the computational efficiency of the reader system can be considerably improved while maintaining the high prediction accuracy.** Moreover, this can also help the system mitigate the distraction that may result from using too many passages for inference and thus can even improve its prediction accuracy.

Following the above intuition, we explore an approach that utilizes both 'closed-book' and 'open-book' inferences and dynamically uses external knowledge in multiple *knowledge iterations*. Specifically, given a question, we first answer it using the low-cost closed-book model that relies on the parametric knowledge. If its prediction confidence is sufficiently high then the prediction is outputted otherwise we use the open-book model with external knowledge. Unlike the standard open-book models that always 'read' a fixed number of passages, in our method, the knowledge provided to the reader is iteratively increased until the model's prediction becomes sufficiently confident. We study four ways to measure the confidence of prediction of the models (Section 9.2) and demonstrate that the confidence shows a positive correlation with the predictive correctness. Thus, instances that are predicted with high confidence using low-cost inference get answered at early stages as their predictions are likely to be correct, and the remaining instances get answered with dynamically used external knowledge. Hence, by avoiding expensive inference primarily for easy instances and dynamically using external knowledge, our approach makes the reader computationally efficient while maintaining high prediction accuracy.

Through comprehensive experiments on NQ and TriviaQA datasets, we first show that our approach considerably improves the computation efficiency of inference of the reader. Comparing with the FiD reader, we show that our approach matches its accuracy by utilizing just 18.32% of its inference computation cost (in FLOPs). Then, we show that our approach also leads to a consistent improvement in prediction accuracy. Specifically, it outperforms FiD by achieving up to 55.10% accuracy on NQ Open and 72.33% on TriviaQA. This improvement is an outcome of mitigating distraction that results from using too many passages for inference. Finally, we note that our approach is intuitive, easy to implement, and also has practical values.

9.2 Approach

Firstly, we note that even a **closed-book** reader (CB) that does not use any external knowledge achieves a non-trivial accuracy by just relying on the knowledge already stored in its network parameters (acquired during pre-training/fine-tuning). This is a low-cost inference as the input contains just the question (without any additional context). So, in our approach, we first infer the given question using the closed-book reader and output the prediction if it is already sufficiently confident. If it is not confident then we leverage the external knowledge with the **open-book** reader (OB). Unlike the standard open-book readers that use all the top retrieved passages for inference, we iteratively increase the number of passages until the reader predicts with sufficient confidence; we refer to these iterations as '**knowledge iterations**'.

This conditional multi-stage inference process achieves computation efficiency benefits for two reasons: first, if CB reader is already sufficiently confident in its prediction then the expensive open-book inference is not used at all (this corresponds to the case of the least inference cost) and second, when OB reader is indeed used for inference, it leverages the external knowledge efficiently by reading just enough passages required to predict confidently instead of reading a static large number of passages. This approach can also help the system mitigate the distraction that may result from using too many passages for inference.

Hence, by avoiding expensive inference and dynamically using the external knowledge, our approach makes the reader inference computationally efficient while maintaining high prediction accuracy. We note that this **doesn't impact the retriever**

as the retrieval is done only once irrespective of the number of knowledge iterations and the number of passages used in each individual iteration. We further note that **in this work, our focus is only on improving the cost of reader inference**. We detail our approach and provide its mathematical formulation in the next subsection.

9.2.1 Mathematical Formulation

Let q be the given question, K be the number of knowledge iterations, and $M^q_{OB_k}$ be the value indicating whether k^{th} iteration ($k \leq K$) with the OB reader is used for inference.

$$M^q_{OB_k} = \begin{cases} 1, & \text{if } k^{th} \text{ knowledge iteration with} \\ & \text{OB reader is used for question } q \\ 0, & \text{otherwise} \end{cases}$$

Sample Scenario: Consider a scenario in which top-100 passages are available and in our approach we are using two knowledge iterations ($K = 2$) using 20 and 100 passages respectively i.e. a question will be first inferred using the CB reader, if it is not sufficiently confident then top-20 (referred as S_1) passages will be used with the OB model (referred as OB_1) and if that prediction is not confident then top-100 (S_2) passages will be used with the OB model (OB_2). In the computationally most efficient case, inference would be made only using the CB reader and in the most expensive case, inference would be made sequentially using CB, then OB_1 (OB using 20 passages), and then OB_2 (OB using 100 passages). In this work, we comprehensively study extensive combinations of CB, OB readers and values of K and S_k.

Cost of Reader Inference: In our formulation, the cost of reader inference for instance q is:

$$Cost^q = C_{CB} + \sum_{k=1}^{K} (M^q_{OB_k} \times S_k \times C_{OB})$$

where C_{CB} and C_{OB} are the inference costs of CB and OB models respectively, and S_k is the number of passages used by the OB model in k^{th} knowledge iteration. This is because the low-cost CB reader is first used for all the instances and then OB model is conditionally used in different knowledge iterations. In the sample scenario, the minimum cost would be C_{CB} and the maximum cost would be $C_{CB} + 20 \times C_{OB} + 100 \times C_{OB}$. Next, we discuss a few important characteristics of this formulation.

Inference Cost of OB Reader: FiD (Izacard and Grave 2021) is one of the top performing open-book models. It uses an encoder-decoder architecture i.e. it first computes the representation of question + passage for each passage independently using the encoder (with fixed number of input tokens) and then concatenates these representations and passes it to the decoder for answer prediction. Thus, to compute the OB model's cost of inference in the kth iteration, we multiply C_{OB} with S_k where C_{OB} is the cost of single inference with that fixed number of tokens and it is done S_k times.

However, we note that this is the upper bound of the inference cost because the encoder representation of the passages of the previous iterations can be reused i.e. the representations of passages of $(k-1)^{th}$ iteration can be reused in the kth iteration. However, this would require auxiliary space for storage and would involve a trade-off between computation cost and storage space. We leave the investigation for future work but **note that the inference cost of our system would be even lower in practice**. Moreover, as we demonstrate through extensive experiments that even with this cost upper bound, the proposed method achieves very high improvements in computation efficiency.

Same Model for CB and OB: We note that the same model can also be used to act as CB when external knowledge is not available and as OB when it is available. However, their cost of inference will still be different as it depends on the number of input tokens used for inference. The CB reader uses only the question as input while the OB reader also uses external knowledge (thus more tokens). Therefore, in our general formulation, we keep two different variables for their respective costs.

Total Cost of Reader Inference: The average cost of reader inference for the dataset D is:

$$Cost_D = \frac{\sum_{q_j \in D} Cost^{q_j}}{|D|}$$

9.2.2 Deciding When to Use More Knowledge

The leading models in ODQA and perhaps in many NLU tasks are mostly seq2seq generative models; for e.g. both closed-book and FiD models are based on T5 (Raffel et al. 2020b). Therefore, a critical design decision in our method pertains to computing the prediction confidence for these models. These models make predictions token by token and output a probability distribution over the entire vocabulary for each token. We explore a number of methods to compute the confidence scores.

Let the maximum softmax probabilities for the prediction having n tokens at each token position be $(p_1, p_2, p_3, ..., p_n)$. We explore the following ways of computing the model's confidence.

Product of probabilities of all tokens (P_{PA}): In this technique, we take the product of probabilities of all prediction tokens i.e. $PROD(p_1, p_2, ..., p_n)$ as the model's prediction confidence. This is the standard technique used in various tasks such as perplexity computation. We also experiment with several other confidence techniques.

Probability of the first token (P_F): Here, we simply use the probability of the first prediction token i.e. p_1 as the model's prediction confidence.

Average probability of first and last token (P_{FL}): Here, we use the average probability of the first and the last token i.e. $AVG(p_1, p_n)$ as the confidence.

Average probability across all tokens (P_A): In this technique, we utilize probabilities of all the tokens of the prediction and take the average probability across all tokens i.e. $AVG(p_1, p_2, ..., p_n)$ as model's prediction confidence.

9.2.3 Baseline Approaches

For a fair comparison of our approach (and various confidence computation methods), we also compare them with several other simple baselines:

Random: In this method, instead of using a metric based on probabilities to decide which instances to pass to the OB model/next knowledge iteration, we do this instance selection process at random.

Heuristic: Here, we use a heuristic derived from the input to decide which instances to pass to the OB model/next knowledge iteration. Specifically, we use the length of the question as the heuristic.

9.2.4 Performance Comparison Metric

We demonstrate the efficacy of our method by showing the **computation efficiency** and **accuracy** improvements. For computation efficiency, we use FLOPs as the metric. An alternative metric could be measuring the **latency** of inference; however, it is a machine-dependent metric. FLOPs, on the other hand, is machine-independent and hence a more reliable metric for comparison.

To <u>further</u> compare various confidence computation methods and baselines, we use another performance metric. For our approach, the reader inference cost (in FLOPs) and the accuracy vary with the prediction confidences thresholds. Therefore, to compare various confidence methods and comprehensively study their efficacy, we compute accuracies for a range of costs and plot an accuracy-cost curve (as shown in Figure 54). We plot a curve for each method and calculate the **area under the curve (AUC)** to quantify the overall performance of each method. The larger the AUC value, the better the method is as it implies higher accuracy on average across all computation costs.

Our work differs from existing work in the following aspects: (1) Firstly, the inference efficiency aspect has mostly been studied for classification tasks using encoder-based models. However, we focus on a more challenging task of open-domain QA with generative models. (2) Existing efficiency improvement methods typically require architectural changes, network manipulation, saliency quantification, knowledge distillation, or even complex training procedures. In contrast, our method is easy to implement, does not require such modifications, and could generalize easily to a variety of applications. Moreover, it can even complement these existing methods. (3) The computation efficiency in existing methods often comes with a compromise on accuracy. In contrast, we show that our method consistently achieves superior accuracy. (4) Finally, existing methods usually do not allow custom computation costs and require training a separate model for each computation budget. In contrast, our system can be adjusted to meet any given computational requirements.

9.3 Experiments and Results

Experimental Details: We conduct experiments with NQ (Kwiatkowski, Palomaki, Redfield, Collins, Parikh, Alberti, Epstein, Polosukhin, Devlin, Lee, et al. 2019a) Open and TriviaQA (Joshi et al. 2017) datasets. We use closed-book models from (Roberts, Raffel, and Shazeer 2020) and FiD open-book models (and retrieved passages) from (Izacard and Grave 2021). The closed-book models take just the question as input and its inference cost is 0.0046×10^{11} FLOPs for T5-small and 0.0615×10^{11} FLOPs for T5-large for the average input size (12 tokens) of NQ questions. On the other hand, the open-book readers also take the retrieved passages as input (truncated to 250 tokens each) and thus have a higher inference cost. For example, the cost of inference for the open-book FiD reader with 1 passage is 0.202×10^{11}

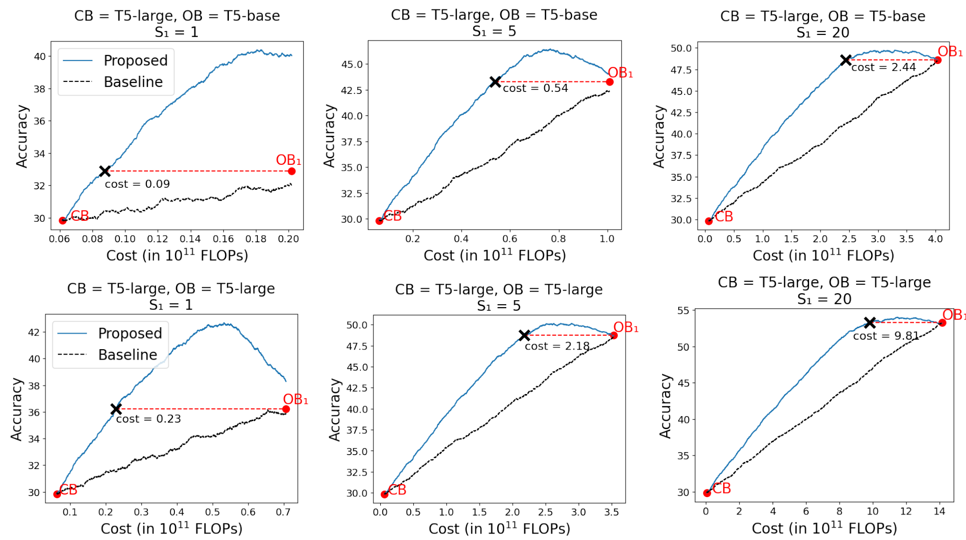


Figure 54. Accuracy-cost curves of the proposed system (blue) and baseline (black) for K=1 setting on NQ. Red points are the (cost, accuracy) values of the individual models CB and OB_1 (using S_1 passages). Point of intersection ($\times$) of red dashed line drawn from OB_1 on the blue curve corresponds to cost at which our system achieves the same accuracy as OB_1. Our method achieves this accuracy at considerably lower reader inference cost.

FLOPs for T5-base and 0.707×10^{11} FLOPs for T5-large. We compute these FLOP values using 'thop' library.

Accuracy-Cost Curves: As motivated previously, we plot accuracy-cost curves to study our method. We conduct experiments in multiple settings that differ in the (CB, OB) model combination, number of knowledge iterations with the OB model (K), and the S_k values. Figure 54, 55, and 56 show these curves for $K = 1$, 2, and 3 settings respectively. Each curve includes the following:

1. **Accuracy-cost points of individual systems**: The costs-accuracy point of the CB and OB_k models are represented by the red scatter points. These individual points correspond to the case in which all the instances are answered using the corresponding reader model.

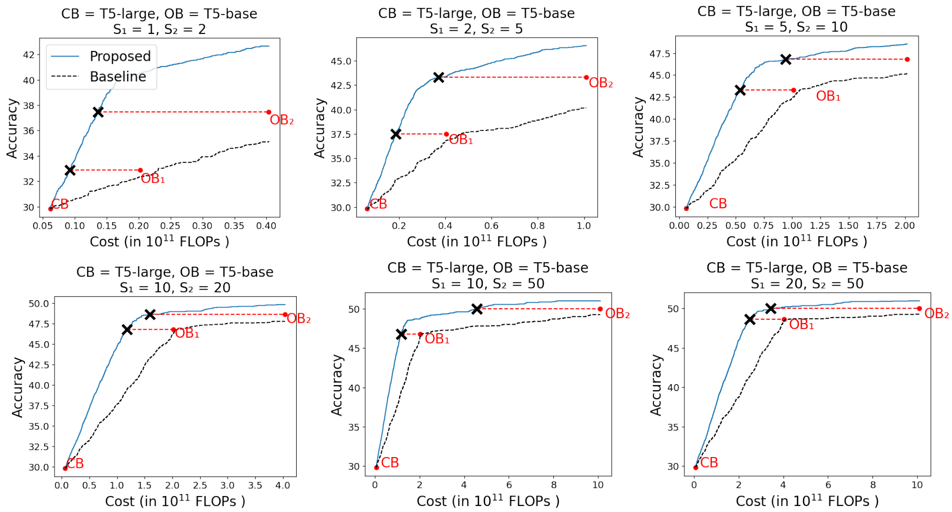


Figure 55. Accuracy-cost curves of the proposed method (in blue) and baseline (in black) for K=2 setting on NQ. Red points correspond to the individual models CB, OB_1 (S_1 passages), and OB_2 (S_2 passages).

2. **Accuracy-cost curve of the proposed method**: We explore multiple ways of computing prediction confidence of the generative models. We compare the AUC (of respective accuracy-cost curves) achieved by each method in We present an exhaustive comparison of these methods in Tables 53 and 54. We note that all confidence methods outperform the baselines. Since $\boldsymbol{P_{PA}}$ **yields the best performance**, we present its accuracy-cost curves in the main results.

3. **Costs at Equal Accuracies**: To measure the improvement in efficiency, we highlight (with $\times$) the costs at which the accuracy of the proposed system matches the accuracy of the OB_k models. For instance, in K=1 setting, we highlight the point at which the proposed system achieves the same accuracy as the OB_1 model (that use a fixed S_1 number of passages for all questions).

4. **Accuracy-cost curve of the baseline method**: To further demonstrate the

efficacy of P_{PA} method, we represent the accuracy-cost curve of the baseline method with black dashed line.

Next, we show the results for configurations with different number of knowledge iterations.

9.3.1 One Knowledge Iteration (K = 1)

In this setting, we first use the CB model and if it is not sufficiently confident then we use the OB model with S_1 knowledge statements. Figure 54 shows the accuracy-cost curves for different configurations of CB, OB, and S_1 values on NQ.

Improvement in Efficiency: The accuracy-cost curves show that the proposed system matches the accuracy of the OB model at a considerably lesser reader inference cost. This cost value corresponds to the point of intersection ($\times$) on the curve with a straight horizontal line drawn from OB_1. For example, in the case of (CB= T5-large, OB=T5-base, $S_1 = 5$), OB_1 achieves 43.30% accuracy using 1.01×10^{11} FLOPs while the proposed system achieves the same accuracy at just 0.54×10^{11} FLOPs. Such improvements are observed for all the cases. We show these curves for **an exhaustive set of configurations**. In this setting, our approach achieves cost improvements of up to 67.77%. This efficiency benefit comes from using the low-cost closed-book model for some instances where it is likely to be correct and additionally using the more expensive open-book model with S_1 passages only for the remaining instances. In the later results sections, we further discuss the overall cost improvement in reader inference (Table 57).

Improvement in Accuracy: From the curves, it is clear that the accuracy achieved by the proposed system surpasses the accuracy of the OB_1 model beyond the cost shown with $\times$. For example, in case of (CB= T5-large, OB=T5-base, S_1 = 1, the top-left figure), our system achieves the accuracy of up to 40.17% as compared to 32.88% accuracy of OB_1. Same as the efficiency improvement, such accuracy improvements are also observed across all configurations. We attribute this improvement to our approach's efficient use of external knowledge i.e. relying on the closed-book model when it is likely to be correct thus avoiding distraction with the external knowledge and using it only when it is required.

Distraction At Inference: The fact that our approach (that outputs its predictions using CB for some instances and OB for the others) outperforms the OB reader highlights that excessive external knowledge distracts the reader into giving incorrect answers while the CB reader answers them correctly. We note that near the OB_1 computation cost, the accuracy of our system begins to come closer to the OB_1's accuracy as more and more instances get answered by the OB_1 model.

Comparison with Baselines: From the accuracy-cost curves, it is clear that our proposed method that uses P_{PA} as the confidence (blue curve) outperforms the baseline (black curve) as the blue curve is consistently above the black and thus has a higher AUC. In Tables 53 and 54, we compare AUC values achieved by all the confidence measures: P_F, P_{FL}, P_A, and P_{PA}. All the approaches achieve considerably higher AUCs than the baselines while P_{PA} achieves the highest. This highlights the effectiveness of our confidence computation methods. We show this comparison

Method \ S_1	1	2	3	4	5	10	20	25	50	100
Random	30.96	33.28	34.73	35.74	36.05	38.12	39.27	39.14	39.96	39.94
Heuristic	31.90	33.97	35.62	36.32	36.90	38.77	39.69	39.62	40.22	40.28
P_{PA}	**36.61**	**38.53**	**39.64**	**40.59**	**41.14**	**42.82**	**43.69**	**43.69**	**44.43**	**44.29**
P_F	36.15	38.09	39.16	40.15	40.77	42.47	43.34	43.36	44.13	43.95
P_{FL}	36.13	38.08	39.15	40.16	40.76	42.46	43.35	43.36	44.16	43.97
P_A	36.39	38.35	39.49	40.45	41.00	42.74	43.56	43.54	44.30	44.15

Table 53. Comparing AUCs of accuracy-cost curves of different cascading techniques for (CB=T5-large and OB=T5-base) configuration.

Method \ S_1	1	2	3	4	5	10	20	25	50	100
Random	33.14	36.4	37.81	38.87	39.47	40.92	41.91	42.21	42.54	42.62
P_{PA}	**38.45**	**41.07**	**42.29**	**43.18**	**43.76**	**45.40**	**46.34**	**46.52**	**46.71**	**46.77**
P_F	38.00	40.68	41.92	42.87	43.47	45.11	46.09	46.31	46.45	46.46
P_{FL}	37.96	40.65	41.89	42.85	43.44	45.09	46.06	46.28	46.44	46.46
P_A	38.26	40.93	42.17	43.08	43.66	45.28	46.25	46.43	46.61	46.69

Table 54. Comparing AUCs of accuracy-cost curves of different cascading techniques for (CB=T5-large and OB=T5-large) configuration.

for other configurations also. As P_{PA} achieves the highest AUC, we use it for our subsequent experiments.

Comparing Overall Performance: We show the exact match accuracies achieved by various ODQA methods in Table 55. Our method achieves slightly higher performance that both FiD base and large models. We note that this is an additional benefit of our approach, though, the primary benefit remains to be the improvement in efficiency. The cost of FiD base model is 20.19×10^{11} FLOPs while our system matches its accuracy at just 13.10×10^{11} FLOPs. We show that the performance improves further on increasing the number of knowledge iterations in our method.

Method	NQ	TQA
Hard EM	28.8	50.9
ORQA	31.3	45.1
REALM	40.4	-
DPR	41.5	57.9
RAG	44.5	56.1
DensePhrases	41.5	56.8
PAQ	52.3	-
KG-FiD	53.4	69.8
FiD* (base)	50.03	68.01
Ours K= 1 (with FiD base)	**50.83**	**68.52**
Ours K= 2 (with FiD base)	**50.94**	**68.69**
Ours K= 3 (with FiD base)	**50.97**	**68.80**
FiD* (large)	54.43	72.07
Ours K= 1 (with FiD large)	**54.90**	**72.16**
Ours K= 2 (with FiD large)	**54.99**	**72.29**
Ours K= 3 (with FiD large)	**55.10**	**72.33**

Table 55. Comparing EM accuracy of ODQA methods. * indicates the highest performance of the latest model.

9.3.2 Two Knowledge Iterations (K = 2)

In this setting, we use CB model and then conditionally use two knowledge iterations with the OB model. Specifically, if CB is not sufficiently confident then we use the OB model with S_1 passages (OB_1), and if OB_1 is not sufficiently confident then we use OB with S_2 passages (OB_2). Figure 55 shows accuracy-cost curves for this setting.

