Towards Understanding the Role of Knowledge in Improving Transformer-based Language Models

by

Kuntal Kumar Pal

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

Approved June 2023 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Ruoyu Wang
Eduardo Blanco
Yezhou Yang

ARIZONA STATE UNIVERSITY

August 2023

ABSTRACT

In natural language processing, language models have achieved remarkable success over the last few years. The Transformers are at the core of most of these models. Their success can be mainly attributed to an enormous amount of curated data they are trained on. Even though such language models are trained on massive curated data, they often need specific extracted knowledge to understand better and reason. This is because often relevant knowledge may be implicit or missing, which hampers machine reasoning. Apart from that, manual knowledge curation is time-consuming and erroneous. Hence, finding fast and effective methods to extract such knowledge from data is important for improving language models. This leads to finding ideal ways to utilize such knowledge by incorporating them into language models. Successful knowledge extraction and integration lead to an important question of knowledge evaluation of such models by developing tools or introducing challenging test suites to learn about their limitations and improve them further. So to improve the transformer-based models, understanding the role of knowledge becomes important.

In the pursuit to improve language models with knowledge, in this dissertation I study three broad research directions spanning across the natural language, biomedical and cybersecurity domains: (1) Knowledge Extraction (KX) - How can transformer-based language models be leveraged to extract knowledge from data? (2) Knowledge Integration (KI) - How can such specific knowledge be used to improve such models? (3) Knowledge Evaluation (KE) - How can language models be evaluated for specific skills and understand their limitations?

I propose methods to extract explicit textual, implicit structural, missing textual, and missing structural knowledge from natural language and binary programs using transformer-based language models. I develop ways to improve the language model's multi-step and commonsense reasoning abilities using external knowledge. Finally,

I develop challenging datasets which assess their numerical reasoning skills in both in-domain and out-of-domain settings.

DEDICATION

Dedicated to my Parents, Sister, Thakuma, and my Nieces.

ACKNOWLEDGMENTS

I would like to thank my Ph.D. advisor, Dr. Chitta Baral, for his support, and motivation and for giving me the opportunity to conduct research in Cognition Intelligence Lab. He has introduced me to new opportunities, and collaborators and has supported me in choosing my research direction. Without his guidance, this thesis would not have been completed. I would also like to extend my sincere gratitude to Dr. Ruoyu Wang for introducing me to applications of NLP in Cybersecurity Research. His advice, guidance, and encouragement have made many projects in my dissertation possible. I would also like to thank Dr. Yezhou Yang and Dr. Eduardo Blanco for accepting to be on my dissertation committee and also for their time, support, and valuable feedback.

I am also grateful to other professors at ASU, Dr. Adam Doupe and Dr. Yan Shoshitaishvili, for their guidance and valuable research ideas in multiple Cybersecurity research programs. I am also thankful to professors, Dr. Murthy Devarakonda, Dr. Saadat Anwar, and Dr. Huan Liu, for various research discussions and collaboration opportunities.

I would like to thank my professors at NIT Calicut Dr. K. S. Sudeep and Dr. SD Madhu Kumar for their guidance and encouragement to pursue Ph.D. I would also like to thank my internship mentors at Microsoft Research, Silviu Cucerzan, Michael Gamon, and Nirupama Chandrasekaran for their constant encouragement, time, support, and valuable feedback. I am also grateful to my industry mentor, Atit Jain, for being supportive during my time at Cavium Networks and recommending me to doctoral programs at various universities.

The advice, encouragement, discussions, collaborations, cricket matches, trips, treks, and other fun memories with my peers at ASU have kept me engaged during my Ph.D. So, I would like to extend my sincere gratitude to my peers at ASU, Pratyay, Arindam, Arpit, Swaroop, Neeraj, Mihir, Kazuaki, Shailaja, Tejas, Man, Ati, Yiran, Garima, Himanshu, Saurabh, Siddhesh, Ujjwala, Mirali, Paras, Pulkit, Pavel, Ming, Maitreya, Savan, Tasneema, Agneet, Kaustav, Sandipan, Shreyas, Zeyad, Romena, Sujit, Munish, Parul, and many others. Next, I want to thank my Bangalore family, Prantik, Arjun, Subhro, PKD, Barun, Zoaib, and Manna Da for the happy memories and for always being there, encouraging me. Special thanks go to my roommates Ankan Mitra and Prantik Howlader for being a constant source of motivation and also being such patient listeners. I was also lucky to have made travel buddies, Debkalpa and Prakash with whom I have made some wonderful memories during my Ph.D. Apart from them, I would also thank ASU, DARPA, NSF, and other agencies for funding my research and providing valuable scholarships. The clusters at ASU HPC, Kingkong, Godzilla, and Godzilla2 enabled me to conduct my research and complete this dissertation.