Improvement in Efficiency: In this setting, our method achieves larger efficiency improvements than the K=1 setting. For example, in case of (CB=T5-large, OB=T5-

CB	OB	S_1	S_2	P_A	P_{PA}	Baseline
Large	Base	1	2	39.56	**39.79**	32.75
Large	Base	2	5	42.90	**43.14**	36.62
Large	Base	10	20	46.23	**46.30**	42.83
Large	Base	20	100	49.04	**49.09**	47.07
Large	Large	1	2	42.31	**42.48**	35.76
Large	Large	2	5	46.24	**46.35**	41.08
Large	Large	10	20	49.55	**49.71**	46.51
Large	Large	20	100	52.87	**52.97**	51.44

Table 56. Comparing AUCs of accuracy-cost curves of the proposed and the baseline methods in K=2 setting.

base, S_1=1, S_2=2), OB_2 achieves 37.48% accuracy at the cost of 0.4×10^{11} FLOPs and cascading system achieves the same accuracy at just 0.13×10^{11} FLOPs. We achieve similar efficiency improvements over OB_1 model also; in the same case, OB_1 achieves 32.88% accuracy at 0.2×10^{11} FLOPs and our system achieves the same accuracy at just 0.09×10^{11} FLOPs. Similar improvements are observed for all the configurations.

Improvement in Accuracy: As can be observed from the accuracy-cost curves, our system achieves a higher accuracy than even the OB_2 model. For example, in case of (CB=T5-large, OB=T5-base, S_1=10, S_2=20), our system achieves accuracy of 49.83% that is considerably higher than the 48.61% accuracy of OB_2 and 46.78% accuracy of OB_1. Furthermore, at the same cost as OB_1, cascading system achieves 48.98% accuracy, 2.2% higher than that of OB_1 system (46.78%). Thus, our method improves both the reader computation efficiency and the prediction accuracy.

Comparison with Baseline: In Table 56, we show AUCs for different config-

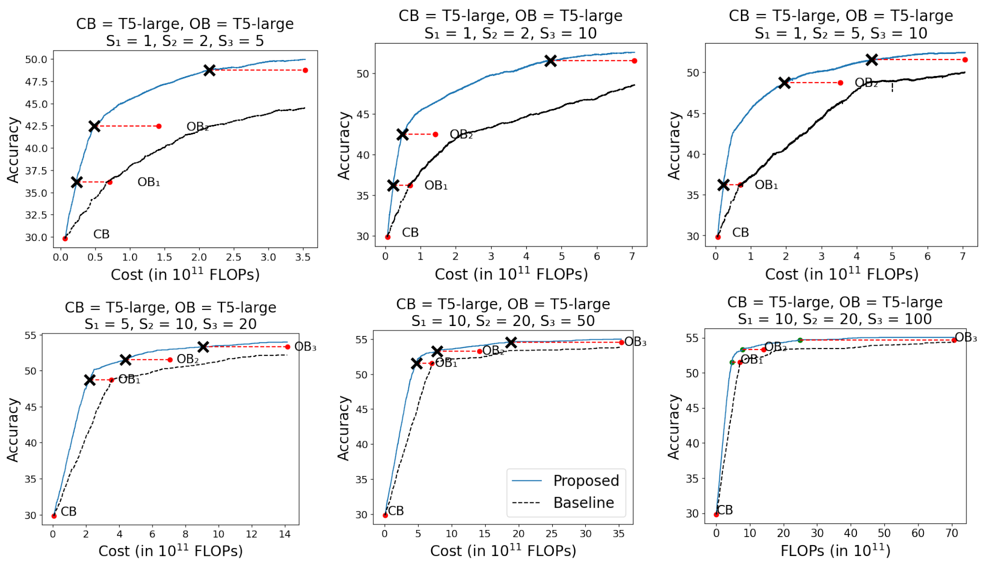


Figure 56. Accuracy-cost curves of the proposed method (in blue) and baseline (in black) for K=3 setting on NQ. Red points correspond to the accuracy and cost values of the individual CB, OB_1, OB_2, and OB_3 models.

urations. Same as K=1 setting, the proposed system clearly achieves higher AUC than the baseline.

9.3.3 Three Knowledge Iterations (K = 3)

In this setting, we first use the CB model and then conditionally use three knowledge iterations with the OB model with S_1, S_2, and S_3 passages respectively. Figure 56 shows the accuracy-cost curves.

Improvement in Efficiency and Accuracy: We observe both efficiency and accuracy improvements in this setting also. For example, in (CB=T5-large, OB=T5-base, S_1=10, S_2=20, S_3=100) configuration, OB_3 achieves 50.03% accuracy at cost of 20.19×10^{11} FLOPs and our system achieves the same accuracy at just 5.03×10^{11}

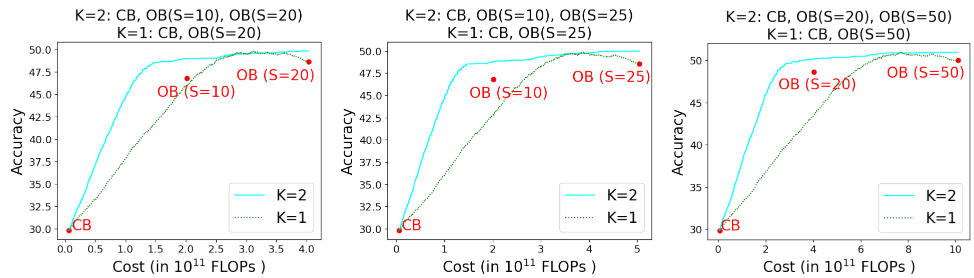


Figure 57. Illustrating the impact of multiple knowledge iterations by plotting accuracy-cost curves for K=1 and K=2 settings together. The system using two iterations (K=2) achieves higher AUC than its counterpart using the same amount of total knowledge (20, 25, and 50 in the three cases respectively) but with just one iteration.

FLOPs. Furthermore, our system even achieves higher accuracy than even OB_3. Finally, the proposed system achieves AUC of 49.62 which is considerably higher than that of the baseline (47.7).

9.4 Impact of Knowledge Iterations

We demonstrate the impact of multiple knowledge iterations by plotting the accuracy-cost curves for K=1 and K=2 settings together in Figure 57. The system using two iterations (K=2) achieves higher AUC than its counterpart using the same amount of knowledge but with just one iteration. For the first case, we use CB and OB model (with S=20) in K=1 setting, and in the K=2 setting, we introduce an intermediate step that uses 10 passages i.e. we use CB, OB_1 (with S_1=10), and OB_2 (with S_2=20). The total amount of knowledge is same in both the scenarios (20 contexts) but K=2 system tries to first answer the question using just 10 passages while the K=1 system directly uses 20 passages. The K=2 system achieves higher

Method	**Cost (in 10^{11} FLOPs)**
FiD (base)	20.19
Ours (at same EM as FiD base)	3.69
FiD (large)	70.69
Ours (at same EM as FiD large)	20.59

Table 57. Comparing reader inference cost of FiD and our system at equivalent exact match accuracies on NQ.

AUC than K=1 (46.25 vs 43.56). This pattern is seen in all the cases thus highlighting the positive impact of using knowledge iterations with OB model.

9.5 Comparing Overall Performance

In Table 55, we compare the performance achieved by our system in different K settings. With the increase in the value of K, the improvement in performance also increases. On NQ Open, our system achieves accuracy of up to 55.10% with large model and up to 50.97% with base model outperforming all other reader methods such as Hard EM, ORQA, REALM, DPR, RAG, DensePhrases, PAQ, FiD, and KG-FiD. Similar improvements are also observed on the TriviaQA. Finally, in Table 57, we compare the cost of reader inference at equal EMs and show that our method achieves considerable efficiency improvements over the FiD reader.

9.6 Conclusion

Addressing the problem of high-inference cost of reader models in open-domain QA, we investigated an approach that utilizes both the ‘closed-book’ and the ‘open-book’ inferences and dynamically reads the external knowledge in multiple ‘knowledge iterations’. Through comprehensive experiments, we demonstrated that this dynamic reading approach improves both the **inference efficiency** and the **prediction accuracy** of the reader. Compared with the Fusion-in-Decoder reader, this approach matches its accuracy by utilizing just 18.32% of its reader inference cost (FLOPs) and also outperforms it by achieving up to 55.10% accuracy on NQ Open and 72.33% on TriviaQA. Finally, we believe our work will encourage further research and facilitate development of **efficient QA reader systems**.

Chapter 10

ACHIEVING TRAINING DATA EFFICIENCY FOR NATURAL LANGUAGE INFERENCE USING PHL TRIPLET GENERATION

Transformer-based models achieve impressive performance on numerous Natural Language Inference (NLI) benchmarks when trained on respective training datasets. However, in certain cases, training samples may not be available or collecting them could be time-consuming and resource-intensive. In this chapter, we address the above challenge and present an explorative study on unsupervised NLI, a paradigm in which no human-annotated training samples are available. We investigate it under three settings: *PH, P*, and *NPH* that differ in the extent of unlabeled data available for learning. As a solution, we propose a procedural data generation approach that leverages a set of sentence transformations to collect PHL (Premise, Hypothesis, Label) triplets for training NLI models, bypassing the need for human-annotated training data. Comprehensive experiments with several NLI datasets show that the proposed approach results in accuracies of up to $66.75\%, 65.9\%, 65.39\%$ in PH, P, and NPH settings respectively, outperforming all existing unsupervised baselines. Furthermore, fine-tuning our model with as little as ~0.1% of the human-annotated training dataset (500 instances) leads to 12.2% higher accuracy than the model trained from scratch on the same 500 instances. Supported by this superior performance, we conclude with a recommendation for collecting high-quality task-specific data.

10.1 Introduction

Natural Language Inference (NLI) is the task of determining whether a "hypothesis" is true (Entailment), false (Contradiction), or undetermined (Neutral) given a "premise". State-of-the-art models have matched human performance on several NLI benchmarks, such as SNLI (Bowman et al. 2015), Multi-NLI (Williams, Nangia, and Bowman 2018), and Dialogue NLI (Welleck et al. 2019). This high performance can be partially attributed to the availability of large training datasets; SNLI (570k), Multi-NLI (392k), and Dialogue-NLI (310k). For new domains, collecting such training data is time-consuming and can require significant resources. What if no training data was available at all?

In this work, we address the above question and explore *Unsupervised NLI*, a paradigm in which no human-annotated training data is provided for learning the task. We study three different unsupervised settings: *PH, P*, and *NPH* that differ in the extent of unlabeled data available for learning. In PH-setting, unlabeled premise-hypothesis pairs are available i.e. data without ground-truth labels. In P-setting, only a set of premises are available i.e. unlabeled partial inputs. The third setting NPH does not provide access to any training dataset, and thus it is the hardest among the three unsupervised settings considered in this work.

We propose to solve these unsupervised settings using a procedural data generation approach. Given a sentence, our approach treats it as a premise (P) and generates multiple hypotheses (H) corresponding to each label (L = Entailment, Contradiction, and Neutral) using a set of sentence transformations (refer to Figure 58). This

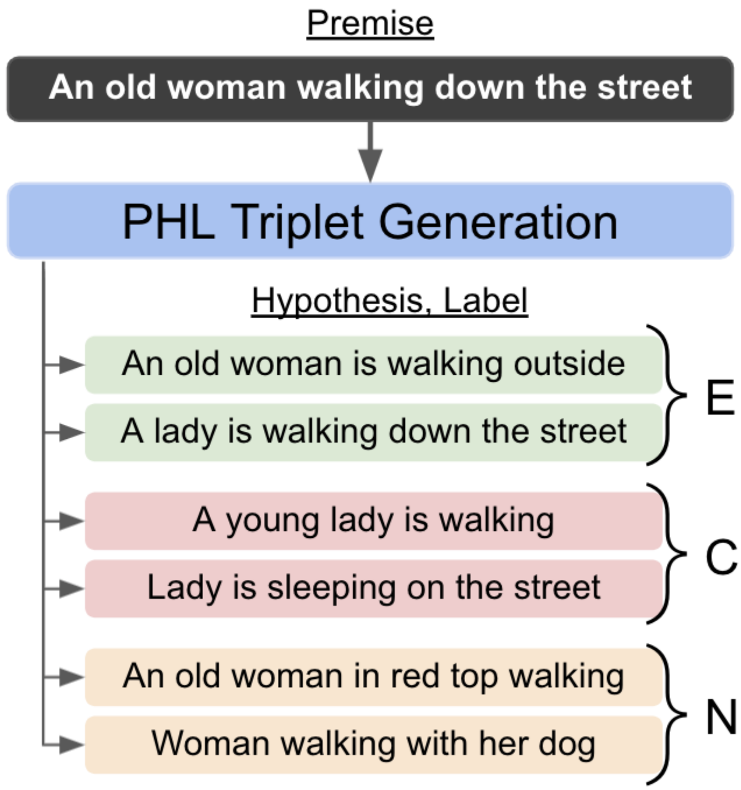


Figure 58. Illustrating our procedural data generation approach for unsupervised NLI. A sentence is treated as premise, and multiple hypotheses conditioned on each label (Entailment- E, Contradiction- C, and Neutral- N) are generated using a set of sentence transformations.

results in creation of Premise-Hypothesis-Label (PHL) triplets that can be used for training the NLI model. In the P and PH settings, we directly apply our sentence transformations over the available premises to generate PHL triplets. However, in the NPH setting, premises are not available. We tackle this challenge by incorporating a premise generation step that extracts sentences from various raw text corpora such as Wikipedia and short stories. We use these extracted sentences as premises to generate PHL triplets. In Figure 59, we compare the four settings (one supervised and three unsupervised) and show our approach to develop an NLI model for each setting.

To evaluate the efficacy of the proposed approach, we conduct comprehensive

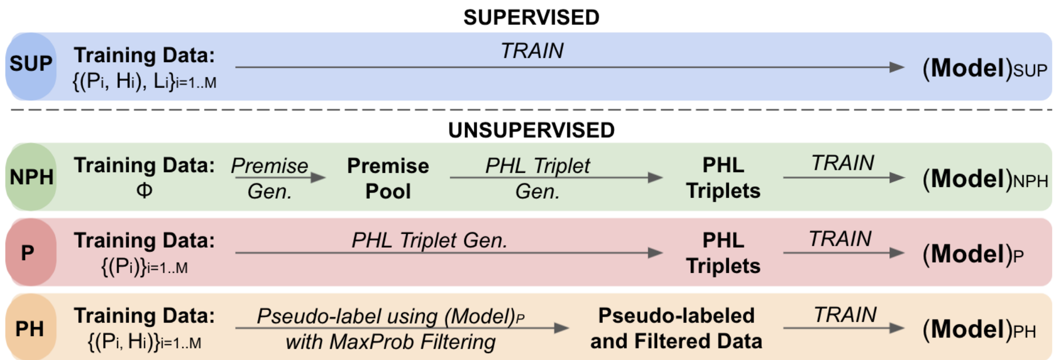


Figure 59. Comparing supervised NLI with our three unsupervised settings. For unsupervised settings, we procedurally generate PHL triplets to train the NLI model. In NPH setting, a premise pool is collected from raw text corpora such as Wikipedia and then used for generating PHL triplets. In P setting, we directly apply these transformations on the available premises. In PH setting, we leverage the P-setting model to pseudo-label and filter the provided unlabeled PH pairs and then train the NLI model using this pseudo-labeled dataset.

experiments with several NLI datasets. We show that our approach results in accuracies of 66.75%, 65.9%, and 65.39% on SNLI dataset in PH, P, and NPH settings respectively, outperforming all existing unsupervised methods by ~13%. We also conduct experiments in low-data regimes where a few human-annotated labeled instances are provided and show that further fine-tuning our models with these instances consistently achieves higher performance than the models fine-tuned from scratch. For example, with just 500 labeled instances, our models achieve 8.4% and 10.4% higher accuracy on SNLI and MNLI datasets respectively. Finally, we show that fine-tuning with 'adversarial' instances instead of randomly selected human-annotated instances further improves the performance of our models; it leads to 12.2% and 10.41% higher accuracy on SNLI and MNLI respectively.

In summary, our contributions are as follows:

1. We explore three unsupervised settings for NLI and propose a procedural data generation approach that outperforms the existing approaches by ~13% and raises the state-of-the-art unsupervised performance on SNLI to 66.75%.
2. We also conduct experiments in low-data regimes and demonstrate that further fine-tuning our models with the provided instances achieves 8.4% and 10.4% higher accuracy on SNLI and MNLI datasets respectively.
3. Finally, we show that using 'adversarial' instances for fine-tuning instead of randomly selected instances further improves the accuracy. It leads to 12.2% and 10.41% higher accuracy on SNLI and MNLI respectively. Supported by this superior performance, we conclude with a recommendation for collecting high-quality task-specific data.

We release the implementation of our procedural data generation approach and hope that our work will encourage research in developing techniques that reduce reliance on expensive human-annotated data for training task-specific models.

10.2 Unsupervised NLI

In NLI, a premise-hypothesis pair (P, H) is provided as input and the system needs to determine the relationship $L \in \{Entailment, Contradiction, Neutral\}$ between P and H. In the **supervised setting**, a labeled dataset $D_{train} = \{(P_i, H_i), L_i\}_{i=1}^{M}$ consisting of M instances which are usually human-annotated is available for training. However in the unsupervised setting, labels L_i are not available, thus posing a significant challenge for training NLI systems. Along with this standard unsupervised setting (referred to as PH), we consider two novel unsupervised settings (P and NPH) that differ in the extent of unlabeled data available for learning: **PH-setting:** It corresponds to the standard unsupervised setting where an unlabeled dataset of PH pairs ($\{(P_i, H_i)\}_{i=1}^{M}$) is provided.

P-setting: In this setting, only premises from D_{train} i.e ($\{(P_i)\}_{i=1}^{M}$) are provided. It is an interesting setting as the large-scale NLI datasets such as SNLI (Bowman et al. 2015) and MultiNLI (Williams, Nangia, and Bowman 2018) have been collected by presenting only the premises to crowd-workers and asking them to write a hypothesis corresponding to each label. Furthermore, this setting presents a harder challenge for training NLI systems than the PH-setting as only partial inputs are provided.

NPH-setting: Here, no datasets (even with partial inputs) are provided. Thus, it corresponds to the hardest unsupervised NLI setting considered in this work. This setting is of interest in scenarios where we need to make inferences on a test dataset but its corresponding training dataset is not available in any form.

From the above formulation, it can be inferred that the hardness of the task increases with each successive setting (PH→P→NPH) as lesser and lesser information is made available. In order to address the challenges of each setting, we propose a two-step approach that includes a pipeline for procedurally generating PHL triplets from the limited information provided in each setting (Section 10.3), followed by training an NLI model using this procedurally generated data (Section 10.4). Figure 59 highlights the differences between four NLI settings (one supervised and three unsupervised) and summarizes our approach to develop an NLI model for each setting.

10.3 PHL Triplet Generation

To compensate for the absence of labeled training data, we leverage a set of sentence transformations and procedurally generate PHL triplets that can be used for training the NLI model. In P and PH settings, we apply these transformations on the provided premise sentences. In the NPH setting where premises are not provided, we extract sentences from various raw text corpora and apply these transformations on them to generate PHL triplets.

10.3.1 $\mathcal{P}$: Premise Generation

We extract sentences from raw text sources, namely, COCO captions (Lin et al. 2014), ROC stories (Mostafazadeh et al. 2016), and Wikipedia to compile a set of premises for the NPH setting. We use these text sources as they are easily available and contain a large number of diverse sentences from multiple domains.

ROC Stories is a collection of short stories consisting of five sentences each. We include all these sentences in our premise pool. **MS-COCO** is a dataset consisting of images with five captions each. We add all captions to our premise pool. From **Wikipedia**, we segment the paragraphs into individual sentences and add them to our premise pool.

We do not perform any sentence filtration during the premise collection process. However, each transformation (described in subsection 10.3.2) has its pre-conditions

Transformation	Original Sentence (Premise)	Hypothesis	Label
PA	Fruit and cheese sitting on a black plate	There is fruit and cheese on a black plate	E
PA + ES + HS	A large elephant is very close to the camera	Elephant is close to the photographic equipment	E
CW-noun	Two horses that are pulling a carriage in the street	Two dogs that are pulling a carriage in the street	C
CV	A young man sitting in front of a TV	A man in green jersey jumping on baseball field	C
PA + CW	A woman holding a baby while a man takes a picture of them	A kid is taking a picture of a male and a baby	C
FCon	A food plate on a glass table	A food plate made of plastic on a glass table	N
PA + AM	Two dogs running through the snow	The big dogs are outside	N

Table 58. Illustrative examples of PHL triplets generated from our proposed transformations. E,C, and N correspond to the NLI labels Entailment, Contradiction, and Neutral respectively.

such as presence of verbs/adjectives/nouns that automatically filter out sentences from the premise pool that can not be used for PHL triplet generation.

10.3.2 $\mathcal{T}$: Transformations

Now, we present our sentence transformations for each NLI label. Table 58 illustrates examples of PHL triplets generated from these transformations.

10.3.2.1 Entailment:

In NLI, the label is entailment when the hypothesis must be true if the premise is true. Table 59 shows examples of our transformations. **Paraphrasing (PA):**

Category	Original Sentence (Premise)	Hypothesis
PA	Fruit and cheese sitting on a black plate.	There is fruit and cheese on a black plate.
ES	person relaxes at home while holding something.	person relaxes while holding something.
HS.	A girl is sitting next to a blood hound.	A girl is sitting next to an animal.
PS	People are walking down a busy city street.	they are walking down a busy city street
CT	A man and woman setup a camera.	Two people setup a camera
Composite	A large elephant is very close to the camera.	elephant is close to the photographic equipment.

Table 59. Illustrative examples of entailment transformations.

Paraphrasing corresponds to expressing the meaning of a text (restatement) using other words and hence results in entailment premise-hypothesis pairs. We use the Pegasus (J. Zhang et al. 2019) tool to generate up to 10 paraphrases of a sentence and use them as hypothesis with the original sentence as the premise

Extracting Snippets (ES): We use dependency parse tree to extract meaningful snippets from a sentence and use them as hypothesis with the original sentence as the premise. Specifically, we extract sub-trees that form a complete phrase or a sentence. For example, from the sentence "*A person with red shirt is running near the garden*", we create entailing hypotheses "*A person is running near the garden*", "*A person is running*", "*A person is near the garden*", etc. We implement 10 such techniques using spacy (Honnibal et al. 2020).

Hypernym Substitution (HS): A hypernym of a word is its supertype, for example, "animal" is a hypernym of "dog". We use WordNet (Miller 1995) to collect hypernyms and replace noun(s) in a sentence with their corresponding hypernyms to create entailment hypothesis. For example, from the premise "*A black dog is sleeping*", we create "*A black animal is sleeping*". Note that swapping the premise and hypothesis in this case gives us another PH pair that has a 'Neutral' relationship.

Pronoun Substitution (PS): Here, we leverage Part-of-Speech (POS) tagging of spacy to heuristically substitute a noun with its mapped pronoun. For example, substituting "boy" with "he" in the sentence "*boy is dancing in arena*" results in an entailing hypothesis "*he is dancing in arena*".

Counting (CT): Here, we count nouns with common hypernyms and use several templates such as "*There are {count} {hypernym}s present*" to generate entailing hypotheses. For instance, from the sentence "*A motorbike and a car are parked*", we create hypothesis "*Two automobiles are parked*". We also create contradiction hypotheses using the same templates by simply changing the *count* value such as "*There are five automobiles present*".

10.3.2.2 Contradiction:

The label is contradiction when the hypothesis can never be true if the premise is true. Table 60 shows examples of our transformations.

Contradictory Words (CW): We replace noun(s) and/or adjective(s) (identified using spacy POS tagging) with their corresponding contradictory words. For example, replacing the word 'big' with 'small' in "*He lives in a big house*" results in a contradictory hypothesis "*He lives in a small house*". For contradictory adjectives, we

Category	Original Sentence (Premise)	Hypothesis
CW-noun	A small bathroom with a sink under a cabinet.	a small kitchen with a sink under a cabinet.
CW-adj	A young man is doing a trick on a surfboard.	A old man is doing a trick on a surfboard.
CV	A couple pose for a picture while standing next to a couch.	A couple sit in a chair on laptops
SOS	A man is flying a kite on the beach.	a beach is flying a kite on the man
NS	Two green traffics lights in a European city.	nine green traffics lights in a European city
IrH.	A flock of sheep grazing in a field.	A man having fun as he glides across the water.
NI.	A boy with gloves on a field throwing a ball.	a boy with gloves on a field not throwing a ball
Composite	A woman holding a baby while a man takes a picture of them	a kid is taking a picture of a male and a baby.

Table 60. Illustrative examples of contradiction transformations.

collect antonyms from wordnet and for nouns, we use the function '*most_similar*' from gensim (Rehurek and Sojka 2011) .

Contradictory Verb (CV): We collect contradictory verbs from gensim and create hypothesis in the following two ways: (i) substituting verb with its contradictory verb: for example, from "*A girl is walking*", we create hypothesis "*A girl is driving*" and (ii) selecting other sentences from the premise pool that have the same subject as the original sentence but have contradictory verbs: for example, sentences like "*A young girl is driving fast on the street*" and "*There is a girl skiing with her mother*". The second approach adds diversity to our synthetically generated PHL triplets.

Subject Object Swap (SOS): We swap the subject and object of a sentence to create a contradictory hypothesis. For example, from the sentence "*A clock is*

Category	Original Sentence (Premise)	Hypothesis
AM	two cats are eating next to each other out of the bowl	two cats are eating next to each other out of the same bowl
SSNCV	A man holds an electronic device over his head.	man is taking photo with a small device
FCon	a food plate on a table with a glass.	a food plate on a table with a glass which is made of plastic.
Composite	two dogs running through the snow.	The big dogs are outside.

Table 61. Illustrative examples of neutral transformations.

standing on top of a concrete pillar", we create a contradictory hypothesis "*a pillar is standing on top of a concrete clock*".

Negation Introduction (NI): We introduce negation into a sentence to create a contradictory hypothesis. For example, from the sentence "*Empty fog covered streets in the night*", we create hypothesis "*Empty fog did not cover streets in the night*".

Number Substitution (NS): Here, we change numbers (tokens with dependency tag '*nummod*' in the parse tree) in a sentence. For example, changing 'four' to 'seven' in the sentence "*Car has four red lights*" results in a contradictory hypothesis.

Irrelevant Hypothesis (IrH): We sample sentences that have different subjects and objects than the premise sentence. For example, for the premise "*Sign for an ancient monument on the roadside*", we sample "*A man goes to strike a tennis ball*" as a contradictory hypothesis.

10.3.2.3 Neutral:

The label is neutral when the premise does not provide enough information to classify a PH pair as either entailment or contradiction. Table 61 shows examples of our transformations.

Adding Modifiers (AM): We introduce a *relevant* modifier for noun(s) in premise to generate a neutral hypothesis. For instance, in the sentence "*A car parked near the fence*", we insert modifier 'silver' for the noun 'car' and create hypothesis "*A silver car parked near the fence*". We collect relevant modifiers for nouns by parsing sentences in the premise pool and selecting tokens with dependency tag '*amod*' and POS tag '*ADJ*'.

ConceptNet (Con): We add relevant information from ConceptNet (Speer, Chin, and Havasi 2017) relations ('AtLocation', 'DefinedAs', etc.) to the premise and create a neutral hypothesis. For instance, from the sentence "*Bunch of bananas are on a table*", we create hypothesis"*Bunch of bananas are on a table at kitchen*" using the 'AtLocation' relation.

Same Subject but Non-Contradictory Verb (SSNCV): For a premise, we select sentences from the premise pool that have the same subject as the premise, contain additional noun(s) but no contradictory verbs as neutral hypotheses. For instance, for premise "*A small child is sleeping in a bed with a bed cover*", we sample "*A child laying in bed sleeping with a chair near by*" as a hypothesis.

We create more examples by swapping premise and hypothesis of the collected PHL triplets and accordingly change the label. For instance, swapping P and H in

HS, ES, etc. results in neutral examples, swapping P and H in **AM, Con** results in entailment examples. Furthermore, we note that transformations **ES, HS, PS, SOS, NI** result in PH pairs with high word overlap between premise and hypothesis sentences, whereas, transformation **PA, CV, IrH, SSNCV, etc.** result in PH pairs with low word overlap. In order to add more diversity to the examples, we use composite transformations on the same sentence such as **PA** + **ES** ($L = E$), **PA** + **CW** ($L = C$) as shown in Table 58.

10.3.3 Data Validation

In order to measure the correctness of our procedurally generated PHL triplets, we validate randomly sampled 50 instances for each transformation. We find that nearly all the instances get correct label assignments in case of **PA, HS, PS, NI, NS, IrH, AM** transformations. While transformations **CW, Con, SSNCV** result in a few mislabeled instances. Specifically, **SSNCV** transformation results in the maximum errors (5). Table 70 shows examples of mis-labeled instances generated by our transformations. While it is beneficial to have noise-free training examples, doing so would require more human effort and increase the data collection cost. Thus, in this work, we study how well we can do solely using the procedurally generated data without investing human effort in either creating instances or eliminating noise.

Transformation $\mathcal{T}$	NPH-Setting			P-Setting
	$\mathcal{T}(\mathcal{P}(\text{C}))$	$\mathcal{T}(\mathcal{P}(\text{R}))$	$\mathcal{T}(\mathcal{P}(\text{W}))$	$\mathcal{T}(\text{SNLI})$
Raw Sentences	591	490	600	548
PA	5083	3072	273	475
ES	2365	196	87	516
PS	37	41	137	38
CT	25	8	2	43
Neg.	1175	1175	2053	990
CW	978	119	116	265
CV	1149	63	5	505
NS	73	16	224	91
SOS	428	180	229	76
AM	1048	125	535	327
SSNCV	1363	2	7	405

Table 62. Sizes of PHL triplet datasets generated by our transformations for the unsupervised settings. All numbers are in thousands. C, R, W denote COCO, ROC Stories, and Wikipedia respectively. For P-Setting, we show stats for SNLI dataset. We do not include PH-Setting in this table because we leverage the PHL triplets generated using the P-Setting to solve it as described in Section 10.4.3.

10.4 Training NLI Model

In this section, we describe our approach to develop NLI models for each unsupervised setting.

Table 62 shows sizes of the generated PHL datasets for each setting.

10.4.1 NPH-Setting

We use the Premise Generation function ($\mathcal{P}$) over raw-text sources, namely, COCO captions, ROC stories, and Wikipedia i.e., $\mathcal{P}$(COCO), $\mathcal{P}$(ROC), and $\mathcal{P}$(Wiki) to compile a set of premises and apply the transformations ($\mathcal{T}$) over them to generate PHL triplets. We then train a transformer-based 3-class classification model (Section 10.5.1) using the generated PHL triplets for the NLI task.

10.4.2 P-Setting

In this slightly relaxed unsupervised setting, premises of the training dataset are provided. We directly apply the transformation functions ($\mathcal{T}$) on the given premises and generate PHL triplets. Similar to the NPH setting, a 3-class classification model is trained using the generated PHL triplets.

10.4.3 PH-Setting

In this setting, unlabeled training data is provided. We present a 2-step approach to develop a model for this setting. In the first step, we create PHL triplets from the premises and train a model using the generated PHL triplets (same as the P-setting). In the second step, we **pseudo-label** the unlabeled PH pairs using the model trained in Step 1.

Here, a naive approach to develop NLI model would be to train using this pseudo-labeled dataset. This approach is limited by confirmation bias i.e overfitting to incorrect pseudo-labels predicted by the model (Arazo et al. 2020). We address this by filtering instances from the pseudo-labeled dataset based on the model's prediction confidence. We use the maximum softmax probability (maxProb) as the confidence measure and select only the instances that have high prediction confidence for training the final NLI model. This approach is based on prior work (Hendrycks and Gimpel 2017) showing that correctly classified examples tend to have greater maximum softmax probabilities than erroneously classified examples. Furthermore, we investigate two ways of training the final NLI model: **Augmenting with $\mathcal{T}(P)$:** Train using the selected pseudo-labeled dataset and the PHL triplets generated in Step 1.

Further Fine-tune P-Model: Further fine-tune the model obtained in Step 1 with the selected pseudo-labeled dataset instead of fine-tuning one from scratch.

10.5 Experiments

10.5.1 Experimental Setup

Datasets: We conduct comprehensive experiments with a diverse set of NLI datasets: SNLI (Bowman et al. 2015) (sentence derived from only a single text genre), Multi-NLI (Williams, Nangia, and Bowman 2018) (sentence derived from multiple text genres), Dialogue NLI (Welleck et al. 2019) (sentences from context of dialogues), and Breaking NLI (Glockner, Shwartz, and Goldberg 2018) (adversarial instances).

Model: We use BERT-BASE model (Devlin et al. 2019) with a linear layer on top of [CLS] token representation for training the 3-class classification model. We trained models for 5 epochs with a batch sizes of 32 and a learning rate ranging in $\{1–5\}e–5$. All experiments are done with Nvidia V100 16GB GPUs.

Model	SNLI	MNLI mat.	MNLI mis.	DNLI	BNLI
BERT*	35.09	-	-	-	-
LXMERT*	39.03	-	-	-	-
VilBert*	43.13	-	-	-	-
$\mathcal{T}(\mathcal{P}(\mathrm{C}))$	64.8	**49.01**	**50.0**	**50.26**	74.73
$\mathcal{T}(\mathcal{P}(\mathrm{R}))$	58.51	45.44	45.93	47.4	67.9
$\mathcal{T}(\mathcal{P}(\mathrm{W}))$	55.06	44.15	44.25	48.48	62.58
$\mathcal{T}(\mathcal{P}(\mathrm{C+R}))$	**65.39**	46.83	46.92	47.95	**77.37**
$\mathcal{T}(\mathcal{P}(\mathrm{C+R+W}))$	65.09	46.63	46.83	44.74	56.11

Table 63. Comparing accuracy of models in the NPH-setting. C, R, and W correspond to the premise sources COCO, ROC, and Wikipedia respectively. Results marked with * have been taken from Cui, Zheng, and Wang 2020.

Baseline Methods: We compare our approach with Multimodal Aligned Contrastive Decoupled learning **(MACD)** (Cui, Zheng, and Wang 2020) , Single-modal pre-training model **BERT** (Devlin et al. 2019), Multi-modal pre-training model **LXMERT** (Tan and Bansal 2019), and **VilBert** (Lu et al. 2019).

10.5.2 Results

NPH-Setting: We utilize three raw text sources: COCO, ROC, and Wikipedia to compile a premise pool and then generate PHL triplets from those premises. Table 63 shows the accuracy of models in this setting. We use equal number of PHL triplets (150k class-balanced) for training the NLI models. We find that **the model trained on PHL triplets generated from COCO captions as premises**

Approach	SNLI	MNLI mat.	MNLI mis.	DNLI	BNLI
BERT*	35.09	-	-	-	-
LXMERT*	39.03	-	-	-	-
VilBert*	43.13	-	-	-	-
MACD*	52.63	-	-	-	-
$\mathcal{T}$(SNLI)	65.72	49.56	50.00	43.27	67.78
+$\mathcal{T}(\mathcal{P}$(C))	65.36	49.91	49.24	46.25	70.07
$\mathcal{T}(\mathcal{P}$(R))	**65.90**	48.53	48.36	44.97	66.43

Table 64. Comparing accuracy of various approaches in the P-Setting. Results marked with * have been taken from Cui, Zheng, and Wang 2020. Note that we utilize the premises of the SNLI training dataset only but evaluate on SNLI (in-domain), and MNLI, DNLI, BNLI (out-of-domain).

Method	Data	SNLI	MNLI mat.	MNLI mis.
From Scratch	MaxProbFilt	66.67	**53.37**	**55.17**
From Scratch	MaxProbFilt+$\mathcal{T}(P)$	**66.75**	50.22	50.37
Finetune P-model	MaxProbFilt	65.60	52.97	53.44

Table 65. Comparing accuracy of our proposed approaches in the PH-Setting. Note that the models are trained using PH pairs only from the SNLI train-set but evaluated on MNLI (out-of-domain dataset) also.

outperforms ROC and Wikipedia models on all datasets. We attribute this superior performance to the short, simple, and diverse sentences present in COCO that resemble the premises of SNLI that were collected from Flickr30K (Plummer et al. 2015) dataset. In contrast, Wikipedia contains lengthy and compositional sentences resulting in premises that differ from those present in SNLI, MNLI, etc. Furthermore, we find that **combining the PHL triplets of COCO and ROC leads to a slight improvement in performance on SNLI (**65.39%**), and BNLI (**77.37%**) datasets**.

Training Dataset	Method	100		200		500		1000		2000	
		SNLI	MNLI	SNLI	MNLI	SNLI	MNLI	SNLI	MNLI	SNLI	MNLI
SNLI	BERT	44.62	37.36	48.97	34.71	58.54	44.01	65.36	37.24	72.51	45.59
	NPH (Random)	**64.82**	**49.72**	**65.06**	**50.48**	**66.97**	**52.33**	**70.61**	**56.75**	**73.7**	**59.0**
	NPH (Adv.)	**68.21**	**51.93**	**69.23**	**56.55**	**70.85**	**58.46**	**73.62**	**59.47**	**74.31**	**60.43**
MNLI	BERT	35.12	36.01	35.14	36.58	46.16	47.1	47.64	56.21	53.68	**63.3**
	NPH (Random)	**63.87**	**52.85**	**63.87**	**53.61**	**64.23**	**57.47**	**65.62**	**60.42**	**66.87**	62.89

Table 66. Comparing performance of various methods on in-domain and out-of-domain datasets in **low-data regimes** (100-2000 training instances). 'BERT' method corresponds to fine-tuning BERT over the provided instances from SNLI/MNLI, 'NPH (Random)' corresponds to further fine-tuning our NPH model with the randomly sampled instances from SNLI/MNLI, 'NPH (Adv.)' corresponds to further fine-tuning our NPH model with the adversarially selected instances from SNLI/MNLI.

P-Setting: Cui, Zheng, and Wang 2020 presented MACD that performs multi-modal pretraining using COCO and Flick30K caption data for the unsupervised NLI task. It achieves 52.63% on the SNLI dataset. **Our approach outperforms MACD and other single-modal and multi-modal baselines by** ~13% on SNLI as shown in Table 64. We also experiment by adding PHL triplets generated from COCO and ROC to the training dataset that further improves the accuracy to 65.90% and establish a new state-of-the-art performance in this setting.

PH-Setting: Here, we first pseudo-label the given unlabeled PH pairs using the P-model and then select instances based on the maximum softmax probability (Section 10.4.3). We refer to this set of selected instances as *MaxProbFilt* dataset. This approach results in accuracy of 66.67% on the SNLI dataset as shown in Table 65. We investigate two more approaches of training the NLI model. In the first approach, we train using *MaxProbFilt* and PHL triplets generated from premises. In the second approach, we further fine-tune the P-model with *MaxProbFilt* dataset. We find that the first approach slightly improves the accuracy to 66.75%. This also represents our best performance across all the unsupervised settings. Furthermore, we observe **improvement in the Out-of-domain datasets also** (53.37% and 55.17% on MNLI matched and mismatched datasets respectively).

10.5.3 Low-Data Regimes

We also conduct experiments in low-data regimes where a few labeled instances are provided. We select these instances from the training dataset of SNLI/MNLI using the following two strategies:

Random: Here, we randomly select instances from the corresponding training dataset. Further fine-tuning our NPH model with the selected instances consistently achieves higher performance than the models fine-tuned from scratch as shown in Table 66. **With just 500 SNLI instances i.e. ~0.1% of training dataset, our models achieve 8.4% and 8.32% higher accuracy on SNLI (in-domain) and MNLI (out-of-domain) respectively.** Furthermore, with 500 MNLI instances, our models achieve 10.37% and 18.07% higher accuracy on MNLI (in-domain) and SNLI (out-of-domain) respectively.

Adversarial: Here, we select those instances from the training dataset on which the NPH model makes incorrect prediction. This is similar to the adversarial data collection strategy (Nie et al. 2020; Kiela et al. 2021) where instances that fool the model are collected. Here, we do not simply fine-tune our NPH model with the adversarial examples as it would lead to catastrophic forgetting (Carpenter and Grossberg 1988). We tackle this by including 20000 randomly sampled instances from the generated PHL triplets and fine-tune on the combined dataset. **It further takes**

Approach	**Δ Accuracy**
NPH model	64.8%
- CV	−5.88%
- CW	−3.07%
- SSNCV	−2.63%
- Neg.	−0.70%
- IrH	−0.50%
- PS	−0.00%

Table 67. Ablation Study of transformations in the NPH-Setting. Each row corresponds to the drop in performance on the SNLI dataset when trained without PHL triplets created using that transformation.

the performance to 70.85%**,** 58.46% **on SNLI and MNLI respectively with** 500 **instances.**

10.5.4 Analysis

Ablation Study:

We conduct ablation study to understand the contribution of individual transformations on NLI performance. Table 67 shows the performance drop observed on removing PHL triplets created using a single transformation in the NPH-Setting. We find that **Contradictory Words (CW) and Contradictory Verbs (CV) lead to the maximum drop in performance,** 5.88% **and** 3.07% **respectively.** In contrast, Pronoun Substitution (PS) transformation doesn't impact the performance significantly. Note that this does not imply that this transformation is not effective, it means that the evaluation dataset (SNLI) does not contain instances requiring this transformation.

Setting	Metric	Label		
		C	**E**	**N**
NPH	Precision	0.65	0.71	0.6
	Recall	0.68	0.77	0.51
P	Precision	0.66	0.72	0.58
	Recall	0.67	0.78	0.52
PH	Precision	0.64	0.74	0.60
	Recall	0.73	0.77	0.50

Table 68. Precision and Recall values achieved by our models under each unsupervised setting.

NC	RS	SNLI-RS	SNLI-NC
84.22	50.07	58.59	75.39

Table 69. Performance of our NPH model on Names-Changed (NC) and Roles-Switched (RS) adversarial test sets Mitra, Shrivastava, and Baral 2020.

NC and RS Evaluation:

We evaluate our model on NER-Changed (NC) and Roles-Switched (RS) datasets presented in (Mitra, Shrivastava, and Baral 2020) that test the ability to distinguish entities and roles. **Our model achieves high performance on these datasets**. Specifically, 84.22% on NC and 75.39% on SNLI-NC as shown in Table 69.

Label-Specific Analysis: Table 68 shows the precision and recall values achieved by our models. We observe that our models perform better on Entailment and Contradiction than Neutral examples. This suggests that **neutral examples are relatively more difficult.** Table 70 shows examples of mis-labeled instances generated by our transformations.

Trans.	Premise	Hypothesis	Assigned Label	True Label
PS	Two dogs on leashes sniffing each other as people walk in a outdoor market	Two dogs on leashes sniffing each other as they walk in a market	E	N
CT	Adult woman eating slice of pizza while standing next to building	There are 2 humans present	E	C
CW	Meal with meat and vegetables served on table	There is a meal with cheese and vegetables	C	N
SSNCV	A person riding skis down a snowy slope	A person riding skis in a body of water	N	C
SSNCV	A person on a skateboard jumping up into the air	A person jumping up in the air on a snowboard	N	C
CV	A male surfer riding a wave on the ocean	A surfer is surfing in the ocean near some swimmers	C	N

Table 70. Examples of mis-labeled PHL triplets generated by our transformations.

10.6 Conclusion and Discussion

We explored three different settings in unsupervised NLI and proposed a procedural data generation approach that outperformed the existing unsupervised methods by ~13%. Then, we showed that fine-tuning our models with a few human-authored instances leads to a considerable improvement in performance. We also experimented using adversarial instances for this fine-tuning step instead of randomly selected instances and showed that it further improves the performance. Specifically, in presence of just 500 adversarial instances, the proposed method achieved 70.85%

accuracy on SNLI, 12.2% higher than the model trained from scratch on the same 500 instances.