Lastly, I would like to thank my parents for their unconditional and eternal love and support. This journey would not have been possible without the freedom to choose my own path irrespective of the situation. I would also like to extend my gratitude to my thakuma (grandmother), sister, nieces, cousins, and other members of my family for their love and support.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Research Contributions

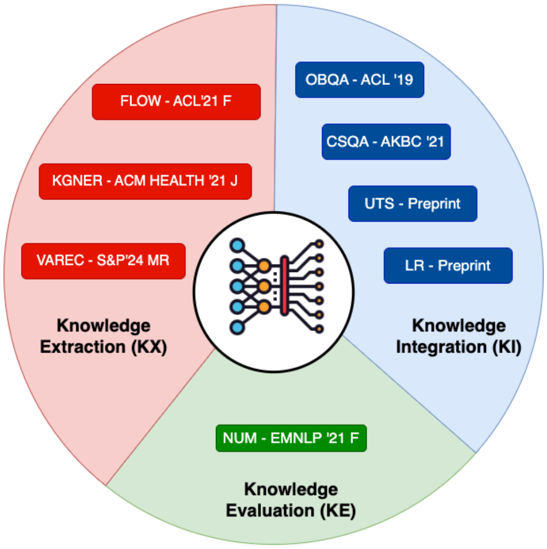
In Natural Language Processing, language models (LM) have achieved remarkable success over the last few years. Most of these recent language models are inspired by either of the transformer (Vaswani *et al.*, 2017) families - Encoder-Only, Decoder-Only, and Encoder-Decoder. The majority of their success can be attributed to the enormous amount of curated data (Wikipedia, BookCorpus, etc) they are trained on.

Even though such language models are trained on massive curated data, they need specific extracted knowledge (structured and unstructured) from such data to understand better and reason. This is because often relevant knowledge may be implicit or missing, which hampers machine reasoning. Apart from that, manual knowledge curation is time-consuming and erroneous. Hence, finding effective methods to *extract knowledge* from data is important for improving the language models.

Extraction of knowledge from data leads to the question of what is the best method of utilizing such knowledge. More concretely, can we find ways to *incorporate such special knowledge into language models* (other than only representation learning) that can make them more effective?

Successful extraction of specific knowledge and their integration into language models leads to an important question of *effective evaluation* of such models. Such kind of model assessment by developing tools or curating challenging datasets can help us learn their limitations and improve them even further.

So, to improve the currently existing transformer-based models three broad research directions become relevant which are the specific research contributions in this dissertation: (1) ***Knowledge Extraction (KX)*** - *How can we leverage transformer-based language models to extract knowledge from data?* (2) ***Knowledge Integration (KI)*** - *How can we use such specific knowledge to improve these models?* (3) ***Knowledge Evaluation (KE)*** - *How can we evaluate language models for specific skills?* . In this dissertation, I will present projects covering a set of diverse domains like natural language, biomedical, and cybersecurity catering to these three research directions. The overall dissertation structure is illustrated in the Figure below.



F : Findings, J : Journal, MR : Major Revision

Chapter 2 (KGNER): Biomedical named entity recognition via knowledge guidance and question answering

Chapter 3 (FLOW): Constructing flow graphs from procedural cybersecurity texts

Chapter 4 (VAREC): ”Len or index or count, anything but v1”: Predicting Variable Names in Decompilation Output with Transfer Learning

Chapter 5 (OBQA): Careful selection of knowledge to solve open book question-answering

Chapter 6 (CSQA): Commonsense Reasoning with implicit knowledge in natural language

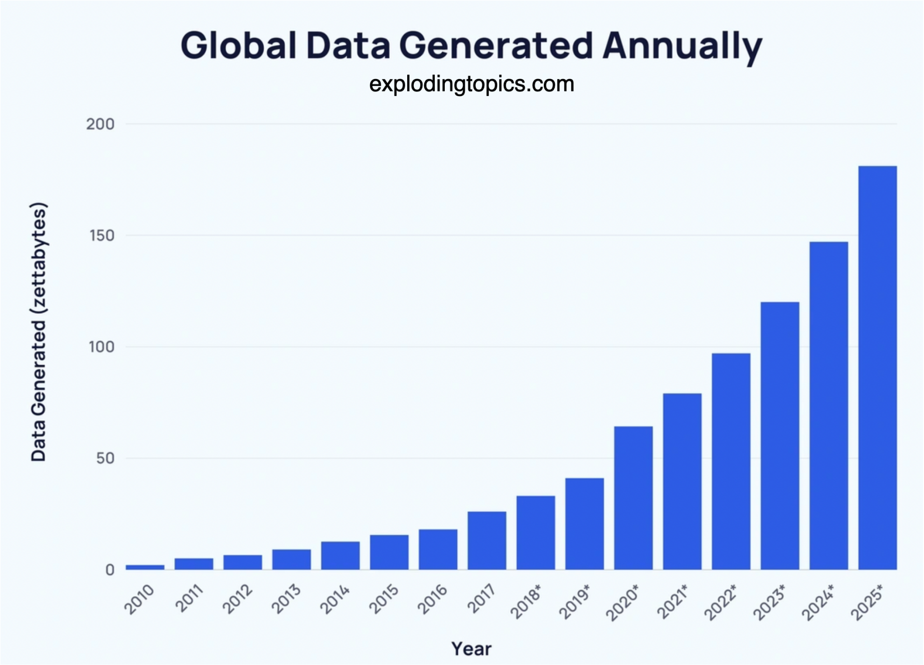
Chapter 7 (UTS): Exploring the Limits of Transfer Learning with Unified Model in the Cybersecurity Domain.