This improvement in performance suggests possibility of an alternative data collection strategy that not only results in high-quality data instances but is also resource efficient. Using a model-in-the-loop technique has been shown to be effective for adversarial data collection (Nie et al. 2020; Kiela et al. 2021; Linjie Li et al. 2021; Sheng et al. 2021; Arunkumar et al. 2020). In these techniques, a model is first trained on a large dataset and then humans are instructed to create adversarial samples that fool the model into making incorrect predictions. Thus, requiring the crowd-sourcing effort twice. However, in our method, a dataset designer can develop a set of simple functions (or transformations) to procedurally generate training data for the model and can directly instruct humans to create adversarial samples to fool the trained model. This is resource efficient and allows dataset designers to control the quality of their dataset.

Chapter 11

ACHIEVING EVALUATION EFFICIENCY VIA INSTANCE-LEVEL DIFFICULTY ANALYSIS OF EVALUATION DATA

Knowledge of difficulty level of questions helps a teacher in several ways, such as estimating students' potential quickly by asking carefully selected questions and improving quality of examination by modifying trivial and hard questions. Can we extract such benefits of instance difficulty in Natural Language Processing? To this end, we conduct **I**nstance-**L**evel **D**ifficulty **A**nalysis of **E**valuation data (ILDAE) in a large-scale setup of 23 datasets and demonstrate its five novel applications: 1) *conducting efficient-yet-accurate evaluations* with fewer instances saving computational cost and time, 2) *improving quality of existing evaluation datasets* by repairing erroneous and trivial instances, 3) *selecting the best model* based on application requirements, 4) analyzing dataset characteristics for *guiding future data creation*, 5) *estimating Out-of-Domain performance reliably.* Comprehensive experiments for these applications lead to several interesting results, such as evaluation using just 5% instances (selected via ILDAE) achieves as high as 0.93 Kendall correlation with evaluation using complete dataset and computing weighted accuracy using difficulty scores leads to 5.2% higher correlation with Out-of-Domain performance.

11.1 Introduction

Transformer-based language models (Devlin et al. 2019; Y. Liu et al. 2019a; K. Clark et al. 2020) have improved state-of-the-art performance on numerous natural language processing benchmarks (A. Wang et al. 2018; A. Wang et al. 2019; Talmor et al. 2019); however, recent studies (Zhong et al. 2021; Sagawa et al. 2020) have raised questions regarding whether these models are uniformly better across all instances. This has drawn attention towards instance-level analysis of evaluation data (Rodriguez et al. 2021; Vania et al. 2021; Mishra and Arunkumar 2021) which was previously limited to training data (Swayamdipta et al. 2020; B. Xu et al. 2020; Mishra and Sachdeva 2020). Furthermore, it is intuitive that *not all instances in a dataset are equally difficult.* However, instance-level difficulty analysis of evaluation data (ILDAE) has remained underexplored in many different ways: what are the potential applications and broad impact associated with ILDAE?

In this work, we address the above question by first computing difficulty scores of evaluation instances (section 11.2) and then demonstrating five novel applications of ILDAE (Figure 60).

1. **Efficient Evaluations:** We propose an approach of conducting efficient-yet-accurate evaluations. *Our approach uses as little as* 5% *evaluation instances (selected via ILDAE) to achieve up to* 0.93 *Kendall correlation with evaluations conducted using the complete dataset.* Thus, without considerably impacting the effectiveness of evaluations, our approach saves computational cost and time.
2. **Improving Evaluation Datasets:** We first show that 'trivial' and 'erroneous'

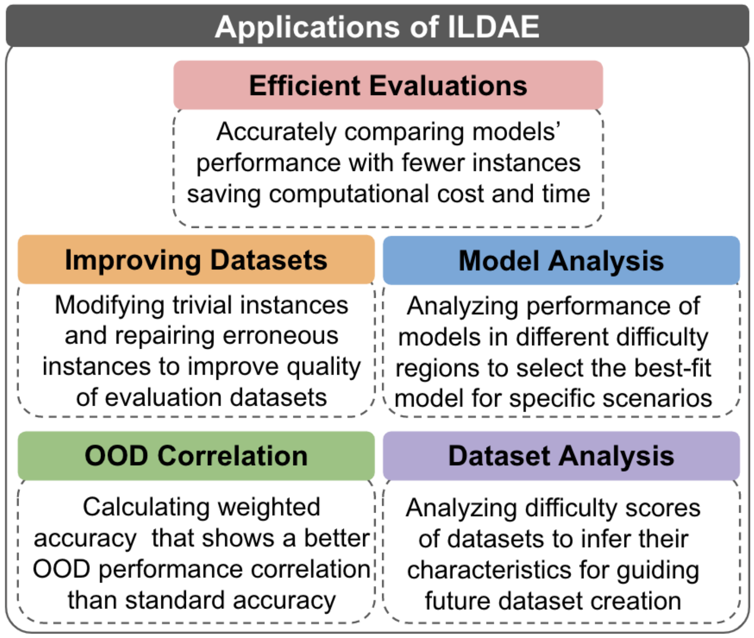


Figure 60. Illustrating five applications of Instance-Level Difficulty Analysis of Evaluation data (ILDAE).

instances can be identified using our difficulty scores and then present a model-and-human-in-the-loop technique to modify/repair such instances resulting in improved quality of the datasets. We instantiate it with SNLI dataset Bowman et al. 2015 and show that *on modifying the trivial instances, the accuracy (averaged over* 27 *models) drops from 77.58% to 26.49%, and on repairing the erroneous instances, it increases from 13.65% to 69.9%.* Thus, improving the dataset quality.

3. **Model Analysis:** We divide evaluation instances into different regions based on difficulty scores and analyze models' performance in each region. We find that *a single model does not achieve the highest accuracy in all difficulty regions.* This implies that the model that achieves best overall performance may not be the best in each difficulty region. Such analyses could benefit in model selection. For

instance, in scenarios where a system is expected to encounter hard instances, the model that performs well in high difficulty regions could be selected.

4. **Dataset Analysis:** ILDAE reveals several important characteristics of datasets that can be leveraged in future data creation processes. For instance, we find that *in SNLI and MNLI datasets, 'contradiction' instances receive lower average difficulty score than 'entailment' and 'neutral' instances.* Thus, more difficult contradiction examples can be created to develop high-quality task-specific datasets.

5. **OOD Correlation:** We compute weighted accuracy leveraging the difficulty scores and show that *it leads to* 5.2% *higher Kendall correlation with Out-of-Domain (OOD) performance than the standard accuracy that treats all instances equally.* Thus, ILDAE helps in getting a more reliable estimation of models' OOD performance.

11.2 Difficulty Score Computation

11.2.1 Desiderata for Difficulty Scores

Interpretation: Human perception of difficulty may not always correlate well with machine's interpretation. Thus, difficulty scores must be computed via a model-in-the-loop technique so that they directly reflect machine's interpretation.

Relationship with Predictive Correctness: Difficulty scores must be negatively correlated with predictive correctness since a difficult instance is less likely to be predicted correctly than a relatively easier instance.

11.2.2 Method

Algorithm 1 Difficulty Score Computation

Input: T: Training Data, M: Model,
D: Evaluation Data E: Training Epochs
Output: Difficulty Score of each instance in D
Auxiliary Function: `GET_CKPTS` (t_r, m, e) - Returns checkpoints on training model m with data t_r for e epochs
Initialization: $Models \leftarrow \varnothing$: List to store ensemble of models trained with different configurations
▷ **Train with Partial Data**
for each $pct \in [100, 50, 25, 20, 15, 10, 5]$ **do**
  T_p = `Sample`(T, pct)
  $Models$ += `Get_Ckpts(`T_p, M, E`)`
end for each
▷ **Train with Corrupted Data**
for each $pct \in [25, 20, 10, 5, 2]$ **do**
  T_c = `Corrupt`(T, pct)
  $Models$ += `Get_Ckpts(`T_c, M, E`)`
end for each
▷ **Infer D using all $Models$ and compute difficulty score d_i for each instance $i \in D$**
for each $i \in D$ **do**
  $d_i = 1 - \frac{\sum_{m \in Models} c_{mi}}{|Models|}$
    ▷ where c_{mi} is the confidence assigned to the ground truth answer by model m
end for each
return d

We incorporate the above desiderata and consider model's prediction confidence in the ground truth answer (indicated by softmax probability assigned to that answer) as the measure of its predictive correctness. Furthermore, we compile an ensemble of models trained with varying configurations and use their mean predictive correctness to compute the difficulty scores. We do this because model's predictions fluctuate greatly when its training configuration is changed (X. Zhou et al. 2020; McCoy, Min, and Linzen 2020) and relying on predictive correctness of only one model could result in difficulty scores that show poor generalization. To this end, we use the following three training configurations to compile predictions from an ensemble of models:

Data Size: *Instances that can be answered correctly even with few training examples are inherently easy and should receive lower difficulty score than the ones that require a large training dataset.* To achieve this, we train a model each with 5, 10, 15, 20, 25, 50, and 100 % of the total training examples and include them in our ensemble.

Data Corruption: *Instances that can be answered correctly even with some level of corruption/noise in the training dataset should receive low difficulty score.* To achieve this, we train a model each with different levels of noise (2, 5, 10, 20, 25% of the examples) in the training data, and add them to our ensemble. For creating noisy examples, we randomly change the ground-truth label in case of classification and multiple-choice datasets and change the answer span for extractive QA datasets.

Training Steps: *Instances that can be consistently answered correctly from the early stages of training should receive low difficulty score.* Here, we add a model checkpoint after every epoch during training to our ensemble.

This results in a total of $N = E * (7 + 5)$ models in our ensemble where E corresponds to the number of training epochs, and 7, 5 correspond to the number of data size and data corruption configurations respectively. We infer the evaluation dataset using these N models and calculate the average predictive correctness for each instance. Finally, we compute the difficulty score by subtracting this averaged correctness value from 1. This ensures that an instance that is answered correctly with high confidence under many training configurations gets assigned a low difficulty score as it corresponds to an easy instance. In contrast, an instance that is often answered incorrectly gets assigned a high difficulty score. Algorithm 1 summarizes this approach.

We use RoBERTa-large model (Y. Liu et al. 2019a) for this procedure and train each model for $E = 10$ epochs, resulting in $N = 120$ predictions for each evaluation instance. *Our difficulty computation method is general and can be used with any other model or configurations; we use RoBERTa-large as it has been shown to achieve high performance across diverse NLP tasks* (Y. Liu et al. 2019a). In addition, we show that difficulty scores computed using our procedure also generalize for other models (11.3.5.1).

We note that difficulty computation is not our primary contribution. Prior work (Swayamdipta et al. 2020; B. Xu et al. 2020) has explored different ways to achieve this. However, our approach uses 120 predictions from models trained with different configurations for its computation and hence is more reliable. Equipped with difficulty scores of evaluation instances, we now demonstrate five applications of ILDAE in the following sections.

11.3 Efficient Evaluations

11.3.1 Problem Statement

Success of BERT (Devlin et al. 2019) has fostered development of several other pre-trained language models such as RoBERTa (Y. Liu et al. 2019a), XLNet (Z. Yang et al. 2019b), DistilBERT (Sanh et al. 2019), ALBERT (Lan et al. 2020). Though, it has resulted in the availability of numerous model options for a task, comparing the performance of such a large number of models has become computationally expensive and time-consuming. For example, in real-world applications like online competitions, the naive approach that evaluates candidate models on the entire test dataset would be too expensive because they receive thousands of model submissions and contain a sizable number of evaluation instances. Moreover, some applications also require additional evaluations to measure Out-of-Domain generalization and robustness making it even more expensive. *Can we make the evaluations* ***efficient****?*

11.3.2 Solution

We address the above question and explore if the performance of candidate models can be accurately compared with a carefully selected smaller subset of the evaluation dataset. Reducing the number of instances would save computational cost and make the evaluations efficient. To this end, we propose an approach that selects evaluation instances based on their difficulty scores. We compare performance of candidate models only on these selected instances and show that without considerably impacting the result of evaluations, our approach saves computational cost and time.

Instance Selection: We argue that *the instances with extreme difficulty scores (very low and very high scores) would not be very effective in distinguishing between the candidate models.* This is because the former instances are trivial and would be answered correctly by many/all candidate models, while the latter ones are hard and would be answered correctly by only a few/none models. Therefore, given a budget on the number of evaluation instances, we select a majority of them with moderate difficulty scores. However, to distinguish amongst very weak and amongst very strong candidates, we also include a small number of instances with extreme difficulty scores. Figure 61 illustrates our approach.

Note that our approach does not add any computational overhead during evaluations as the difficulty scores are pre-computed. Furthermore, *we do not compute separate difficulty scores for each candidate model as it would defy the sole purpose of 'efficient' evaluations.* Instead, we compute difficulty scores using only one model (RoBERTa-large) and exclude it from the list of candidate models for a fair evaluation of our approach. For our instance selection approach to work in this setting, the difficulty scores should generalize for other models. We empirically prove this generalization capability and demonstrate the efficacy of our efficient evaluations approach in 11.3.5.

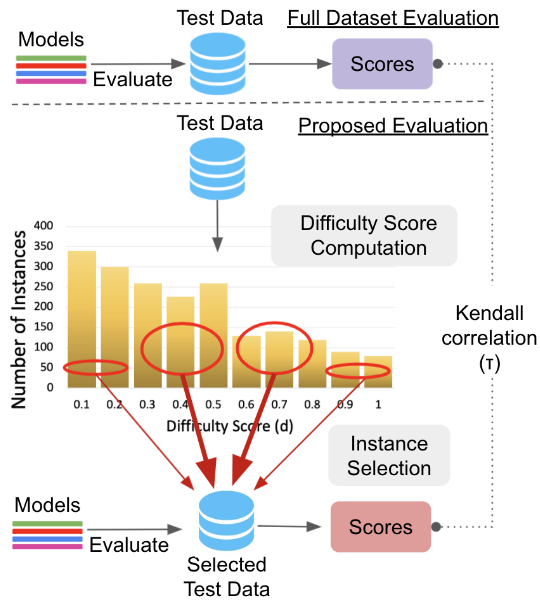


Figure 61. Comparing standard evaluation approach (top) with our proposed 'efficient' approach (bottom). We leverage difficulty scores to select a small subset of evaluation instances on which the performance of models can be efficiently compared. Our selected subset contains a majority of the instances with moderate difficulty scores and only a few with extreme difficulty scores. We use Kendall correlation between the performance scores to measure the efficacy of our approach.

11.3.3 Experimental Details

Performance Metric: We measure the efficacy of an instance selection technique by computing accuracies of candidate models on the selected instances and calculating their Kendall's correlation (Kendall 1938) with accuracies obtained on the full evaluation dataset. High correlation implies that the performance scores obtained using the selected instances display the same behavior as the performance scores obtained using the complete dataset. Hence, high correlations values are preferred.

Datasets: We experiment with a total of 23 datasets across Natural Language Inference, Duplicate Detection, Sentiment Analysis, Question Answering, Commonsense

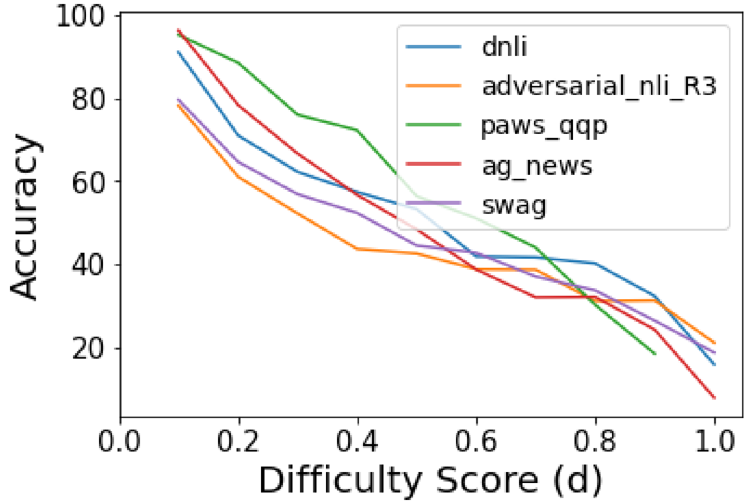


Figure 62. Demonstrating difficulty score generalization. Difficulty scores computed using RoBERTa-large show negative correlation with accuracy averaged over 27 other models, hence satisfying the desiderata mentioned in Section 11.2.1. Note that we depict this trend for a few datasets only to avoid cluttering the image. Similar trend is observed for other dataset also.

Reasoning, and several other tasks. We experiment with the following datasets: SNLI (Bowman et al. 2015), Multi-NLI (Williams, Nangia, and Bowman 2018), Dialogue NLI (Welleck et al. 2019), Adversarial NLI (R1, R2, R3) (Nie et al. 2020), QNLI (A. Wang et al. 2018), QQP (Iyer, Dandekar, and Csernai 2017), MRPC (Dolan and Brockett 2005), PAWS-QQP, PAWS-Wiki (Zhang, Baldridge, and He 2019), SST-2 (Socher et al. 2013), COLA (Warstadt, Singh, and Bowman 2019) AG's News (Zhang, Zhao, and LeCun 2015), ARC-Easy, ARC-Challenge (P. Clark et al. 2018), SWAG (Zellers et al. 2018), Abductive-NLI (Bhagavatula et al. 2020), Winogrande (Sakaguchi et al. 2020), CommonsenseQA (Talmor et al. 2019), QuaRel (Tafjord, Clark, et al. 2019), QuaRTz (Tafjord, Gardner, et al. 2019), and SocialIQA (Sap et al. 2019).

Candidate Models: We use BERT (Devlin et al. 2019), DistilBERT (Sanh

% Instances →	0.5%			1%			2%	5%	10%	20%
Dataset ↓	**Random**	**Heuristic**	**Proposed**	**Random**	**Heuristic**	**Proposed**	**Proposed**	**Proposed**	**Proposed**	**Proposed**
SNLI	$0.55_{0.09}$	$0.38_{0.17}$	$\mathbf{0.68}_{0.13}$	$0.68_{0.05}$	$0.58_{0.08}$	$\mathbf{0.78}_{0.08}$	$0.83_{0.04}$	$0.88_{0.04}$	$0.91_{0.01}$	$0.93_{0.02}$
PAWS Wiki	$0.67_{0.07}$	$0.68_{0.04}$	$\mathbf{0.78}_{0.06}$	$0.73_{0.05}$	$0.78_{0.02}$	$\mathbf{0.86}_{0.05}$	$0.89_{0.02}$	$0.91_{0.03}$	$0.95_{0.01}$	$0.96_{0.01}$
AgNews	$0.12_{0.26}$	$0.14_{0.27}$	$\mathbf{0.47}_{0.05}$	$0.25_{0.34}$	$0.41_{0.14}$	$\mathbf{0.52}_{0.1}$	$0.65_{0.07}$	$0.75_{0.06}$	$0.8_{0.04}$	$0.89_{0.03}$
QNLI	$0.41_{0.1}$	$0.44_{0.04}$	$\mathbf{0.48}_{0.13}$	$0.57_{0.04}$	$0.55_{0.1}$	$\mathbf{0.57}_{0.07}$	$0.7_{0.06}$	$0.78_{0.06}$	$0.85_{0.03}$	$0.91_{0.03}$
MRPC	$0.04_{0.09}$	$-0.03_{0.18}$	$\mathbf{0.21}_{0.16}$	$-0.02_{0.09}$	$0.05_{0.2}$	$\mathbf{0.29}_{0.21}$	$0.36_{0.15}$	$0.45_{0.08}$	$0.58_{0.12}$	$0.65_{0.14}$
SocialIQA	$0.19_{0.09}$	$0.15_{0.29}$	$\mathbf{0.37}_{0.17}$	$0.34_{0.07}$	$0.28_{0.21}$	$\mathbf{0.4}_{0.09}$	$0.58_{0.1}$	$0.67_{0.04}$	$0.75_{0.08}$	$0.81_{0.05}$
QQP	$0.63_{0.06}$	$0.64_{0.05}$	$\mathbf{0.65}_{0.05}$	$0.74_{0.03}$	$0.74_{0.01}$	$\mathbf{0.77}_{0.06}$	$0.84_{0.04}$	$0.9_{0.04}$	$0.94_{0.04}$	$0.95_{0.01}$
DNLI	$0.58_{0.05}$	$0.59_{0.1}$	$\mathbf{0.58}_{0.11}$	$0.68_{0.1}$	$0.71_{0.04}$	$\mathbf{0.76}_{0.07}$	$0.84_{0.04}$	$0.92_{0.05}$	$0.94_{0.03}$	$0.96_{0.01}$
COLA	–	–	–	$-0.01_{0.18}$	$\mathbf{0.25}_{0.26}$	$0.24_{0.45}$	$0.41_{0.41}$	$0.63_{0.23}$	$0.75_{0.08}$	$0.78_{0.02}$
SWAG	$0.72_{0.04}$	$0.66_{0.02}$	$\mathbf{0.75}_{0.06}$	$0.79_{0.03}$	$0.77_{0.03}$	$\mathbf{0.78}_{0.05}$	$0.86_{0.03}$	$0.89_{0.02}$	$0.93_{0.01}$	$0.95_{0.01}$
PAWS QQP	–	–	–	$0.13_{0.24}$	$0.36_{0.05}$	$\mathbf{0.34}_{0.13}$	$0.55_{0.19}$	$0.8_{0.05}$	$0.84_{0.03}$	$0.87_{0.04}$
MNLI	$0.7_{0.04}$	$0.71_{0.03}$	$\mathbf{0.73}_{0.07}$	$0.8_{0.02}$	$0.8_{0.04}$	$\mathbf{0.82}_{0.08}$	$0.89_{0.03}$	$0.93_{0.02}$	$0.95_{0.02}$	$0.96_{0.01}$
Adv. NLI R1	$0.0_{0.08}$	$-0.07_{0.06}$	$\mathbf{0.17}_{0.27}$	$0.02_{0.13}$	$\mathbf{0.09}_{0.11}$	$0.08_{0.2}$	$0.13_{0.18}$	$0.3_{0.18}$	$0.47_{0.05}$	$0.59_{0.05}$
Adv. NLI R2	$-0.08_{0.04}$	$\mathbf{-0.01}_{0.06}$	$-0.08_{0.16}$	$-0.08_{0.07}$	$\mathbf{0.02}_{0.03}$	$-0.03_{0.21}$	$0.0_{0.12}$	$0.17_{0.03}$	$0.26_{0.11}$	$0.42_{0.15}$
Adv. NLI R3	$-0.15_{0.12}$	$\mathbf{0.15}_{0.1}$	$0.1_{0.21}$	$-0.03_{0.06}$	$0.07_{0.1}$	$\mathbf{0.1}_{0.11}$	$0.18_{0.16}$	$0.12_{0.17}$	$0.31_{0.15}$	$0.58_{0.05}$
SST-2	–	–	–	$0.08_{0.15}$	$0.16_{0.35}$	$\mathbf{0.29}_{0.25}$	$0.4_{0.2}$	$0.52_{0.16}$	$0.65_{0.13}$	$0.81_{0.08}$
ARC Easy	–	–	–	$0.0_{0.2}$	$-0.03_{0.12}$	$\mathbf{0.42}_{0.19}$	$0.47_{0.19}$	$0.59_{0.13}$	$0.6_{0.14}$	$0.74_{0.11}$
ARC Diff	–	–	–	–	–	–	$0.15_{0.29}$	$0.28_{0.13}$	$0.33_{0.31}$	$0.3_{0.26}$
Abductive NLI	$0.08_{0.26}$	$\mathbf{0.17}_{0.05}$	$0.16_{0.09}$	$0.19_{0.19}$	$0.26_{0.08}$	$\mathbf{0.3}_{0.07}$	$0.42_{0.13}$	$0.57_{0.08}$	$0.61_{0.07}$	$0.68_{0.07}$
Winogrande	$-0.19_{0.11}$	$-0.03_{0.06}$	$\mathbf{0.0}_{0.17}$	$-0.11_{0.09}$	$-0.05_{0.12}$	$\mathbf{0.11}_{0.15}$	$0.09_{0.14}$	$0.03_{0.1}$	$0.14_{0.1}$	$0.21_{0.14}$
CSQA	$0.29_{0.11}$	$0.28_{0.1}$	$\mathbf{0.31}_{0.07}$	$0.36_{0.14}$	$0.37_{0.08}$	$\mathbf{0.39}_{0.09}$	$0.49_{0.09}$	$0.69_{0.08}$	$0.78_{0.04}$	$0.83_{0.05}$
QuaRel	–	–	–	–	–	–	$0.32_{0.26}$	$0.33_{0.25}$	$0.39_{0.07}$	$0.51_{0.1}$
QuaRTz	–	–	–	–	–	–	$0.34_{0.19}$	$0.36_{0.04}$	$0.34_{0.12}$	$0.37_{0.08}$
Average	$0.28_{0.1}$	$0.3_{0.11}$	$\mathbf{0.39}_{0.13}$	$0.31_{0.11}$	$0.35_{0.11}$	$\mathbf{0.43}_{0.14}$	$0.46_{0.17}$	$0.58_{0.11}$	$0.66_{0.08}$	$0.72_{0.07}$