Chapter 8 (LR): Injecting Logical Reasoning Skills into Language Models.

Chapter 9 (NUM): Investigating numeracy learning ability of a text-to-text transfer model

1.2 Knowledge Extraction (KX)

The amount of data generated annually is exponentially growing since 2010. In fact, it is estimated that 90% of the world's data was generated in the last two



years alone and approximately 328.77 million terabytes of data are created each day (Figure above). A large portion of data on the World Wide Web is encoded in natural language form and therefore unstructured. An estimate shows that the data of a typical corporation is in natural language and hence goes unutilized as Dark Data (Ritsko and D.I., 2004). But this data, created each day, is nothing but an aggregation of symbols, and characters collected and measured by various devices. They do not have meaning unless they are related to their contexts and processed into information. The understanding of the data becomes much more clear when this processed information is interlinked based on the contexts into knowledge, which influences people to take informed actions.

However, for humans, extracting essential knowledge from data is challenging because (1) the *form of data vary* from texts, documents, codes, programs, images, to videos, (2) the *nature of knowledge* we need can be explicit, implicit or contextual, and (3) *manual processing is inefficient*: error-prone and time-consuming. Hence such challenges demand efficient, effective, generic, error-minimizing automated systems to extract the most knowledge out of these under-utilized data. So, in this dissertation, I show various ways in which we can use the latest transformer-based models to extract knowledge from multiple forms of data.

Outline of Knowledge Extraction

In Chapter 2, I show how LMs can be used to extract explicit entity-specific knowledge in the biomedical domain. Here, I show that by using *entity-type definitions and a few examples, a question-answering-based LM can effectively extract named entities* from 18 biomedical datasets comprising patient reports and healthcare documents.

This is perhaps the first work that explores the use of definitions and examples in instructions. Later such instructions have been adopted by various instruction-tuning works (Mishra *et al.*, 2021a; Wang *et al.*, 2022c; Efrat and Levy, 2020) when the prompting literature gained popularity.

In Chapter 3, I present a relatively harder task of *extracting inherent structured information from unstructured public forum cybersecurity documents.* Here I demonstrate that a modified LM along with Graph Neural Network (GNN) can be effectively applied to generate a structured flow graph in three datasets in both cybersecurity and natural language domains.

In Chapter 4, I show a much harder task of retrieving missing knowledge from software binaries. The task is harder since binaries are human-unreadable. Here I show that a customized LM, three times smaller than the BERT-base can be used effectively to *recover missing human-authored variable names from compiled source code* for two different state-of-the-art binary decompilers.

1.3 Knowledge Integration (KI)

As we keep building tools to process knowledge from ever-growing unutilized data, our understanding of the data, ability to gain important insights, and readability improve. We now can take informed actions about the data or person toward a goal. On the other hand, the automated NLP systems which assist humans in various tasks have also developed and have shown tremendous performance.

One of the main reasons for the success of the NLP systems in recent times is the large parameter space. The number of parameters is continuously growing making the models heavier and difficult to train on smaller compute and resources. They show extraordinary abilities to perform general tasks. But on multiple occasions, it has been shown that they lack specific human-level skills, for example commonsense, logical, or multi-hop reasoning skills.

Hence it is important to develop principled approaches to inject such special skills into these general-purpose language models. Additionally, such approaches also could help smaller models, which do not have large enough parameter space to learn everything from huge data repositories into their parameters or to gain such specific skills. Hence in this dissertation, I explore approaches to integrate specially curated explicit instance-specific knowledge from various available sources to improve the reasoning skills of transformer-based language models.

Outline of Knowledge Integration

In Chapter 5, I present an approach to incorporate external scientific knowledge into language models from relevant sources which helps them to perform *multi-step reasoning* with missing knowledge.

In Chapter 6, I show that we can retrieve instance-specific external knowledge from relevant knowledge sources, to assist smaller language models to *understand how humans interact with physical objects to achieve a goal.* Here I used four knowledge integration methods and three training approaches.

In Chapter 7, I show that training a generative LM on the diverse nature of cybersecurity texts and tasks in a multi-task setting can help it *to improve and also adapt to unseen domains and tasks.*

In Chapter 8, I demonstrate that by continued pre-training, *logical reasoning skills can be incorporated* into language models. I also show that language models with such skills can achieve good performance over vanilla language models on a logical MCQ dataset.

1.4 Knowledge Evaluation (KE)

Developing an artificial general intelligence (AGI) agent has been an age-old goal of researchers working in AI. Even though the term AGI was first introduced by Gubrud (1997) way back in the year 1997, it has gained popularity only in recent times. This is mostly because of the remarkable achievements of general-purpose LLMs like GPT3 (Brown *et al.*, 2020a), ChatGPT (OpenAI, 2023), and GPT4 (OpenAI, 2023).

GPT3 has led to the development and public release of models in a similar line of research like T5 (Raffel *et al.*, 2020a), PaLM (Chowdhery *et al.*, 2022), FLAN (Wei *et al.*, 2021), Chinchilla (Hoffmann *et al.*, 2022), Jurrasic (Lieber *et al.*, 2021), LLAMA (Touvron *et al.*, 2023), and many others. But as new models get released, there is the need to assess their skills, their reasoning abilities, and the limitations of their knowledge. Such kind of *Knowledge Evaluation* of models can help us figure out opportunities for their improvements leading to the development of more general artificial intelligence.

Hence, I consider Knowledge Evaluation (KE) of transformer-based language models as the third direction of my research in this dissertation. More specifically, my goal is to test LMs on their reasoning skills especially numerical reasoning skills by developing synthetic test suites in various experimental settings.

Outline of Knowledge Evaluation

In Chapter 9, I show the limits to which language models can understand various numerical reasoning tasks both in interpolation and extrapolation settings through synthetic and curated datasets. Here I demonstrate how far the three versions of T5 language models (small, base, and large) can reason for numeration tasks (number form understanding), magnitude order prediction tasks (number value understanding), and finding minimum-maximum (number comparison) or sorting numbers (number comparison and manipulation) in a numerical series.

All these projects have led to the development of datasets, tools, and models which have been open-sourced for the community for future researchers. The GitHub links can be found within individual chapters.

Chapter 2

BIOMEDICAL NAMED ENTITY RECOGNITION VIA KNOWLEDGE GUIDANCE AND QUESTION ANSWERING

ABSTRACT

In this work, we formulated the named entity recognition (NER) task as a multi-answer knowledge-guided question-answer task (KGQA) and showed that the knowledge guidance helps to achieve state-of-the-art results for 11 out of 18 biomedical NER datasets. We prepended five different knowledge contexts – entity types, questions, definitions, and examples – to the input text and trained and tested BERT-based neural models on such input sequences from a combined dataset of 18 different datasets. This novel formulation of task (a) improved named entity recognition and illustrated the impact of different knowledge contexts, (b) reduced system confusion by limiting prediction to a single entity class for each input token (i.e., B, I, O only) compared to multiple entity classes in traditional NER (i.e. B-entity$_1$, B-entity$_2$, I-entity$_1$, I-entity$_2$, O), (c) made detection of nested entities easier (d) enabled the models to jointly learn NER-specific features from a large number of datasets. We performed extensive experiments of this KGQA formulation on biomedical datasets, and through the experiments, we showed that knowledge improved named entity recognition. We analyzed the effect of the task formulation, the impact of the different knowledge contexts, the multi-task aspect of the generic format, and the generalization ability of KGQA. We also probed the model to better understand the key contributors to these improvements.

2.1 Introduction

Named Entity Recognition (NER) is an essential step in biomedical natural language processing, often serving as the first step on tasks such as relation extraction. Although recent developments in transformer language models, such as BERT (Devlin *et al.*, 2019) have led to high (> 90 F1) NER F1 score in a few datasets (Krallinger *et al.*, 2015; Wei *et al.*, 2015; Lee *et al.*, 2020), in several other datasets, such as Ex-PTM (Pyysalo *et al.*, 2011), JNLPBA (Kim *et al.*, 2004) and BIONLP13GE (Nédellec *et al.*, 2013), the state-of-the-art F1 is still below 85. Historically, NER has been considered a relatively difficult task in the biomedical domain due to the stylized writing and domain-specific terminology. Moreover, the target entities are usually proper nouns or unregistered words, with new words for drugs, diseases, and chemicals being generated frequently. The same phrases can also be recognized as different named entities depending on the current context (Cohen and Hunter, 2004; Liu *et al.*, 2006; Song *et al.*, 2018). For these reasons, external knowledge can help guide automated systems to identify the entities in the biomedical domain.