Table 71. Kendall correlation with full evaluation dataset achieved by various instance selection approaches for different percentage of instances. Each cell shows the mean and standard deviation obtained from 5 different runs. – cell indicates 0 selected instances. We show the expanded version of this table in supplementary.

et al. 2019), ConvBERT (Z.-H. Jiang et al. 2020) , XLNET (Z. Yang et al. 2019a), SqueezeBERT (Iandola et al. 2020), ELECTRA (K. Clark et al. 2020) in our experiments. We also use different variants of ConvBert (small, medium-small, base) and ELECTRA (small, base) models. For comprehensive experiments, we train each of the above models with training data of three different sizes ($2k$, $5k$, and $10k$ examples) resulting in 27 candidate models for each dataset. We intentionally exclude RoBERTa from this list as we use it for computing the difficulty scores.

Instance Selection Baselines: We compare the proposed instance selection approach with the following baselines:

Random Selection: Select a random subset of instances from the evaluation dataset.

Heuristic Selection: Select instances based on the length heuristic (number of characters in the instance text) instead of the difficulty scores.

11.3.4 Related Work

Adaptive evaluation (Weiss 1982) is used in educational settings for evaluating performance of students. It uses Item Response Theory (IRT) (Baker and Kim 2004) from psychometrics that requires a large number of subjects and items to estimate system parameters (Lalor, Wu, and Yu 2016; Lalor et al. 2018). Moreover, adaptive evaluation is computationally very expensive as it requires calculating performance after each response to select the next instance based on the previous responses of the subject. Thus, it is not fit for our setting as we intend to improve the computational efficiency. In contrast, our approach is much simpler and efficient as it does not incur any additional cost during the evaluation.

11.3.5 Results

We first study generalization of our computed difficulty scores and then show the efficacy of the proposed instance selection approach in conducting efficient evaluations.

11.3.5.1 Generalization of Difficulty Scores:

In Figure 62, we plot accuracy (averaged over all 27 candidate models) against difficulty scores (computed using RoBERTa-large). We find that with the increase in difficulty score, the accuracy consistently decreases for all datasets. We also study this behavior for each individual candidate model and find results supporting the above observation (Figure 65). This proves that the difficulty scores follow the desiderata mentioned in Section 11.2.1 for other models also and our intuitions behind instance selection for conducting efficient evaluations hold true. Note that these difficulty scores are computed using a specific model but our approach is general and will replicate this generalization capability if used with any other model.

11.3.5.2 Efficient Evaluations:

Table 71 shows Kendall correlation with full dataset evaluation achieved by various instance selection approaches for different percentages of instances.

Proposed Approach Outperforms Baselines: Our proposed approach is consistently better than the Random and Heuristic approaches. For instance, with just 0.5% and 1% evaluation instances, our approach outperforms the baseline methods by ~ 30% and ~ 22.8% respectively.

Correlation Change with % of Instances: As expected, Kendall correlation consistently increases as a higher percentage of instances are selected for evaluation. In case of SNLI, PAWS Wiki, QQP, DNLI, SWAG, and MNLI, just 2% instances are sufficient to achieve correlation of > 0.8. For most datasets, with just 20% of the evaluation instances, our approach achieves Kendall correlation of > 0.8. This suggests that the evaluations can be conducted with fewer instances without significantly compromising the accuracy of comparison. We further analyze performance of our approach for higher percentage of instances in Table 77.

Thus, for practical settings where candidate models can't be compared on the entire dataset due to computational and time constraints, evaluating only on the selected instances can result in fairly accurate performance comparison.

Performance on Multiple-Choice QA datasets: Though, we perform better than the baselines approaches on almost all datasets, we achieve a lower correlation value for multiple-choice question answering datasets such as QuaRel, QuaRTz, and Winogrande. We attribute this behavior to the close scores (accuracies) achieved by many candidate models even in case of full dataset evaluation. Thus, it is difficult to differentiate such models as they achieve nearly the same performance. Furthermore, in some difficult datasets such as Adversarial NLI (R1, R2, and R3), ARC Difficult, and Winogrande, many candidate models achieve accuracies very close to the random baseline (33% for NLI, 50% for Winogrande). So, comparing their performance even with full dataset does not provide any significant insights.

Difficult Instance
Premise: Dog standing with 1 foot up in a large field. **Hyp.**: The dog is standing on one leg. **Label:** Contradiction.
Premise: A salt-and-pepper-haired man with beard and glasses wearing black sits on the grass. **Hyp.**: An elderly bearded man sitting on the grass. **Label:** Entailment.
Premise: A man is standing in front of a building holding heart shaped balloons and a woman is crossing the street. **Hyp.**: Someone is holding something heavy outside. **Label:** Contradiction.
Premise: A group of people plays a game on the floor of a living room while a TV plays in the background. **Hyp.**: A group of friends are playing the xbox while other friends wait for their turn. **Label:** Contradiction.

Table 72. Illustrative examples of instances that receive high difficulty score but are not erroneous. Such instances are difficult even for humans as they require reasoning ability.

11.4 Improving Evaluation Datasets

11.4.1 Problem Statement

Recent years have seen a rapid increase in the number and size of NLP datasets. Crowd-sourcing is a prominent way of collecting these datasets. Prior work (Gururangan et al. 2018; Tan et al. 2019; Mishra, Arunkumar, Sachdeva, et al. 2020) has shown that crowd-sourced datasets can contain: (a) *erroneous instances* that have annotation mistakes or ambiguity, (b) too many *trivial instances* that are very easy to answer. This hampers the quality of the dataset and makes it less reliable for drawing conclusions. *Can difficulty scores aid in improving the quality of evaluation datasets?*

Dataset	Instance
SNLI (72%)	**Premise**: Trucks racing. **Hypothesis**: Four trucks are racing against each other in the relay. **Label:** Entailment, Neutral
CSQA (50%)	Why would a band be performing when there are no people nearby? **O1**: record album, **O2**: play music, **O3**: hold concert, **O4**: blaring, **O5**: practice
WG (36%)	Maria was able to keep their weight off long term, unlike Felicia, because _ followed a healthy diet. **O1**: Maria, **O2**: Felicia
aNLI	**O1**: Ella was taking her final exam. **O2**: Ella was able to finish her exam on time. **H1**: Ella got to class early and was in no hurry. **H2**: Ella broke her pencil.

Table 73. Examples of erroneous instances from SNLI, CSQA, Winogrande, and Abductive NLI. Orange (ambiguous) and red (mislabeled) correspond to the originally annotated answer while blue corresponds to the correct/equally probable answer.

11.4.2 Solution

We first show that erroneous and trivial instances can be identified using the difficulty scores and then present a human-and-model-in-the-loop technique to modify/repair such instances resulting in improved quality of the datasets.

Identifying Erroneous and Trivial Instances: We inspect 50 instances each with very high and very low difficulty scores and find that a significant percentage of the former are either mislabeled or contain ambiguity and the latter are too easy to be answered.

Table 73 shows examples of erroneous instances from SNLI, Winogrande, CSQA, and Abductive NLI. We find 72% of the inspected SNLI instances to be erroneous. Furthermore, we find that some high difficulty score instances are actually difficult

Dataset	Instance
SNLI (72%)	**Premise**: Trucks racing. **Hyp.**: Four trucks are racing against each other in the relay. Entailment, Neutral **Premise**: Two elderly men having a conversation. **Hyp.**: Two elderly woman having a conversation with their children. Neutral, Contradiction
CSQA (50%)	Why would a band be performing when there are no people nearby? **O1**: record album, **O2**: play music, **O3**: hold concert, **O4**: blaring, **O5**: practice What do audiences clap for? **O1**: cinema, **O2**: theatre, **O3**: movies, **O4**: show, **O5**: hockey game
WG (36%)	Maria was able to keep their weight off long term, unlike Felicia, because _ followed a healthy diet. **O1**: Maria, **O2**: Felicia When Derrick told Christopher about quitting school to provide for their family, _ started panicking. **O1**: Derrick, **O2**: Christopher
aNLI	**O1**: Ella was taking her final exam. **O2**: Ella was able to finish her exam on time. **H1**: Ella got to class early and was in no hurry. **H2**: Ella broke her pencil. **O1**: Cathy was happy that she finally had some time to sew. **O2**: Cathy tapped her metal fingertips on the table in frustration. **H1**: Cathy put the thimbles on. **H2**: Cathy could not get the thread into the fabric.

Table 74. Illustrative examples of erroneous instances in SNLI, CSQA, Winogrande, and Abductive NLI. Orange (ambiguous) and red (mislabeled) indicate the originally annotated answer while blue indicates the True/equally probable answer.

even for humans because they require abilities such as commonsense reasoning. Table 72 shows such instances. We also provide examples of trivial instances (Table 75) and note that such instances are trivial from model's perspective as they can be answered correctly (with high confidence) by simply latching on to some statistical cues present in the training data.

Technique: Since the trivial instances are too easy to be answered, we propose

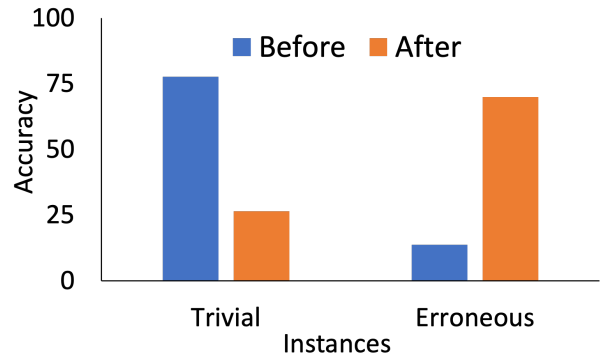


Figure 63. Comparing accuracy (averaged over 27 models) before and after modifying the SNLI instances using our model-and-human-in-the-loop technique. The accuracy on trivial instances decreases as we make them more difficult while the accuracy on erroneous instances increases as we repair them.

to modify them in an *adversarial* way such that they no longer remain trivial. Specifically, we include a human-in-the-loop who needs to modify a trivial instance in a label-preserving manner such that the modified version fools the model into making an incorrect prediction. For adversarial attack, we use the strongest model from our ensemble of 120 models. It has two key differences with the standard adversarial data creation approach presented in (Nie et al. 2020; Kiela et al. 2021): (a) it requires modifying an already existing instance instead of creating a new instance from scratch. (b) it does not increase the size of the evaluation dataset as we replace an already saturated instance (trivial) with its improved not-trivial version. We use a human instead of leveraging automated ways to modify the trivial instances because our objective is to improve the quality of instances and prior work has shown that these automated techniques often result in unnatural and noisy instances. Therefore, such techniques could be cost-efficient but might not solve the sole purpose of improving quality.

To further improve the quality, we provide instances with very high difficulty score (potentially erroneous) and ask a human to repair them such that the repaired versions follow the task definition. The human can either change the instance text or its answer to achieve the goal. Note that this scenario is model-independent.

Dataset	Instance
SNLI	**Premise**: A woman playing with her cats while taking pictures. **Hyp.**: A woman is playing with her dolls. Contradiction
CSQA	What will a person going for a jog likely be wearing? **O1**: grope, **O2**: acknowledgment, **O3**: comfortable clothes, **O4**: ipod, **O5**: passionate kisses
WG	Katrina did not value the antique pictures as much as Lindsey because _ was a history buff. **O1**: Katrina, **O2**: Lindsey
aNLI	**O1**: I bought a house with an ugly yard. **O2**: He carved the rock into a lion head and kept it. **H1**: There was a large rock in the yard. **H2**: I decided to tear the whole notebook up.

Table 75. Illustrative examples of trivial instances in SNLI, CSQA, Winogrande, and Abductive NLI. Text in blue corresponds to the ground-truth answer.

11.4.3 Results

Table 76 shows original and modified instances from SNLI. Top two examples correspond to the trivial instances where the human modified the hypothesis in a label-preserving manner such that it fooled the model into making incorrect prediction. The bottom two correspond to the mislabeled instances where the human rectified the label. Figure 63 compares the performance of models on the original instances and the their modified/repaired versions. As expected, the performance drops on the previously trivial instances as they are no longer trivial and improves on the previously erroneous instances. We release the improved version of the dataset compiled via our technique.

Original Instance	Modification
P: A man standing in front of a chalkboard points at a drawing. **H**: A kid washes a chalkboard **L:** Contradiction	**H'**: A 4 year old male standing in front of a chalkboard points at a drawing. **Predicted L:** Neutral
P: A man is performing tricks with his superbike. **H**: A bike is in the garage. **L:** Contradiction	**H'**: He is performing stunts on a four wheeler. **Predicted L:** Neutral
P: A skateboarder does a trick at a skate park. **H**: The skateboarder is performing a heelie kick flip. **L:** Entailment	**L'**: Neutral
P: A little blond girl is running near a little blond boy. **H**: A sister and brother are playing in their yard. **L:** Entailment	**L'**: Neutral

Table 76. Illustrative examples from SNLI dataset modified using our technique. Top two correspond to trivial instances for which a human modified the hypothesis in a label-preserving manner such that the model's prediction changed. Bottom two correspond to mislabeled instances where the human rectified the label.

11.5 Other Applications of ILDAE

We now briefly discuss other ILDAE applications.

11.5.1 Dataset Analysis

ILDAE reveals several useful characteristics of datasets such as which class label has the easiest instances. We study this for NLI datasets: SNLI, MNLI, DNLI, and Adversarial NLI (Figure 64). For SNLI and MNLI, we find that the contradiction instances receive lower average difficulty score than entailment and neutral instances.

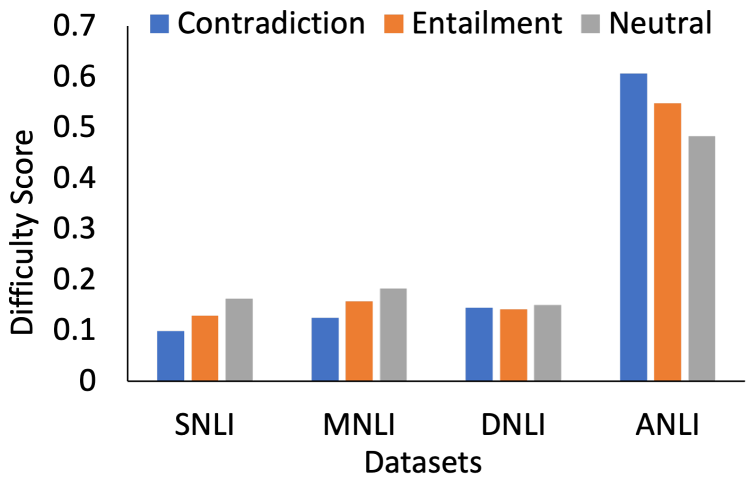


Figure 64. Comparing average difficulty of NLI labels for various datasets.

For Adversarial NLI, the order is reversed. For DNLI, all the labels get assigned nearly the same average difficulty. Such analysis can serve as a guide for future data creation as it indicates for which type of instances more data collection effort needs to be invested. It can also be used to compare average difficulty at dataset level. Furthermore, a new harder task-specific benchmark can be created by combining high difficulty instances from all the datasets of that task.

11.5.2 Model Analysis

We divide the evaluation instances into different regions based on the difficulty scores and analyze models' performance in each region. We find that a single model does not achieve the highest accuracy across all regions. Figure 65 illustrates this pattern for SNLI dataset. This implies that the model that achieves the highest performance on easy instances may not necessarily achieve the highest performance on

Dataset	25% P	30% P	40% P	50% P	60% P	75% P
SNLI	$0.95_{0.0}$	$0.95_{0.01}$	$0.96_{0.01}$	$0.96_{0.01}$	$0.96_{0.01}$	$0.97_{0.01}$
PAWS Wiki	$0.98_{0.01}$	$0.98_{0.02}$	$0.98_{0.01}$	$0.98_{0.01}$	$0.98_{0.01}$	$0.99_{0.01}$
AgNews	$0.93_{0.01}$	$0.93_{0.02}$	$0.93_{0.01}$	$0.96_{0.01}$	$0.96_{0.01}$	$0.97_{0.01}$
QNLI	$0.92_{0.02}$	$0.92_{0.03}$	$0.93_{0.02}$	$0.96_{0.01}$	$0.96_{0.01}$	$0.97_{0.01}$
MRPC	$0.67_{0.13}$	$0.7_{0.11}$	$0.75_{0.08}$	$0.84_{0.05}$	$0.84_{0.03}$	$0.88_{0.03}$
SocialIQA	$0.84_{0.04}$	$0.87_{0.02}$	$0.89_{0.02}$	$0.91_{0.01}$	$0.93_{0.02}$	$0.94_{0.03}$
QQP	$0.96_{0.01}$	$0.96_{0.01}$	$0.96_{0.01}$	$0.97_{0.01}$	$0.98_{0.0}$	$0.99_{0.01}$
DNLI	$0.96_{0.02}$	$0.97_{0.02}$	$0.97_{0.02}$	$0.98_{0.01}$	$0.98_{0.01}$	$0.98_{0.01}$
COLA	$0.8_{0.05}$	$0.82_{0.07}$	$0.89_{0.06}$	$0.91_{0.02}$	$0.92_{0.04}$	$0.96_{0.02}$
SWAG	$0.97_{0.01}$	$0.96_{0.01}$	$0.97_{0.01}$	$0.98_{0.01}$	$0.99_{0.0}$	$0.99_{0.01}$
PAWS QQP	$0.89_{0.02}$	$0.92_{0.02}$	$0.92_{0.02}$	$0.93_{0.02}$	$0.94_{0.01}$	$0.94_{0.02}$
MNLI	$0.95_{0.01}$	$0.97_{0.01}$	$0.97_{0.01}$	$0.98_{0.0}$	$0.97_{0.01}$	$0.98_{0.01}$
Adv. NLI R1	$0.62_{0.06}$	$0.64_{0.08}$	$0.67_{0.06}$	$0.73_{0.06}$	$0.79_{0.05}$	$0.84_{0.07}$
Adv. NLI R2	$0.42_{0.08}$	$0.46_{0.1}$	$0.54_{0.14}$	$0.63_{0.05}$	$0.71_{0.05}$	$0.77_{0.03}$
Adv. NLI R3	$0.61_{0.05}$	$0.59_{0.06}$	$0.66_{0.1}$	$0.75_{0.06}$	$0.79_{0.06}$	$0.85_{0.04}$
SST-2	$0.83_{0.05}$	$0.86_{0.04}$	$0.87_{0.02}$	$0.87_{0.04}$	$0.91_{0.03}$	$0.92_{0.01}$
ARC Easy	$0.76_{0.07}$	$0.78_{0.08}$	$0.84_{0.08}$	$0.85_{0.05}$	$0.89_{0.03}$	$0.94_{0.02}$
ARC Diff	$0.41_{0.36}$	$0.49_{0.32}$	$0.62_{0.28}$	$0.59_{0.18}$	$0.75_{0.1}$	$0.86_{0.06}$
Abductive NLI	$0.72_{0.03}$	$0.77_{0.03}$	$0.79_{0.06}$	$0.82_{0.03}$	$0.86_{0.02}$	$0.88_{0.04}$
Winogrande	$0.24_{0.13}$	$0.3_{0.16}$	$0.39_{0.17}$	$0.44_{0.16}$	$0.53_{0.09}$	$0.63_{0.07}$
CSQA	$0.85_{0.04}$	$0.86_{0.03}$	$0.89_{0.03}$	$0.91_{0.02}$	$0.94_{0.02}$	$0.95_{0.01}$
QuaRel	$0.57_{0.12}$	$0.58_{0.16}$	$0.73_{0.1}$	$0.8_{0.15}$	$0.79_{0.07}$	$0.81_{0.07}$
QuaRTz	$0.37_{0.12}$	$0.44_{0.07}$	$0.51_{0.12}$	$0.57_{0.09}$	$0.62_{0.11}$	$0.64_{0.08}$

Table 77. Kendall correlation with full dataset evaluation achieved by our proposed instance selection approach for different percentage of instances. Each cell shows the mean and standard deviation obtained from 5 different runs.

difficult instances. The similar pattern is observed for other datasets. Such analysis would benefit in model selection. For instance, in scenarios where a system is expected to encounter hard instances, we can select the model that has the highest accuracy on instances of difficult regions. Whereas, for scenarios containing easy instances, the model that has the highest accuracy on instances of easy regions.