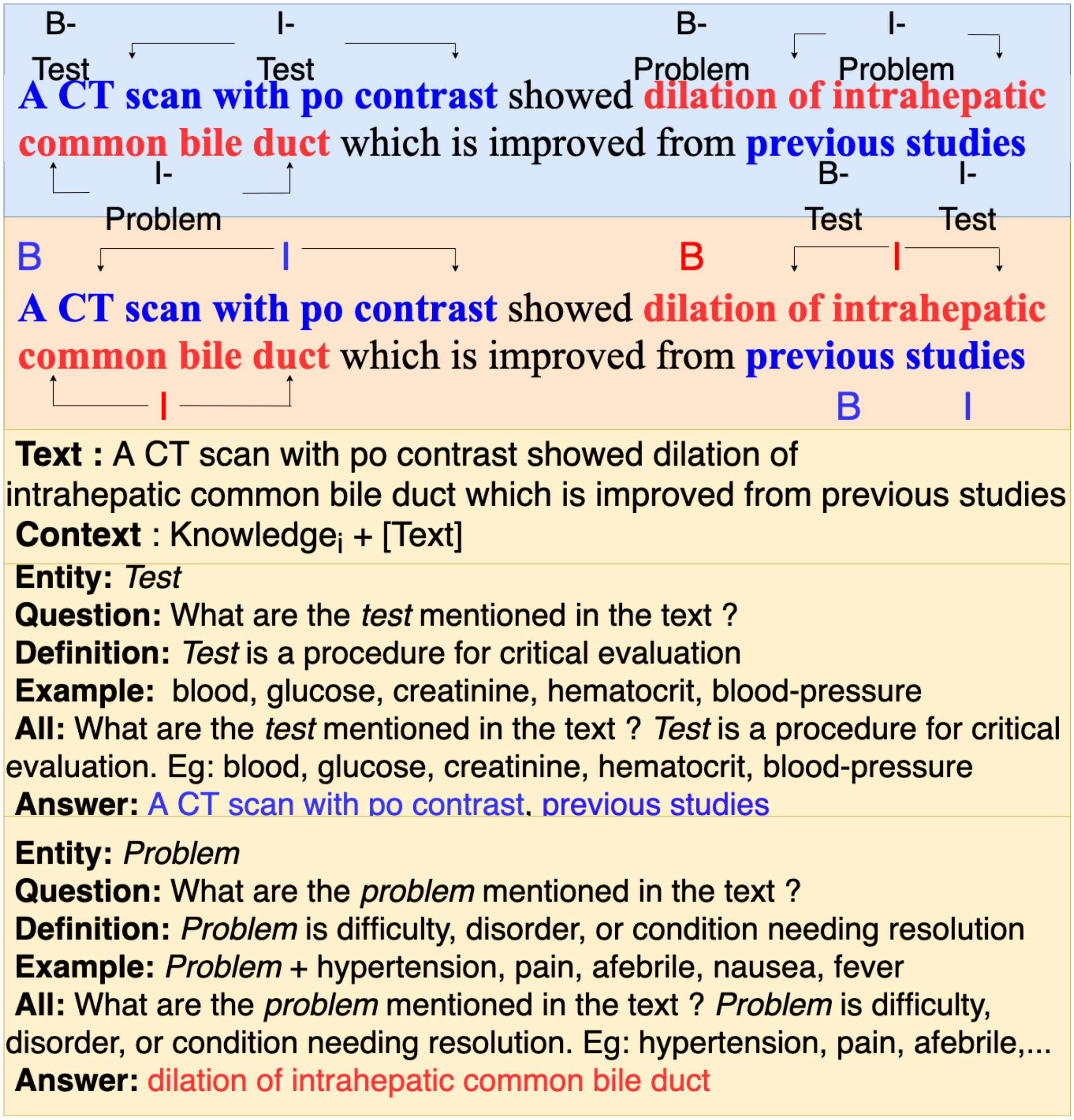
In general natural language processing, extensive external knowledge has helped systems in tasks such as commonsense question answering (Sap *et al.*, 2019; Talmor *et al.*, 2019) and science question answering (Lai *et al.*, 2017; Mihaylov *et al.*, 2018). In biomedical named entity recognition, external knowledge can be about entities and their relations. Knowledge like entity types, their definition, and examples can allow attention mechanisms to compare and learn to detect new entities, primarily if the entity is newly generated and has infrequent mentions in standard biomedical texts. Apart from the use of external knowledge, framing an NLP task as a question-answering task can lead to better performance (McCann *et al.*, 2018). Motivated by this approach, we hypothesize that a Knowledge guided QA framework may help biomedical NER.

Thus, in this paper, we focus on NER in biomedical text and test our hypothesis using different kinds of knowledge. The ways of expressing knowledge include asking a question about the entity, giving the entity type, providing a definition of the entity type, and mentioning some of its examples, as seen in Figure 2.1.

Figure 2.1 also shows how traditional NER systems formulate the problem as a classification task. This traditional task formulation leads to the following challenges: (a) *labeling error*, i.e., even though a system can identify the location of an entity correctly, it fails to predict the correct type; (b) inability to leverage more information for a particular entity type, since the conventional task formulation only allows to predict all entity types jointly; and (c) lack of labeled data for each entity type, especially in the biomedical domain. Challenges (a) and (b) are even more profound in the presence of nested named entities.

We can avoid challenges (a) and (c) by modeling the task as a multi-answer extraction task, where we predict only one type of entity at a time, given a context determining which entity is being extracted at the current time. This formulation allows us to avert the issue of nested named entities and learn from multiple biomedical datasets with similar entities. We specifically address the challenge (b) by providing five different knowledge contexts as shown in Figure 2.1. We perform an empirical study of which knowledge type has the most significant impact on the NER task.

Our task formulation enables us to create a considerably large dataset with knowledge context utilizing 18 biomedical datasets. The goal is to learn jointly from multiple domains containing different target entities. We also propose a new NER model over BERT using a re-contextualization layer, called BERT-CNN. This layer uses token-local features to recompute token encoding to enable the model to better understand the start and end locations of an entity. We use the BERT-base model


B-Test
I-Test
B-Problem
I-Problem
A CT scan with po contrast showed dilation of intrahepatic common bile duct which is improved from previous studies
I-Problem
B-Test
I-Test
B
I
B
I
A CT scan with po contrast showed dilation of intrahepatic common bile duct which is improved from previous studies
I
B
I

Text : A CT scan with po contrast showed dilation of intrahepatic common bile duct which is improved from previous studies
Context : Knowledge$_i$ + [Text]

Entity: *Test*
Question: What are the *test* mentioned in the text ?
Definition: *Test* is a procedure for critical evaluation
Example: blood, glucose, creatinine, hematocrit, blood-pressure
All: What are the *test* mentioned in the text ? *Test* is a procedure for critical evaluation. Eg: blood, glucose, creatinine, hematocrit, blood-pressure
Answer: A CT scan with po contrast, previous studies

Entity: *Problem*
Question: What are the *problem* mentioned in the text ?
Definition: *Problem* is difficulty, disorder, or condition needing resolution
Example: *Problem* + hypertension, pain, afebrile, nausea, fever
All: What are the *problem* mentioned in the text ? *Problem* is difficulty, disorder, or condition needing resolution. Eg: hypertension, pain, afebrile,...
Answer: dilation of intrahepatic common bile duct

Figure 2.1: The Top Block Shows the Traditional Way of NER. In Our Method, We Predict Only B, I, and O Tags for a given Context, i.e., Only Red Tags Are Predicted If the Context Is about an Entity Problem. O Tags Are Not Shown but Are Predicted for Non-answer Words.

and show our task formulation and model perform better than a strong baseline of a BERT-large model pre-trained on the biomedical corpus and finetuned using the traditional NER task. We perform extensive experiments to analyze the impact of each of our contributions. We also study the transfer learning ability of our knowledge-guided BERT-CNN model, as one of the significant challenges currently faced by the biomedical community is the low ability of the models to transfer to real-life applications (different writing styles and domains compared to training domain). The code, along with the data, has been made publicly available for further research[1].

To summarize our contributions:

- We reformulate the task of named entity recognition as a multi-answer question answering task using knowledge as a context.
- We make available a significantly large, cleaned, and pre-processed dataset with knowledge context utilizing 18 biomedical datasets having in total 398495 training, 148166 validation, and 502306 test samples.
- We propose a BIO tagging based model for the knowledge guided named entity recognition task, with a re-contextualization layer.
- We perform extensive experiments to evaluate our models, including the model's ability to adapt to new domains.
- Finally, all our contributions together further push the state-of-the-art exact match F1 scores by 1.78-12% for 11 publicly available biomedical NER datasets.

2.2 Related Work

External Knowledge: In the past, there have been several attempts to incorporate external knowledge through feature engineering and lexicons (Liu *et al.*, 2019;

[1] https://github.com/kuntalkumarpal/KGQA

Borthwick *et al.*, 1998; Ciaramita and Altun, 2005; Kazama and Torisawa, 2007), or incorporating knowledge in the feature extraction stage (Crichton *et al.*, 2017; Yadav and Bethard, 2018), or using document context (Devlin *et al.*, 2019). Our work incorporates simple textual knowledge sentences and shows how to integrate them into named entity recognition tasks.

Multi-Task Learning : Multi-task learning has been used in the past to tackle the labeling problem of NER. For example, multi-task learning with simple word embedding and CNN (Crichton *et al.*, 2017), cross-type NER with Bi-LSTM and CRF (Wang *et al.*, 2018), MTL with private and shared Bi-LSTM-CRF using character and word2Vec word embeddings (Wang *et al.*, 2019a). In our work, we do multi-task learning by reducing all different NER tasks to the same generic format and use transformer encoders. Our method uses a single encoder and span-prediction layer used for all tasks compared to existing multi-task methods that share a common encoder or layers and have different task-specific layers. The joint learning of multiple different NER tasks using a generic format makes our task formulation multi-task learning.

Language Models and Transfer Learning: There have been prior attempts to reduce the labeling confusion by using a single model to predict each entity type (Lee *et al.*, 2020) and using transfer-learning (Lee *et al.*, 2020; Beltagy *et al.*, 2019; Si *et al.*, 2019). Our work is similar to theirs, which also uses pre-trained language models (BERT) and predicts different types of entities separately, but differs in task formulation and explicit external knowledge context. We show jointly learned single model is better than a per entity-type model.