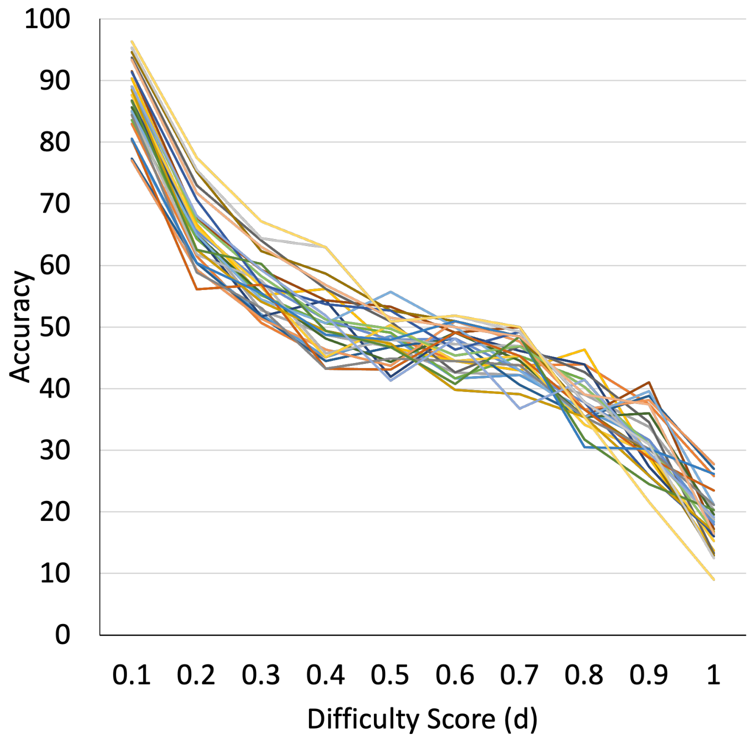


Figure 65. Comparing accuracy of various models in different difficulty regions for SNLI dataset. Each line corresponds to a candidate model (27 in total). It shows that a single model does not achieve the highest accuracy in all difficulty regions.

11.5.3 Correlation with OOD Performance

Large pre-trained language models can achieve high In-Domain performance on numerous tasks. However, it does not correlate well with OOD performance (Hendrycks and Dietterich 2019; Hendrycks et al. 2020). To this end, we present an approach to compute a weighted accuracy that shifts away from treating all the evaluations instances equally and assigns weight based on their difficulty scores. We define the weight w_i of an instance i with difficulty score d_i as:

$$w_i = \frac{1 + \mu * d_i}{N + \mu * \sum_{j=1}^{N} d_j}$$

where N corresponds to the total number of evaluation instances, and μ is a hyper-parameter that controls influence of difficulty score on the weight. Then, weighted accuracy W is simply:

$$W = \sum_{i=1}^{N} w_i * v_i$$

where v_i is 1 when the model's prediction is correct else 0. This implies that high accuracy may not always translate to high weighted accuracy.

We take SNLI as the in-domain dataset and MNLI, DNLI, and HANS (McCoy, Pavlick, and Linzen 2019) (Constituent, Lexical Overlap, Subsequence) as OOD datasets. We calculate unweighted and weighted accuracy of the 27 models (described in Section 11.3.3) and compare their Kendall correlation with the accuracy on OOD datasets. Figure 66 shows this comparison. It can be observed that weighted accuracy shows 5.2% higher correlation with OOD performance that the standard accuracy. Most improvement is observed in hard datasets i.e. HANS. Thus, weighting instances based on their difficulty score is more informative than the standard accuracy that treats all instances equally.

11.6 Conclusion

We conducted Instance-Level Difficulty Analysis of Evaluation data (ILDAE) in a large-scale setup of 23 datasets and presented its five novel applications. With these applications, we demonstrated ILDAE's impact in several important areas,

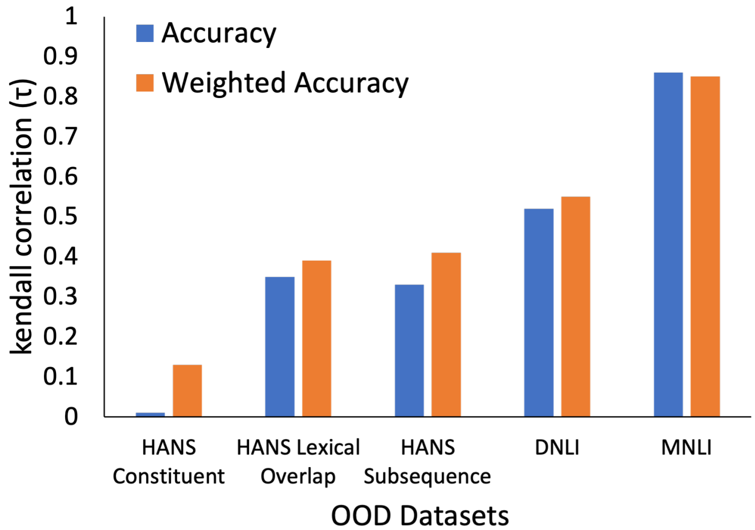


Figure 66. Comparing Kendall correlation of standard unweighted accuracy and weighted accuracy with OOD accuracy. Weighted accuracy achieves 5.2% higher correlation on average.

such as conducting efficient evaluations with fewer instances, improving dataset quality, and estimating out-of-domain performance reliably. We release our computed difficulty scores and hope that our work will encourage research in this important yet understudied field of leveraging instance difficulty in evaluations.

Chapter 12

CONCLUSIONS AND FUTURE WORK

Despite recently developed pre-trained language models achieving impressive performance, there are several challenges pertaining to their practical application, such as they have a large number of parameters which makes their inference slow and computationally expensive and they are not absolutely perfect always i.e. they often make incorrect predictions which hampers their reliability. In this dissertation, I identified and provided effective solutions to several efficiency and reliability related challenges. Specifically, towards improving the reliability of models, I first focused on addressing the hallucination problem which corresponds to the phenomenon when the model outputs factually incorrect text. Then, I focused on another aspect of reliability which is selective prediction. Selective prediction enables the models to abstain from making predictions when the predictions are likely to be incorrect. This prevents the model from making erroneous predictions and thus improves its reliability. On the efficiency front, I covered several crucial aspects such as 'inference efficiency' which focuses on making model inferences in a computationally efficient manner without sacrificing the prediction accuracy, 'open-domain QA reader efficiency' in which I leveraged the external knowledge efficiently while answering open-domain questions, 'LLM decoding efficiency' where I improved the computational efficiency of text generation without sacrificing the quality of the generation, 'data sample efficiency' in which I focused on efficiently collecting data instances for training a task-specific system, and ‘evaluation efficiency’ where I compared the performance of different models efficiently. Overall, with the research mission of developing reliable and efficient natural language processing systems, this dissertation identified several crucial challenges pertinent to the ‘efficiency’ and ‘reliability’ of NLP systems and provided effective solutions to address these two important elements of practical NLP.

We present the key takeaways and future research avenues from different components of this dissertation below:

- **Addressing the Hallucination problem**: LLMs exhibit hallucination in their output which hampers their reliability and trustworthiness. Due to the autoregressive nature of generation of the models, active detection and mitigation during the generation process does not only reduce these hallucinations but also prevents the propagation of hallucinations in the model’s output. Future avenues for research in this direction include studying the active detection and mitigation approach with the multi-modal models and systematically comparing the effectiveness of various no-resource and retrieval augmented methods in a unified testbench.
- **Improving Reliability via Selective Prediction and Post-Abstention**: Model’s are not absolutely perfect, i.e., they often output incorrect predictions which hamper their reliability. Selective prediction with approaches like maximum softmax probability, monte-carlo dropout, label smoothing, and calibration can partly address this problem. This is because selective prediction enables the models to abstain from answering when their prediction is likely to be incorrect. Future avenues for research in this direction include studying post-abstention

methods such as RETоP that leverage the logit outputs to detect uncertainty with LLMs and multi-modal LLMs.

- **Improving the Decoding Efficiency**: Models' have a high number of parameters which makes their inference slow and computationally expensive. Dynamic confidence based early exiting from the intermediate layers of a model tuned with explicit intermediate losses from the intermediate layers addresses this problem. This approaches improves the computational efficiency of text generation without sacrificing the quality of generation. Future avenues for research in this direction include studying speculative sampling from the intermediate layers to further improve the computational efficiency, and exploring the detection of hallucination from the outputs of the intermediate layers and their uncertainties.
- **Improving the Inference Efficiency**: With the increase in the model capacity (number of parameters), the performance typically tends to increase; however, it also increases the computation cost associated with inference. Model cascading is a simple yet effective solution to efficiently yet accurately output predictions. Furthermore, introducing additional models in the cascade further increases the efficiency improvements. Future avenues for research in this direction include studying the efficacy of cascading with the same large model but pruned to various sparsity targets.
- **Improving Open-domain QA Reader Efficiency**: Efficiently leveraging the closed-book and open-book inferences improve the efficiency of open-domain QA reader model. Dynamic reading in multiple iterations based on the model's

confidence saves the computation cost of the reader model while maintaining the prediction performance.

- **Improving Data Sample Efficiency for Training**: Crowdsourcing training data for a task has several challenges such as creating of too many trivial examples, similar examples, and erroneous examples. Synthetically creating the initial set of trivial examples and then adversarially collecting the non-trivial examples reduces the requirement of collecting expensive crowdsourced data. Future avenues for research in this direction include utilizing this approach for collecting instruction tuning data for LLMs and also selecting the honest examples for tuning safety critical models.
- **Improving Evaluation Efficiency**: Success of pre-trained language models has fostered in development of large number of models. Though, it has resulted in the availability of numerous model options for a task, comparing the performance of such a large number of models has become computationally expensive and time-consuming. Computing the difficult scores of evaluation instances helps in conducting efficient yet accurate evaluations with fewer instances. Future avenues for research in this direction include exploring ways to efficiently compute difficult scores of evaluation instances and incorporating more dimensions (apart from the performance of models) to the definition of difficulty.

REFERENCES

Adlakha, Vaibhav, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. 2023. "Evaluating correctness and faithfulness of instruction-following models for question answering." *arXiv preprint arXiv:2307.16877,* https://arxiv.org/abs/2307.16877.

Aliannejadi, Mohammad, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail Burtsev. 2020. "ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ)." *arXiv preprint arXiv:2009.11352,* https://arxiv.org/abs/2009.11352.

———. 2021. "Building and Evaluating Open-Domain Dialogue Corpora with Clarifying Questions." In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing,* 4473–4484. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2021.emnlp-main.367.

Anantha, Raviteja, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. "Open-Domain Question Answering Goes Conversational via Question Rewriting." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 520–534. Online: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/2021.naacl-main.44.

Aralikatte, Rahul, Shashi Narayan, Joshua Maynez, Sascha Rothe, and Ryan McDonald. 2021. "Focus Attention: Promoting Faithfulness and Diversity in Summarization." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers),* 6078–6095. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.acl-long.474.

Arase, Yuki, and et al. 2021. "Efficient NLP policy document, https://bit.ly/EfficientNLP."

Arazo, Eric, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. 2020. "Pseudo-labeling and confirmation bias in deep semi-supervised learning." In *2020 International Joint Conference on Neural Networks (IJCNN),* 1–8. IEEE. https://arxiv.org/abs/1908.02983.

Arunkumar, Anjana, Swaroop Mishra, Bhavdeep Sachdeva, Chitta Baral, and Chris Bryan. 2020. "Real-time visual feedback for educative benchmark creation: A human-and-metric-in-the-loop workflow," https://arxiv.org/abs/2302.04434.

Azaria, Amos, and Tom Mitchell. 2023a. "The Internal State of an LLM Knows When It's Lying." In *Findings of the Association for Computational Linguistics: EMNLP 2023,* edited by Houda Bouamor, Juan Pino, and Kalika Bali, 967–976. Singapore: Association for Computational Linguistics, December. https://aclanthology.org/2023.findings-emnlp.68.

Azaria, Amos, and Tom Mitchell. 2023b. "The internal state of an llm knows when its lying." *arXiv preprint arXiv:2304.13734,* https://arxiv.org/abs/2304.13734.

Bai, Yuntao, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. "Constitutional ai: Harmlessness from ai feedback." *arXiv preprint arXiv:2212.08073,* https://arxiv.org/abs/2212.08073.

Baker, Frank B, and Seock-Ho Kim. 2004. *Item response theory: Parameter estimation techniques.* CRC Press. https://www.publichealth.columbia.edu/research/population-health-methods/item-response-theory.

Banerjee, Pratyay, Chitta Baral, Man Luo, Arindam Mitra, Kuntal Pal, Tran C Son, and Neeraj Varshney. 2020. "Can Transformers Reason About Effects of Actions?" *arXiv preprint arXiv:2012.09938,* https://arxiv.org/abs/2012.09938.

Banerjee, Pratyay, Tejas Gokhale, and Chitta Baral. 2021. "Self-Supervised Test-Time Learning for Reading Comprehension." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 1200–1211. Online: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/2021.naacl-main.95.

Belinkov, Yonatan, and Yonatan Bisk. 2018. "Synthetic and Natural Noise Both Break Neural Machine Translation." In *International Conference on Learning Representations.* https://openreview.net/forum?id=BJ8vJebC-.

Bhagavatula, Chandra, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen-tau Yih, and Yejin Choi. 2020. "Abductive Commonsense Reasoning." In *International Conference on Learning Representations.* https://openreview.net/forum?id=Byg1v1HKDB.

Bowman, Samuel R., Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. "A large annotated corpus for learning natural language inference." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 632–642. Lisbon, Portugal: Association for Computational Linguistics, September. https://doi.org/10.18653/v1/D15-1075.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. "Language Models are Few-Shot Learners." In *Advances in Neural Information Processing Systems,* edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, 33:1877–1901. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cao, Meng, Yue Dong, and Jackie Cheung. 2022. "Hallucinated but Factual! Inspecting the Factuality of Hallucinations in Abstractive Summarization." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 3340–3354. Dublin, Ireland: Association for Computational Linguistics, May. https://doi.org/10.18653/v1/2022.acl-long.236.

Carpenter, Gail A., and Stephen Grossberg. 1988. "The ART of adaptive pattern recognition by a self-organizing neural network." *Computer* 21 (3): 77–88. https://ieeexplore.ieee.org/document/33.

Chen, Anthony, Panupong Pasupat, Sameer Singh, Hongrae Lee, and Kelvin Guu. 2023. "PURR: Efficiently Editing Language Model Hallucinations by Denoising Language Model Corruptions." *arXiv preprint arXiv:2305.14908,* https://arxiv.org/abs/2305.14908.

Chen, Charlie, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. "Accelerating large language model decoding with speculative sampling." *arXiv preprint arXiv:2302.01318,* https://arxiv.org/abs/2302.01318.

Chen, Danqi, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. "Reading Wikipedia to Answer Open-Domain Questions." In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 1870–1879. Vancouver, Canada: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P17-1171.

Chen, Dian, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. 2022. "Contrastive Test-Time Adaptation." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 295–305. https://arxiv.org/abs/2204.10377.

Chen, Sihao, Fan Zhang, Kazoo Sone, and Dan Roth. 2021. "Improving Faithfulness in Abstractive Summarization with Contrast Candidate Generation and Selection." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 5935–5941. Online: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/2021.naacl-main.475.

Chen, Tianlong, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. "The Lottery Ticket Hypothesis for Pre-trained BERT Networks." In *Advances in Neural Information Processing Systems,* edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, 33:15834–15846. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/b6af2c9703f203a2794be03d443af2e3-Paper.pdf.

Cheng, Zhoujun, Jungo Kasai, and Tao Yu. 2023. "Batch Prompting: Efficient Inference with Large Language Model APIs." In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track,* edited by Mingxuan Wang and Imed Zitouni, 792–810. Singapore: Association for Computational Linguistics, December. https://aclanthology.org/2023.emnlp-industry.74.

Chern, I, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. "FacTool: Factuality Detection in Generative AI–A Tool Augmented Framework for Multi-Task and Multi-Domain Scenarios." *arXiv preprint arXiv:2307.13528,* https://arxiv.org/abs/2307.13528.

Chiang, Wei-Lin, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, et al. 2023. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality,* March. https://lmsys.org/blog/2023-03-30-vicuna/.

Chow, Chi-Keung. 1957. "An optimum character recognition system using decision functions." *IRE Transactions on Electronic Computers,* no. 4, 247–254. https://ieeexplore.ieee.org/document/5222035.

Chowdhery, Aakanksha, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. "Palm: Scaling language modeling with pathways." *arXiv preprint arXiv:2204.02311,* https://arxiv.org/abs/2204.02311.

Chrysostomou, George, Zhixue Zhao, Miles Williams, and Nikolaos Aletras. 2023. "Lighter, yet More Faithful: Investigating Hallucinations in Pruned Large Language Models for Abstractive Summarization." *arXiv preprint arXiv:2311.09335,* https://arxiv.org/abs/2311.09355.

Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. "Scaling instruction-finetuned language models." *arXiv preprint arXiv:2210.11416,* https://arxiv.org/abs/2210.11416.

Clark, Kevin, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. "BAM! Born-Again Multi-Task Networks for Natural Language Understanding." In *Proceedings of the 57th Annual Meeting of the As-*

sociation for Computational Linguistics, 5931–5937. Florence, Italy: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P19-1595.

Clark, Kevin, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators." In *ICLR.* https://openreview.net/pdf?id=r1xMH1BtvB.

Clark, Peter, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. "Think you have solved question answering? try arc, the ai2 reasoning challenge." *arXiv preprint arXiv:1803.05457,* https://arxiv.org/abs/1803.05457.

Clark, Peter, Oyvind Tafjord, and Kyle Richardson. 2020. "Transformers as soft reasoners over language." *arXiv preprint arXiv:2002.05867,* https://arxiv.org/abs/2002.05867.

Cui, Wanyun, Guangyu Zheng, and Wei Wang. 2020. "Unsupervised Natural Language Inference via Decoupled Multimodal Contrastive Learning." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 5511–5520. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.444.

Dettmers, Tim, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. "GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale." In *Advances in Neural Information Processing Systems,* edited by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. https://openreview.net/forum?id=dXiGWqBoxaD.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N19-1423.

Dhingra, Bhuwan, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. "Handling Divergent Reference Texts when Evaluating Table-to-Text Generation." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 4884–4895. Florence, Italy: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P19-1483.

Dhuliawala, Shehzaad, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. "Chain-of-verification reduces hallucination in large language models." *arXiv preprint arXiv:2309.11495,* https://arxiv.org/abs/2309.11495.

Din, Alexander Yom, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. "Jump to Conclusions: Short-Cutting Transformers With Linear Transformations." *arXiv preprint arXiv:2303.09435,* https://arxiv.org/abs/2303.09435.

Dolan, William B, and Chris Brockett. 2005. "Automatically constructing a corpus of sentential paraphrases." In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005).* https://aclanthology.org/I05-5002/.

Du, Yilun, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. "Improving Factuality and Reasoning in Language Models through Multiagent Debate." *arXiv preprint arXiv:2305.14325,* https://arxiv.org/abs/2305.14325.

Dua, Dheeru, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. "DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 2368–2378. Minneapolis, Minnesota: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N19-1246.

Dunn, Matthew, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. "Searchqa: A new q&a dataset augmented with context from a search engine." *arXiv preprint arXiv:1704.05179,* https://arxiv.org/abs/1704.05179.

Dziri, Nouha, Andrea Madotto, Osmar Zaıane, and Avishek Joey Bose. 2021. "Neural Path Hunter: Reducing Hallucination in Dialogue Systems via Path Grounding." In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language*

Processing, 2197–2214. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2021.emnlp-main.168.

Elbayad, Maha, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. "Depth-Adaptive Transformer." In *International Conference on Learning Representations.* https://openreview.net/forum?id=SJg7KhVKPH.

Elsahar, Hady, and Matthias Gallé. 2019. "To Annotate or Not? Predicting Performance Drop under Domain Shift." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 2163–2173. Hong Kong, China: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/D19-1222.

Ettinger, Allyson. 2020. "What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models." Edited by Mark Johnson, Brian Roark, and Ani Nenkova. *Transactions of the Association for Computational Linguistics* (Cambridge, MA) 8:34–48. https://doi.org/10.1162/tacl_a_00298.

Feng, Yang, Wanying Xie, Shuhao Gu, Chenze Shao, Wen Zhang, Zhengxin Yang, and Dong Yu. 2020. "Modeling fluency and faithfulness for diverse neural machine translation." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 34:59–66. 01. https://arxiv.org/abs/1912.00178.

Fisch, Adam, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. "MRQA 2019 Shared Task: Evaluating Generalization in Reading Comprehension." In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering,* 1–13. https://aclanthology.org/D19-5801/.

Frantar, Elias, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. "OPTQ: Accurate Quantization for Generative Pre-trained Transformers." In *The Eleventh International Conference on Learning Representations.* https://openreview.net/forum?id=tcbBPnfwxS.

Gal, Yarin, and Zoubin Ghahramani. 2016. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In *international conference on machine learning,* 1050–1059. PMLR. https://arxiv.org/abs/1506.02142.

Garg, Siddhant, and Alessandro Moschitti. 2021. "Will this Question be Answered? Question Filtering via Answer Model Distillation for Efficient Question Answering." In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing,* 7329–7346. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2021.emnlp-main.583.

Geifman, Yonatan, and Ran El-Yaniv. 2017. "Selective Classification for Deep Neural Networks." In *NIPS.* https://arxiv.org/abs/1705.08500.

———. 2019. "SelectiveNet: A Deep Neural Network with an Integrated Reject Option." In *ICML.* https://arxiv.org/abs/1901.09192.

Geng, Xinyang, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. *Koala: A Dialogue Model for Academic Research.* Blog post, April. Accessed April 3, 2023. https://bair.berkeley.edu/blog/2023/04/03/koala/.

Gera, Ariel, Roni Friedman, Ofir Arviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim, and Eyal Shnarch. 2023. "The Benefits of Bad Advice: Autocontrastive Decoding across Model Layers." In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* edited by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, 10406–10420. Toronto, Canada: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2023.acl-long.580.

Glockner, Max, Vered Shwartz, and Yoav Goldberg. 2018. "Breaking NLI Systems with Sentences that Require Simple Lexical Inferences." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* 650–655. Melbourne, Australia: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P18-2103.

Gou, Zhibin, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. "Critic: Large language models can self-correct with tool-interactive critiquing." *arXiv preprint arXiv:2305.11738,* https://arxiv.org/abs/2305.11738.

Goyal, Saurabh, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. "PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination." In *International Conference on Machine Learning,* 3690–3699. PMLR. https://proceedings.mlr.press/v119/goyal20a/goyal20a.pdf.

Guo, Demi, Alexander Rush, and Yoon Kim. 2021. "Parameter-Efficient Transfer Learning with Diff Pruning." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers),* 4884–4896. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.acl-long.378.

Gururangan, Suchin, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. "Annotation Artifacts in Natural Language Inference Data." In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers),* 107–112. New Orleans, Louisiana: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N18-2017.

He, Zhenyu, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. 2023. "REST: Retrieval-Based Speculative Decoding." *arXiv preprint arXiv:2311.08252,* https://arxiv.org/abs/2311.08252.

Hendrycks, Dan, and Thomas Dietterich. 2019. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations." In *International Conference on Learning Representations.* https://openreview.net/forum?id=HJz6tiCqYm.

Hendrycks, Dan, and Kevin Gimpel. 2017. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks." *Proceedings of International Conference on Learning Representations,* https://arxiv.org/abs/1610.02136.

Hendrycks, Dan, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. "Pretrained Transformers Improve Out-of-Distribution Robustness." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 2744–2751. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.244.

Holtzman, Ari, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. "The Curious Case of Neural Text Degeneration." In *International Conference on Learning Representations.* https://openreview.net/forum?id=rygGQyrFvH.

Hossain, Md Mosharaf, and Eduardo Blanco. 2022. "Leveraging Affirmative Interpretations from Negation Improves Natural Language Understanding." In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing,* edited by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, 5833–5847.

Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, December. https://doi.org/10.18653/v1/2022.emnlp-main.393.

Hossain, Md Mosharaf, Dhivya Chinnappa, and Eduardo Blanco. 2022. "An Analysis of Negation in Natural Language Understanding Corpora." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* edited by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, 716–723. Dublin, Ireland: Association for Computational Linguistics, May. https://doi.org/10.18653/v1/2022.acl-short.81.

Hossain, Md Mosharaf, Venelin Kovatchev, Pranoy Dutta, Tiffany Kao, Elizabeth Wei, and Eduardo Blanco. 2020. "An Analysis of Natural Language Inference Benchmarks through the Lens of Negation." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* edited by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, 9106–9118. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.732.

Hosseini, Arian, Siva Reddy, Dzmitry Bahdanau, R Devon Hjelm, Alessandro Sordoni, and Aaron Courville. 2021. "Understanding by Understanding Not: Modeling Negation in Language Models." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* edited by Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, 1301–1312. Online: Association for

Computational Linguistics, June. https://doi.org/10.18653/v1/2021.naacl-main.102.

Hou, Lu, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. "DynaBERT: Dynamic BERT with Adaptive Width and Depth." In *Advances in Neural Information Processing Systems,* edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, 33:9782–9793. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/6f5216f8d89b086c18298e043bfe48ed-Paper.pdf.

Hsieh, Cheng-Yu, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. "Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes." In *Findings of the Association for Computational Linguistics: ACL 2023,* edited by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, 8003–8017. Toronto, Canada: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2023.findings-acl.507.

Hsu, Yen-Chang, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2020. "Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data." *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 10948–10957. https://arxiv.org/abs/2002.11297.

Huang, Yichong, Xiachong Feng, Xiaocheng Feng, and Bing Qin. 2021. "The factual inconsistency problem in abstractive text summarization: A survey." *arXiv preprint arXiv:2104.14839,* https://arxiv.org/abs/2104.14839.

Iandola, Forrest, Albert Shaw, Ravi Krishna, and Kurt Keutzer. 2020. "SqueezeBERT: What can computer vision teach NLP about efficient neural networks?" In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing,* 124–135. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.sustainlp-1.17.

Iyer, Shankar, Nikhil Dandekar, and Kornél Csernai. 2017. "First quora dataset release: Question pairs." *data. quora. com,* https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Iyyer, Mohit, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. "Adversarial Example Generation with Syntactically Controlled Paraphrase Networks." In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers),* 1875–1885. New Orleans, Louisiana: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N18-1170.

Izacard, Gautier, and Edouard Grave. 2021. "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering." In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational*

Linguistics: Main Volume, 874–880. Online: Association for Computational Linguistics, April. https://doi.org/10.18653/v1/2021.eacl-main.74.

Izacard, Gautier, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Sebastian Riedel, and Edouard Grave. 2020. "A memory efficient baseline for open domain question answering." *arXiv preprint arXiv:2012.15156,* https://arxiv.org/abs/2012.15156.

Jang, Joel, Seonghyeon Ye, and Minjoon Seo. 2023. "Can large language models truly understand prompts? a case study with negated prompts." In *Transfer Learning for Natural Language Processing Workshop,* 52–62. PMLR. https://arxiv.org/abs/2209.12711.

Ji, Ziwei, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. "Survey of hallucination in natural language generation." *ACM Computing Surveys* 55 (12): 1–38. https://arxiv.org/abs/2202.03629.

Jia, Robin, and Percy Liang. 2017. "Adversarial Examples for Evaluating Reading Comprehension Systems." In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* 2021–2031. Copenhagen, Denmark: Association for Computational Linguistics, September. https://doi.org/10.18653/v1/D17-1215.

Jiang, Zi-Hang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. "ConvBERT: Improving BERT with Span-based Dynamic Convolution." In *Advances in Neural Information Processing Systems,* edited by H.

Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, 33:12837–12848. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/96da2f590cd7246bbde0051047b0d6f7-Paper.pdf.

Jiang, Huiqiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. "Llmlingua: Compressing prompts for accelerated inference of large language models." *arXiv preprint arXiv:2310.05736,* https://arxiv.org/abs/2310.05736.

Jiang, Zhengbao, Jun Araki, Haibo Ding, and Graham Neubig. 2021. "How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering." *Transactions of the Association for Computational Linguistics* 9 (September): 962–977. https://doi.org/10.1162/tacl_a_00407. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00407/1962628/tacl_a_00407.pdf.

Jiao, Xiaoqi, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. "TinyBERT: Distilling BERT for Natural Language Understanding." In *Findings of the Association for Computational Linguistics: EMNLP 2020,* 4163–4174. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.findings-emnlp.372.

Joshi, Mandar, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension." In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), 1601–1611. Vancouver, Canada: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P17-1147.

Kadavath, Saurav, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. "Language models (mostly) know what they know." *arXiv preprint arXiv:2207.05221,* https://arxiv.org/abs/2207.05221.

Kamath, Amita, Robin Jia, and Percy Liang. 2020. "Selective Question Answering under Domain Shift." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 5684–5696. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.503.

Karpukhin, Vladimir, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. "Dense Passage Retrieval for Open-Domain Question Answering." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 6769–6781. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.550.

Kassner, Nora, and Hinrich Schütze. 2020. "Negated and Misprimed Probes for Pretrained Language Models: Birds Can Talk, But Cannot Fly." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* edited by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, 7811–

7818. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.698.

Kendall, Maurice G. 1938. "A new measure of rank correlation." *Biometrika* 30 (1/2): 81–93. https://www.jstor.org/stable/2332226.

Khattab, Omar, and Matei Zaharia. 2020. "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT." In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval,* 39–48. SIGIR '20. Virtual Event, China: Association for Computing Machinery. https://doi.org/10.1145/3397271.3401075.

Khot, Tushar, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. "Decomposed Prompting: A Modular Approach for Solving Complex Tasks." In *The Eleventh International Conference on Learning Representations.* https://openreview.net/forum?id=_nGgzQjzaRy.

Kiela, Douwe, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, et al. 2021. "Dynabench: Rethinking Benchmarking in NLP." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 4110–4124. Online: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/2021.naacl-main.324.

Kim, Daesik, Seonhoon Kim, and Nojun Kwak. 2019. "Textbook Question Answering with Multi-modal Context Graph Understanding and Self-supervised Open-set

Comprehension.” In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 3568–3584. Florence, Italy: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P19-1347.

Kim, Gyuwan, and Kyunghyun Cho. 2021. “Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search.” In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers),* 6501–6511. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.acl-long.508.

Kim, Sehoon, Karttikeya Mangalam, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. 2023. “Big little transformer decoder.” *arXiv preprint arXiv:2302.07863,* https://arxiv.org/abs/2302.07863.

Koh, Pang Wei, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, et al. 2021. “WILDS: A Benchmark of in-the-Wild Distribution Shifts.” In *Proceedings of the 38th International Conference on Machine Learning,* edited by Marina Meila and Tong Zhang, 139:5637–5664. Proceedings of Machine Learning Research. PMLR, 18–24 Jul. https://proceedings.mlr.press/v139/koh21a.html.

Kulkarni, Mayank, Debanjan Mahata, Ravneet Arora, and Rajarshi Bhowmik. 2021. “Learning rich representation of keyphrases from text.” *arXiv preprint arXiv:2112.08547,* https://arxiv.org/abs/2112.08547.

Kwiatkowski, Tom, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019a. "Natural questions: a benchmark for question answering research." *Transactions of the Association for Computational Linguistics* 7:453–466. https://aclanthology.org/Q19-1026/.

Kwiatkowski, Tom, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019b. "Natural Questions: A Benchmark for Question Answering Research." *Transactions of the Association for Computational Linguistics* 7 (March): 452–466. https://doi.org/10.1162/tacl_a_00276.

Lai, Guokun, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. "RACE: Large-scale ReAding Comprehension Dataset From Examinations." In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* 785–794. Copenhagen, Denmark: Association for Computational Linguistics, September. https://doi.org/10.18653/v1/D17-1082.

Lalor, John P., Hao Wu, Tsendsuren Munkhdalai, and Hong Yu. 2018. "Understanding Deep Learning Performance through an Examination of Test Set Difficulty: A Psychometric Case Study." In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,* 4711–4716. Brussels, Belgium: Association for Computational Linguistics, October. https://doi.org/10.18653/v1/D18-1500.

Lalor, John P., Hao Wu, and Hong Yu. 2016. "Building an Evaluation Scale using Item Response Theory." In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 648–657. Austin, Texas: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/D16-1062.

Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations." In *International Conference on Learning Representations.* https://openreview.net/forum?id=H1eA7AEtvS.

Lebret, Rémi, David Grangier, and Michael Auli. 2016. "Neural Text Generation from Structured Data with Application to the Biography Domain." In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 1203–1213. Austin, Texas: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/D16-1128.

Lee, Katherine, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. "Deduplicating Training Data Makes Language Models Better." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 8424–8445. Dublin, Ireland: Association for Computational Linguistics, May. https://doi.org/10.18653/v1/2022.acl-long.577.

Lee, Kimin, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks."

In *Advances in Neural Information Processing Systems,* edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf.

Lee, Nayeon, Wei Ping, Peng Xu, Mostofa Patwary, Pascale N Fung, Mohammad Shoeybi, and Bryan Catanzaro. 2022. "Factuality enhanced language models for open-ended text generation." *Advances in Neural Information Processing Systems* 35:34586–34599. https://arxiv.org/abs/2206.04624.

Leviathan, Yaniv, Matan Kalman, and Yossi Matias. 2023. "Fast inference from transformers via speculative decoding." In *International Conference on Machine Learning,* 19274–19286. PMLR. https://arxiv.org/abs/2211.17192.

Levy, Omer, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. "Zero-Shot Relation Extraction via Reading Comprehension." In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017),* 333–342. Vancouver, Canada: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/K17-1034.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.* arXiv: 1910.13461 [cs.CL]. https://aclanthology.org/2020.acl-main.703/.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 7871–7880. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.703.

Li, Haoran, Junnan Zhu, Jiajun Zhang, and Chengqing Zong. 2018. "Ensure the Correctness of the Summary: Incorporate Entailment Knowledge into Abstractive Sentence Summarization." In *Proceedings of the 27th International Conference on Computational Linguistics,* 1430–1441. Santa Fe, New Mexico, USA: Association for Computational Linguistics, August. https://aclanthology.org/C18-1121.

Li, Lei, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. "CascadeBERT: Accelerating Inference of Pre-trained Language Models via Calibrated Complete Models Cascade." In *Findings of the Association for Computational Linguistics: EMNLP 2021,* 475–486. Punta Cana, Dominican Republic: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2021.findings-emnlp.43.

Li, Linjie, Jie Lei, Zhe Gan, and Jingjing Liu. 2021. "Adversarial VQA: A New Benchmark for Evaluating the Robustness of VQA Models." In *International Conference on Computer Vision (ICCV).* https://arxiv.org/abs/2106.00245.

Li, Zheng, Zijian Wang, Ming Tan, Ramesh Nallapati, Parminder Bhatia, Andrew Arnold, Bing Xiang, and Dan Roth. 2022. "DQ-BART: Efficient Sequence-to-Sequence Model via Joint Distillation and Quantization." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* 203–211. Dublin, Ireland: Association for Computational Linguistics, May. https://doi.org/10.18653/v1/2022.acl-short.22.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and Larry Zitnick. 2014. "Microsoft COCO: Common Objects in Context." In *ECCV.* European Conference on Computer Vision, September. https://www.microsoft.com/en-us/research/publication/microsoft-coco-common-objects-in-context/.

Liu, Weitang, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. "Energy-based Out-of-distribution Detection." In *Advances in Neural Information Processing Systems,* edited by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, 33:21464–21475. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/f5496252609c43eb8a3d147ab9b9c006-Paper.pdf.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *ArXiv* abs/1907.11692. https://arxiv.org/abs/1907.11692.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692,* https://arxiv.org/abs/1907.11692.

Longpre, Shayne, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. "Entity-Based Knowledge Conflicts in Question Answering." In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing,* 7052–7063. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2021.emnlp-main.565.

Lu, Jiasen, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks." In *Advances in Neural Information Processing Systems,* edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/c74d97b01eae257e44aa9d5bade97baf-Paper.pdf.

Luo, Man, Shashank Jain, Anchit Gupta, Arash Einolghozati, Barlas Oguz, Debojeet Chatterjee, Xilun Chen, Chitta Baral, and Peyman Heidari. 2022. "A Study on the Efficiency and Generalization of Light Hybrid Retrievers." *arXiv preprint arXiv:2210.01371,* https://aclanthology.org/2023.acl-short.139/.

Luo, Man, Shrinidhi Kumbhar, Mihir Parmar, Neeraj Varshney, Pratyay Banerjee, Somak Aditya, Chitta Baral, et al. 2023. “Towards logiglue: A brief survey and a benchmark for analyzing logical reasoning capabilities of language models.” *arXiv preprint arXiv:2310.00836,* https://arxiv.org/abs/2310.00836.

Manakul, Potsawee, Adian Liusie, and Mark Gales. 2023. “SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models.” In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing,* edited by Houda Bouamor, Juan Pino, and Kalika Bali, 9004–9017. Singapore: Association for Computational Linguistics, December. https://aclanthology.org/2023.emnlp-main.557.

Manakul, Potsawee, Adian Liusie, and Mark JF Gales. 2023. “Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models.” *arXiv preprint arXiv:2303.08896,* https://arxiv.org/abs/2303.08896.

Marneffe, Marie-Catherine de, Mandy Simons, and Judith Tonhauser. 2019. “The CommitmentBank: Investigating projection in naturally occurring discourse.” https://semanticsarchive.net/Archive/Tg3ZGI2M/Marneffe.pdf.

Maynez, Joshua, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. “On Faithfulness and Factuality in Abstractive Summarization.” In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 1906–1919. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.173.

McCoy, R. Thomas, Junghyun Min, and Tal Linzen. 2020. "BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance." In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP,* 217–227. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.blackboxnlp-1.21.

McCoy, Tom, Ellie Pavlick, and Tal Linzen. 2019. "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 3428–3448. Florence, Italy: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P19-1334.

Michel, Paul, Xian Li, Graham Neubig, and Juan Pino. 2019. "On Evaluation of Adversarial Perturbations for Sequence-to-Sequence Models." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 3103–3114. Minneapolis, Minnesota: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N19-1314.

Miller, George A. 1995. "WordNet: a lexical database for English." *Communications of the ACM* 38 (11): 39–41. https://dl.acm.org/doi/10.1145/219717.219748.

Miller, John, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. 2020. "The effect of natural distribution shift on question answering models." In *International*

Conference on Machine Learning, 6905–6916. PMLR. https://arxiv.org/abs/2004.14444.

Min, Sewon, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, et al. 2021. "NeurIPS 2020 EfficientQA Competition: Systems, Analyses and Lessons Learned." In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track,* edited by Hugo Jair Escalante and Katja Hofmann, 133:86–111. Proceedings of Machine Learning Research. PMLR, June. https://proceedings.mlr.press/v133/min21a.html.

Min, Sewon, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. "FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation." *arXiv preprint arXiv:2305.14251,* https://arxiv.org/abs/2305.14251.

Mirzadeh, Seyed Iman, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. "Improved knowledge distillation via teacher assistant." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 34:5191–5198. 04. https://ojs.aaai.org/index.php/AAAI/article/view/5963.

Mishra, Swaroop, and Anjana Arunkumar. 2021. "How Robust are Model Rankings: A Leaderboard Customization Approach for Equitable Evaluation." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 35:13561–13569. 15. https://arxiv.org/abs/2106.05532.

Mishra, Swaroop, Anjana Arunkumar, Chris Bryan, and Chitta Baral. 2020. "Our Evaluation Metric Needs an Update to Encourage Generalization." *arXiv preprint arXiv:2007.06898,* https://arxiv.org/abs/2007.06898.

———. 2022. "Hardness of samples need to be quantified for a reliable evaluation system: Exploring potential opportunities with a new task." *arXiv preprint arXiv:2210.07631.*

Mishra, Swaroop, Anjana Arunkumar, Bhavdeep Sachdeva, Chris Bryan, and Chitta Baral. 2020. "DQI: Measuring Data Quality in NLP." *arXiv preprint arXiv:2005.00816,* https://arxiv.org/abs/2005.00816.

Mishra, Swaroop, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. "Cross-Task Generalization via Natural Language Crowdsourcing Instructions." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 3470–3487. Dublin, Ireland: Association for Computational Linguistics, May. https://doi.org/10.18653/v1/2022.acl-long.244.

Mishra, Swaroop, and Bhavdeep Singh Sachdeva. 2020. "Do We Need to Create Big Datasets to Learn a Task?" In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing,* 169–173. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.sustainlp-1.23.

Mitra, A., Ishan Shrivastava, and Chitta Baral. 2020. "Enhancing Natural Language Inference Using New and Expanded Training Data Sets and New Learning Models." In *AAAI.* https://ojs.aaai.org/index.php/AAAI/article/view/6371.

Mitra, Arindam, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, et al. 2023. *Orca 2: Teaching Small Language Models How to Reason.* arXiv: 2311.11045 [cs.AI]. https://arxiv.org/abs/2311.11045.

Modarressi, Ali, Hosein Mohebbi, and Mohammad Taher Pilehvar. 2022. "AdapLeR: Speeding up Inference by Adaptive Length Reduction." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 1–15. Dublin, Ireland: Association for Computational Linguistics, May. https://doi.org/10.18653/v1/2022.acl-long.1.

Mostafazadeh, Nasrin, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. "A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories." In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 839–849. San Diego, California: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N16-1098.

Naik, Aakanksha, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. "Stress Test Evaluation for Natural Language Inference." In *Proceedings of the 27th International Conference on Computational Linguis-*

tics, 2340–2353. Santa Fe, New Mexico, USA: Association for Computational Linguistics, August. https://www.aclweb.org/anthology/C18-1198.

Nakamura, Mutsumi, Santosh Mashetty, Mihir Parmar, Neeraj Varshney, and Chitta Baral. 2023. "LogicAttack: Adversarial Attacks for Evaluating Logical Consistency of Natural Language Inference." In *Findings of the Association for Computational Linguistics: EMNLP 2023,* edited by Houda Bouamor, Juan Pino, and Kalika Bali, 13322–13334. Singapore: Association for Computational Linguistics, December. https://doi.org/10.18653/v1/2023.findings-emnlp.889.

Nie, Yixin, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. "Adversarial NLI: A New Benchmark for Natural Language Understanding." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 4885–4901. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.441.

Ning, Xuefei, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. "Skeleton-of-thought: Large language models can do parallel decoding." *arXiv preprint arXiv:2307.15337,* https://arxiv.org/abs/2307.15337.

O'Brien, Sean, and Mike Lewis. 2023. "Contrastive decoding improves reasoning in large language models." *arXiv preprint arXiv:2309.09117,* https://arxiv.org/abs/2309.09117.

OpenAI. 2023. "GPT-4 Technical Report." *ArXiv* abs/2303.08774. https://api.semanticscholar.org/CorpusID:257532815.