NER as a Question Answering Task: In the general domain, researchers have formulated multiple NLP tasks as a question-answering format in DecaNLP (McCann *et al.*, 2018), semantic-role labeling as in QASRL (He *et al.*, 2015) and others have argued that question-answering is a format, not a task (Gardner *et al.*, 2019). We also use QA format as a part of our task to address previously mentioned challenges. A possibly concurrent work, BERT-MRC (Li *et al.*, 2020) also attempts at NER as a QA task in the general domain by span predictions for individual entity types in a reading comprehension style approach. However, we differ in the task formulation using the BIO tagging scheme, our model design, and our focus on Biomedical NER. A detailed comparison is in Section 2.5.1.

BioMedical NER: In the Biomedical domain, CollaboNet (Yoon *et al.*, 2019) uses multiple expert models for each dataset collaborating to reduce the misclassification error, and NER uses variational dropout (Giorgi and Bader, 2020). Dictionary-based distantly supervised methods (Wang *et al.*, 2019b) have been proposed to reduce the need for human annotations but are still far from fully-supervised methods. Other methods use more nuanced approaches of adding a word and character-level features (Yadav *et al.*, 2018). The most common approach of using BiLSTM-CRFs (recurrent neural networks) for NER, has been applied to several biomedical or chemical applications, such as Medline indexing (Savery *et al.*, 2020), entity extraction for fMRI (Abacha *et al.*, 2017), postpartum depression detection (Chowdhuri *et al.*, 2019), and conversational agents (Amith *et al.*, 2020). Recurrent neural networks are prevalent for clinical and biomedical sequence labeling tasks such as NER (Wu *et al.*, 2020). A significant drawback to such approaches is the need for multiple models for each dataset. We hope our work can motivate the adaptation of KGQA and transformer encoders, which reduces the need for multiple models and improves overall task performance.

2.3 Method

2.3.1 Task Formulation

Traditional systems define named entity recognition as a multi-class classification task. Given a context $C = \{c_1, c_2, ..., c_n\}$, any token c_i is classified as one of the three tags B-e_k, I-e_k, O in the BIO-Tagging scheme, where $e_k \in E$ (the set of entity types for a dataset). For example, from Figure 2.1 a traditional NER method will predict, $B-Test$ for token "*A*", $I-Test$ for all the tokens in span "*CT scan with po contrast*", $B-Problem$ for "*dilation*" and $I-Problem$ for all the tokens in span "*of intrahepatic common bile duct*". This formulation leads to *labeling error*. A token c_i is classified as B-e_k or I-e_k when the token is actually a B-e_j or I-e_j where $j \neq k$. In the above example, a labeling error will arise if instead of $I-Test$ for all the tokens in span "*CT scan with po contrast*", even for one token the model predicts $B-Problem$, $B-Test$ or $I-Problem$. It means that even though a system could identify an entity's location correctly, it fails to identify the correct type.

In our approach, we tackle this issue by formulating the NER task in the following way. Given a context $C = \{c_1, c_2, ..., c_n\}$, any token c_i is classified as B, I and O. To identify which entity type the token belongs to, we provide external knowledge K to the context, containing the entity type information. For example, if we want to extract two entities e_1 and e_2 from context C, we first provide K_{e_1} and C as input to our model to extract e_1 entities, then provide K_{e_2} and C as input to extract e_2 entities. For example, in Figure 2.1 our method defines a set of knowledge contexts. Given the knowledge context "*Question*", if the user wants to extract "*Test*" entities, it provides an input "*What are the Test mentioned in the text?*" along with the input sentence. The model then predicts the spans using BIO tagging, with an expected output span of "*dilation of intrahepatic common bile duct*". If the user wants to extract the "*Problem*" entities, a different question is formulated "*What are the Problem mentioned in the text?*". Both the queries are given input to the same model, and the model is guided by the knowledge context provided to infer which type of entities to predict. This formulation decouples the classification and the entity location tasks, enabling the model to learn from multiple datasets and overcoming *labeling error*. Moreover, in comparison to other methods that learn per-entity classification models (Lee *et al.*, 2020), our task formulation trains only a single model to extract all entities. Similarly, our knowledge-context guided entity extraction is a single-step single model approach compared to a two-step method of first identifying entity locations using the BIO tagging model and then using a classification model to identify entity types for extracted spans (Yu *et al.*, 2020).

Comparison to Machine Reading Comprehension There is a difference between the "context" we refer to in our task formulation and the traditional machine reading comprehension task. In traditional machine reading comprehension, such as SQuAD (Rajpurkar *et al.*, 2016), the "context" refers to the paragraph text from which the answers are extracted for a given question. In our task formulation, the answers are extracted from the input text, whereas the "knowledge context" provides the relevant context or guidance to extract the particular entities' answers.