Pan, Liangming, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. "Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies." *arXiv preprint arXiv:2308.03188,* https://arxiv.org/abs/2308.03188.

Pan, Yikang, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Wang. 2023. "On the Risk of Misinformation Pollution with Large Language Models." In *Findings of the Association for Computational Linguistics: EMNLP 2023,* edited by Houda Bouamor, Juan Pino, and Kalika Bali, 1389–1403. Singapore: Association for Computational Linguistics, December. https://doi.org/10.18653/v1/2023.findings-emnlp.97.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. "Scikit-learn: Machine learning in Python." *the Journal of machine Learning research* 12:2825–2830. https://www.jmlr.org/papers/v12/pedregosa11a.html.

Platt, John, et al. 1999. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods." *Advances in large margin classifiers* 10 (3): 61–74. https://home.cs.colorado.edu/~mozer/Teaching/syllabi/6622/papers/Platt1999.pdf.

Plummer, Bryan A, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models." In *Proceedings of the IEEE international conference on computer vision,* 2641–2649. https://arxiv.org/abs/1505.04870.

Rae, Jack W, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. "Scaling language models: Methods, analysis & insights from training gopher." *arXiv preprint arXiv:2112.11446,* https://arxiv.org/abs/2112.11446.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020a. "Exploring the limits of transfer learning with a unified text-to-text transformer." *The Journal of Machine Learning Research* 21 (1): 5485–5551. https://arxiv.org/abs/1910.10683.

———. 2020b. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." *Journal of Machine Learning Research* 21 (140): 1–67. http://jmlr.org/papers/v21/20-074.html.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 2383–2392. Austin, Texas: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/D16-1264.

Ranaldi, Leonardo, and Giulia Pucci. 2023. "When Large Language Models contradict humans? Large Language Models' Sycophantic Behaviour." *arXiv preprint arXiv:2311.09410,* https://arxiv.org/abs/2311.09410.

Rao, Sudha, and Hal Daumé III. 2018. "Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 2737–2746. Melbourne, Australia: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P18-1255.

Rehurek, Radim, and Petr Sojka. 2011. "Gensim–python framework for vector space modelling." *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3 (2). https://pypi.org/project/gensim/#:~:text=Gensim%20is%20a%20Python%20library,information%20retrieval%20(IR)%20community.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2018. "Semantically Equivalent Adversarial Rules for Debugging NLP models." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 856–865. Melbourne, Australia: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P18-1079.

Roberts, Adam, Colin Raffel, and Noam Shazeer. 2020. "How Much Knowledge Can You Pack Into the Parameters of a Language Model?" In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 5418–5426. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.437.

Rodriguez, Pedro, Joe Barrow, Alexander Miserlis Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. 2021. "Evaluation Examples are not Equally Informative: How should that change NLP Leaderboards?" In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers),* 4486–4503. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.acl-long.346.

Sagawa, Shiori, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. "An investigation of why overparameterization exacerbates spurious correlations." In *International Conference on Machine Learning,* 8346–8356. PMLR. https://arxiv.org/abs/2005.04345.

Saha, Amrita, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018. "DuoRC: Towards Complex Language Understanding with Paraphrased Reading Comprehension." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 1683–1693. Melbourne, Australia: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P18-1156.

Sakaguchi, Keisuke, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. "WINOGRANDE: An Adversarial Winograd Schema Challenge at Scale." In *AAAI.* https://arxiv.org/abs/1907.10641.

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *ArXiv* abs/1910.01108. https://arxiv.org/abs/1910.01108.

Sanh, Victor, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, et al. 2022. "Multitask Prompted Training Enables Zero-Shot Task Generalization." In *International Conference on Learning Representations.* https://openreview.net/forum?id=9Vrb9D0WI4.

Santilli, Andrea, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. 2023. "Accelerating Transformer Inference for Translation via Parallel Decoding." *arXiv preprint arXiv:2305.10427,* https://arxiv.org/abs/2305.10427.

Sap, Maarten, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. "Social IQA: Commonsense Reasoning about Social Interactions." In *EMNLP 2019.* https://arxiv.org/abs/1904.09728.

Schuster, Tal, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. "Confident adaptive language modeling." *Advances in Neural Information Processing Systems* 35:17456–17472. https://arxiv.org/abs/2207.07061.

Sharma, Mrinank, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. "Towards understanding sycophancy in language models." *arXiv preprint arXiv:2310.13548,* https://arxiv.org/abs/2310.13548.

Shen, Sheng, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. "Q-bert: Hessian based ultra low precision quantization of bert." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 34:8815–8821. 05. https://arxiv.org/abs/1909.05840.

Sheng, Sasha, Amanpreet Singh, Vedanuj Goswami, Jose Alberto Lopez Magana, Wojciech Galuba, Devi Parikh, and Douwe Kiela. 2021. "Human-Adversarial Visual Question Answering." https://arxiv.org/abs/2106.02280.

Smith, Shaden, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. "Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model." *arXiv preprint arXiv:2201.11990,* https://arxiv.org/abs/2201.11990.

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* 1631–1642.

Seattle, Washington, USA: Association for Computational Linguistics, October. https://aclanthology.org/D13-1170.

Soldaini, Luca, and Alessandro Moschitti. 2020. "The Cascade Transformer: an Application for Efficient Answer Sentence Selection." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 5697–5708. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.504.

Soo, Sander. 2014. "Object detection using Haar-cascade Classifier." *Institute of Computer Science, University of Tartu* 2 (3): 1–12. https://www.semanticscholar.org/paper/Object-detection-using-Haar-cascade-Classifier-Soo/0f1e866c3acb8a10f96b432e86f8a61be5eb6799.

Spector, Benjamin, and Chris Re. 2023. "Accelerating llm inference with staged speculative decoding." *arXiv preprint arXiv:2308.04623,* https://arxiv.org/abs/2308.04623.

Speer, Robyn, Joshua Chin, and Catherine Havasi. 2017. "Conceptnet 5.5: An open multilingual graph of general knowledge." In *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 31. 1. https://arxiv.org/abs/1612.03975.

Sun, Weiwei, Zhengliang Shi, Shen Gao, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2023. "Contrastive learning reduces hallucination in conversations." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 37:13618–13626. 11. https://arxiv.org/abs/2212.10400.

Swayamdipta, Swabha, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. "Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 9275–9293. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.746.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. "Rethinking the Inception Architecture for Computer Vision." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2818–2826. https://arxiv.org/abs/1512.00567.

Tafjord, Oyvind, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019. "Quarel: A dataset and models for answering questions about qualitative relationships." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 33:7063–7071. 01. https://arxiv.org/abs/1811.08048.

Tafjord, Oyvind, Matt Gardner, Kevin Lin, and Peter Clark. 2019. "QuaRTz: An Open-Domain Dataset of Qualitative Relationship Questions." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 5941–5946. Hong Kong, China: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/D19-1608.

Talmor, Alon, and Jonathan Berant. 2019. "MultiQA: An Empirical Investigation of Generalization and Transfer in Reading Comprehension." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 4911–4921. Florence, Italy: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P19-1485.

Talmor, Alon, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. "CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 4149–4158. Minneapolis, Minnesota: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N19-1421.

Talmor, Alon, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020. "Leap-Of-Thought: Teaching Pre-Trained Models to Systematically Reason Over Implicit Knowledge." In *Advances in Neural Information Processing Systems,* edited by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, 33:20227–20237. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/e992111e4ab9985366e806733383bd8c-Paper.pdf.

Tan, Hao, and Mohit Bansal. 2019. "LXMERT: Learning Cross-Modality Encoder Representations from Transformers." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 5100–

5111. Hong Kong, China: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/D19-1514.

Tan, Shawn, Yikang Shen, Chin-Wei Huang, and Aaron C. Courville. 2019. "Investigating Biases in Textual Entailment Datasets." *ArXiv* abs/1906.09635. https://arxiv.org/abs/1906.09635.

Tao, Chaofan, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. "Compression of Generative Pre-trained Language Models via Quantization." *arXiv preprint arXiv:2203.10705,* https://aclanthology.org/2022.acl-long.331.pdf.

Taori, Rohan, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. *Stanford Alpaca: An Instruction-following LLaMA model.* https://github.com/tatsu-lab/stanford_alpaca.

Tian, Ran, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. "Sticking to the facts: Confident decoding for faithful data-to-text generation." *arXiv preprint arXiv:1910.08684,* https://arxiv.org/abs/1910.08684.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. "Llama: Open and efficient foundation language models." *arXiv preprint arXiv:2302.13971,* https://arxiv.org/abs/2302.13971.

Trischler, Adam, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. "NewsQA: A Machine Comprehension Dataset." In *Proceedings of the 2nd Workshop on Representation Learning for NLP,* 191–200. Vancouver, Canada: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/W17-2623.

Truong, Thinh Hung, Timothy Baldwin, Karin Verspoor, and Trevor Cohn. 2023. "Language models are not naysayers: an analysis of language models on negation benchmarks." In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023),* edited by Alexis Palmer and Jose Camacho-collados, 101–114. Toronto, Canada: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2023.starsem-1.10.

Turc, Iulia, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "Well-read students learn better: On the importance of pre-training compact models." *arXiv preprint arXiv:1908.08962,* https://openreview.net/forum?id=BJg7x1HFvB.

Vakulenko, Svitlana, Nikos Voskarides, Zhucheng Tu, and Shayne Longpre. 2021. "A comparison of question rewriting methods for conversational passage retrieval." In *European Conference on Information Retrieval,* 418–424. Springer. https://arxiv.org/abs/2101.07382.

Vania, Clara, Phu Mon Htut, William Huang, Dhara Mungra, Richard Yuanzhe Pang, Jason Phang, Haokun Liu, Kyunghyun Cho, and Samuel R. Bowman. 2021. "Comparing Test Sets with Item Response Theory." In *Proceedings of the 59th*

Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 1141–1158. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.acl-long.92.

Varshney, Neeraj, Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. 2022. "Unsupervised Natural Language Inference Using PHL Triplet Generation." In *Findings of the Association for Computational Linguistics: ACL 2022,* 2003–2016. Dublin, Ireland: Association for Computational Linguistics, May. https://aclanthology.org/2022.findings-acl.159.

Varshney, Neeraj, and Chitta Baral. 2022. "Model Cascading: Towards Jointly Improving Efficiency and Accuracy of NLP Systems." In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing,* 11007–11021. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, December. https://doi.org/10.18653/v1/2022.emnlp-main.756.

———. 2023. "Post-Abstention: Towards Reliably Re-Attempting the Abstained Instances in QA." In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 967–982. Toronto, Canada: Association for Computational Linguistics, July. https://aclanthology.org/2023.acl-long.55.

Varshney, Neeraj, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. 2023. "Accelerating llm inference by enabling intermediate layer decoding." *arXiv preprint arXiv:2310.18581.*

Varshney, Neeraj, Man Luo, and Chitta Baral. 2022. "Can Open-Domain QA Reader Utilize External Knowledge Efficiently like Humans?" *arXiv preprint arXiv:2211.12707,* https://arxiv.org/abs/2211.12707.

Varshney, Neeraj, Swaroop Mishra, and Chitta Baral. 2022a. "ILDAE: Instance-Level Difficulty Analysis of Evaluation Data." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 3412–3425. Dublin, Ireland: Association for Computational Linguistics, May. https://aclanthology.org/2022.acl-long.240.

———. 2022b. "Investigating Selective Prediction Approaches Across Several Tasks in IID, OOD, and Adversarial Settings." In *Findings of the Association for Computational Linguistics: ACL 2022,* 1995–2002. Dublin, Ireland: Association for Computational Linguistics, May. https://aclanthology.org/2022.findings-acl.158.

———. 2022c. "Towards Improving Selective Prediction Ability of NLP Systems." In *Proceedings of the 7th Workshop on Representation Learning for NLP,* 221–226. Dublin, Ireland: Association for Computational Linguistics, May. https://doi.org/10.18653/v1/2022.repl4nlp-1.23.

Varshney, Neeraj, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. "A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation." *arXiv preprint arXiv:2307.03987,* https://arxiv.org/abs/2307.03987.

Wallace, Eric, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. "Universal Adversarial Triggers for Attacking and Analyzing NLP." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 2153–2162. Hong Kong, China: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/D19-1221.

Wang, Alex, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems." In *Advances in Neural Information Processing Systems,* edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding." In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP,* 353–355.

Brussels, Belgium: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/W18-5446.

Wang, Hongmin. 2019. "Revisiting Challenges in Data-to-Text Generation with Fact Grounding." In *Proceedings of the 12th International Conference on Natural Language Generation,* 311–322. Tokyo, Japan: Association for Computational Linguistics, October. https://doi.org/10.18653/v1/W19-8639.

Wang, Peiyi, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. "Large language models are not fair evaluators." *arXiv preprint arXiv:2305.17926,* https://arxiv.org/abs/2305.17926.

Wang, Xinyi, Yulia Tsvetkov, Sebastian Ruder, and Graham Neubig. 2021. "Efficient Test Time Adapter Ensembling for Low-resource Language Varieties." In *Findings of the Association for Computational Linguistics: EMNLP 2021,* 730–737. Punta Cana, Dominican Republic: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2021.findings-emnlp.63.

Wang, Yizhong, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. "Self-Instruct: Aligning Language Models with Self-Generated Instructions." In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* edited by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, 13484–13508. Toronto, Canada: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2023.acl-long.754.

Wang, Yizhong, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, et al. 2022. "Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks." In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing,* 5085–5109. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, December. https://aclanthology.org/2022.emnlp-main.340.

Wang, Ziheng, Jeremy Wohlwend, and Tao Lei. 2020. "Structured Pruning of Large Language Models." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 6151–6162. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.496.

Warstadt, Alex, Amanpreet Singh, and Samuel R. Bowman. 2019. "Neural Network Acceptability Judgments." *Transactions of the Association for Computational Linguistics* 7 (March): 625–641. https://doi.org/10.1162/tacl_a_00290.

Wei, Jason, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. "Finetuned Language Models are Zero-Shot Learners." In *International Conference on Learning Representations.* https://openreview.net/forum?id=gEZrGCozdqR.

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. "Chain-of-thought prompting elicits reasoning in large language models." *Advances in Neural Information Processing Systems* 35:24824–24837. https://arxiv.org/abs/2201.11903.

Wei, Jerry, Da Huang, Yifeng Lu, Denny Zhou, and Quoc V Le. 2023. "Simple synthetic data reduces sycophancy in large language models." *arXiv preprint arXiv:2308.03958,* https://arxiv.org/abs/2308.03958.

Weiss, David J. 1982. "Improving measurement quality and efficiency with adaptive testing." *Applied psychological measurement* 6 (4): 473–492. https://journals.sagepub.com/doi/10.1177/014662168200600408.

Welleck, Sean, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. "Dialogue Natural Language Inference." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 3731–3741. Florence, Italy: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/P19-1363.

Williams, Adina, Nikita Nangia, and Samuel Bowman. 2018. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference." In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers),* 1112–1122. New Orleans, Louisiana: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N18-1101.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, et al. 2020. "Transformers: State-of-the-Art Natural Language Processing." In *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing: System Demonstrations, 38–45. Online: Association for Computational Linguistics, October. https://doi.org/10.18653/v1/2020.emnlp-demos.6.

Xin, Ji, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. "DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 2246–2251. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.204.

Xin, Ji, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. "The Art of Abstention: Selective Prediction and Error Regularization for Natural Language Processing." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers),* 1040–1051. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.acl-long.84.

Xu, Benfeng, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. "Curriculum Learning for Natural Language Understanding." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics,* 6095–6104. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-main.542.

Xu, Can, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. "Wizardlm: Empowering large language models to follow complex instructions." *arXiv preprint arXiv:2304.12244,* https://arxiv.org/abs/2304.12244.

Yamada, Ikuya, Akari Asai, and Hannaneh Hajishirzi. 2021. "Efficient Passage Retrieval with Hashing for Open-domain Question Answering." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers),* 979–986. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.acl-short.123.

Yang, Nan, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. "Inference with reference: Lossless acceleration of large language models." *arXiv preprint arXiv:2304.04487,* https://arxiv.org/abs/2304.04487.

Yang, Seongjun, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. 2023. "Predictive Pipelined Decoding: A Compute-Latency Trade-off for Exact LLM Decoding." *arXiv preprint arXiv:2307.05908,* https://arxiv.org/abs/2307.05908.

Yang, Zhilin, Zihang Dai, Yiming Yang, J. Carbonell, R. Salakhutdinov, and Quoc V. Le. 2019a. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." In *NeurIPS.* https://arxiv.org/abs/1906.08237.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. "XLNet: Generalized Autoregressive Pretraining for Language

Understanding." In *Advances in Neural Information Processing Systems,* edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Yang, Zhilin, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering." In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,* 2369–2380. Brussels, Belgium: Association for Computational Linguistics, October. https://doi.org/10.18653/v1/D18-1259.

El-Yaniv, Ran, et al. 2010. "On the Foundations of Noise-free Selective Classification." *Journal of Machine Learning Research* 11 (5). https://jmlr.csail.mit.edu/papers/volume11/el-yaniv10a/el-yaniv10a.pdf.

Yao, Zhewei, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers." *Advances in Neural Information Processing Systems* 35:27168–27183. https://arxiv.org/abs/2206.01861.

Ye, Mengyu, Tatsuki Kuribayashi, Jun Suzuki, Goro Kobayashi, and Hiroaki Funayama. 2023. "Assessing Step-by-Step Reasoning against Lexical Negation: A Case Study on Syllogism." In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing,* edited by Houda Bouamor, Juan

Pino, and Kalika Bali, 14753–14773. Singapore: Association for Computational Linguistics, December. https://doi.org/10.18653/v1/2023.emnlp-main.912.

Yue, Murong, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. 2023. "Large Language Model Cascades with Mixture of Thoughts Representations for Cost-efficient Reasoning." *arXiv preprint arXiv:2310.03094,* https://arxiv.org/abs/2310.03094.

Zamani, Hamed, Susan T. Dumais, Nick Craswell, Paul N. Bennett, and Gord Lueck. 2020. "Generating Clarifying Questions for Information Retrieval." *Proceedings of The Web Conference 2020,* https://dl.acm.org/doi/10.1145/3366423.3380126.

Zamani, Hamed, Gord Lueck, Everest Chen, Rodolfo Quispe, Flint Luu, and Nick Craswell. 2020. "MIMICS: A Large-Scale Data Collection for Search Clarification." In *Proceedings of the 29th ACM International on Conference on Information and Knowledge Management.* CIKM '20. https://arxiv.org/abs/2006.10174.

Zellers, Rowan, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. "SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference." In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP).* https://arxiv.org/abs/1808.05326.

Zhang, Jingqing, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.* arXiv: 1912.08777 `[cs.CL]`. https://arxiv.org/abs/1912.08777.

Zhang, Shujian, Chengyue Gong, and Eunsol Choi. 2021. "Knowing More About Questions Can Help: Improving Calibration in Question Answering." In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021,* 1958–1970. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.findings-acl.172.

Zhang, Tianhua, Hongyin Luo, Yung-Sung Chuang, Wei Fang, Luc Gaitskell, Thomas Hartvigsen, Xixin Wu, Danny Fox, Helen Meng, and James Glass. 2023. "Interpretable unified language checking." *arXiv preprint arXiv:2304.03728,* https://arxiv.org/abs/2304.03728.

Zhang, Wei, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. "TernaryBERT: Distillation-aware Ultra-low Bit BERT." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 509–521. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.37.

Zhang, Xiang, Junbo Zhao, and Yann LeCun. 2015. "Character-level Convolutional Networks for Text Classification." In *Advances in Neural Information Processing Systems,* edited by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

Zhang, Yizhe, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. "DIALOGPT : Large-Scale

Generative Pre-training for Conversational Response Generation." In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations,* 270–278. Online: Association for Computational Linguistics, July. https://doi.org/10.18653/v1/2020.acl-demos.30.

Zhang, Yuan, Jason Baldridge, and Luheng He. 2019. "PAWS: Paraphrase Adversaries from Word Scrambling." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),* 1298–1308. Minneapolis, Minnesota: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/N19-1131.

Zhao, Ruochen, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023. "Verify-and-edit: A knowledge-enhanced chain-of-thought framework." *arXiv preprint arXiv:2305.03268,* https://arxiv.org/abs/2305.03268.

Zhao, Tiancheng, Xiaopeng Lu, and Kyusong Lee. 2021. "SPARTA: Efficient Open-Domain Question Answering via Sparse Transformer Matching Retrieval." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 565–575. Online: Association for Computational Linguistics, June. https://doi.org/10.18653/v1/2021.naacl-main.47.

Zhao, Zheng, Shay B. Cohen, and Bonnie Webber. 2020. "Reducing Quantity Hallucinations in Abstractive Summarization." In *Findings of the Association for Com-*

putational Linguistics: EMNLP 2020, 2237–2249. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.findings-emnlp.203.

Zheng, Lianmin, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena." *arXiv preprint arXiv:2306.05685,* https://arxiv.org/abs/2306.05685.

Zhong, Ruiqi, Dhruba Ghosh, Dan Klein, and Jacob Steinhardt. 2021. "Are Larger Pretrained Language Models Uniformly Better? Comparing Performance at the Instance Level." In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021,* 3813–3827. Online: Association for Computational Linguistics, August. https://doi.org/10.18653/v1/2021.findings-acl.334.

Zhou, Denny, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, et al. 2023. "Least-to-Most Prompting Enables Complex Reasoning in Large Language Models." In *The Eleventh International Conference on Learning Representations.* https://openreview.net/forum?id=WZH7099tgfM.

Zhou, Wenxuan, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023. "Context-faithful prompting for large language models." *arXiv preprint arXiv:2303.11315,* https://arxiv.org/abs/2303.11315.

Zhou, Xiang, Yixin Nie, Hao Tan, and Mohit Bansal. 2020. "The Curse of Performance Instability in Analysis Datasets: Consequences, Source, and Suggestions."

In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 8215–8228. Online: Association for Computational Linguistics, November. https://doi.org/10.18653/v1/2020.emnlp-main.659.