2.3.2 Knowledge Context Generation

We experiment with five types of knowledge context (K) to identify entities and their types. These are: (a) *Entity types* ($e_k \in E$) (b) separate *Question* (Q_k) created using each entity type, (c) *Definition* (D_k) of each entity type along with the entity type itself (e_k), (d) *Examples* (Eg_k) along with entity type (e_k) and (e) *All* of the above. If there are entities of n entity types in a text, we create a set of five knowledge contexts for each different entity type during training. Since the approach works one entity type at a time, we make sure that the entity type is mentioned in each of the five contexts. Only the best knowledge context is used during inference, i.e, if *Question* performs best for a dataset, we use only that context. For the example mentioned in Figure 2.1, E is {"*Problem*", "*Test*"} , Q is {"*What are the problem mentioned in the text?*", "*What are the test mentioned in the text?*"}, D is the definition text, {"*Problem is a difficulty, disorder, or condition needing resolution*", "*Test is a procedure for critical evaluation*"}, and Eg are the examples {"*hypertension, pain, afebrile, nausea, fever*", "*blood, glucose, creatinine, hematocrit, blood-pressure*"}. We analyze the impact of the different knowledge context formulation in section 2.5.2, including different question formats.

2.3.3 Datasets

We create the dataset for NER using fifteen publicly available biomedical datasets[2] (Crichton *et al.*, 2017) and three datasets from previous i2b2 challenges (Sun *et al.*, 2013; Uzuner *et al.*, 2011, 2010, 2012). One of the samples is shown in Figure 2.1. Our task formulation enables us to combine the datasets and create a significantly large dataset that enables deep neural model learning. Moreover, the multi-task learning for different entity types enables the model to generalize better. We also checked for overlap between samples of training and testing data. We find that the i2b2 datasets have a maximum overlap of 3.57%, whereas, among the remaining, the overlaps are much less than 1%. We remove the overlapping samples from the training data.

Bionlp Shared Task and Workshop: Six of the datasets Bionlp09 (Kim *et al.*, 2009), Bionlp11ID (Nguyen *et al.*, 2011), Bionlp11EPI (Nguyen *et al.*, 2011), Bionlp13PC (Nédellec *et al.*, 2013), Bionlp13CG (Nédellec *et al.*, 2013), Bionlp13GE (Nédellec

[2] https://github.com/cambridgeltl/MTL-Bioinformatics-2016

et al., 2013) are from the Biomedical Natural Language Processing Workshops. Some of these datasets' basic entities are gene or gene products, proteins, chemicals, and organisms.

i2b2 Shared Task and Workshop: We use three datasets from the i2b2 shared task and Workshop Challenges in Natural Language Processing for Clinical Data. We only use training and testing data from the 2010 Relations Challenge (Uzuner *et al.*, 2011), 2011 Coreference Challenge (Uzuner *et al.*, 2012), and 2012 Temporal Relations Challenge (Sun *et al.*, 2013). These datasets primarily contain entities like problems, tests, and treatments.

Bio-Creative Challenge and Workshop: These workshops provide datasets for information extraction tasks in the biological domain. We only use three datasets namely BC4CHEMD (Chemical) (Krallinger *et al.*, 2015), BC5CDR (Chemical and disease) (Wei *et al.*, 2015) and BC2GM (gene or protein) (Smith *et al.*, 2008). We consider these datasets since they are similar to biomedical texts and can be augmented to be trained together to generalize on the extraction of some of the entities.

Others: Apart from these 12 datasets we also include CRAFT (Bada *et al.*, 2012), AnatEM (Pyysalo and Ananiadou, 2014), Linnaeus (Gerner *et al.*, 2010), JNLPBA (Kim *et al.*, 2004), Ex-PTM (Pyysalo *et al.*, 2011) and NCBI-Disease (Doğan *et al.*, 2014) to increase our training and evaluation set. They include entities such as anatomy, species, diseases, cell-line, DNA, RNA, gene or protein and chemicals.

Entity Type Normalization: Since we intend to train together with 18 biomedical datasets, we find that there are entity types like *disease* in BC5CDR, which are a specific type of another entity *problem* in i2b2 datasets. However, we cannot map them together into any particular group since some entities are problems but not diseases for example *numb right leg*, *bleeding* or *slow to awake.* We cannot also separate them altogether because of the subtle distinction between such entity types. So, to avoid confusion and also to compare our approach with the current state-of-the-art results on each dataset, including single-task specific models, we do not normalize any entity types. However, we want to reap the benefits of training with more data of similar contexts (because of similar entity types), which we hypothesize will improve our model. 47 distinct entity types have been considered across all 18 datasets. Distribution of the entity types for each dataset is present in Table 2.3 and 2.4 for reference.

Data Preprocessing: We have done the experiments on 18 biomedical datasets available in different formats. We used the BIO annotated files for the 15 publicly available datasets and automatically extracted the spans based on the tags. The three i2b2 files have different format. They have individual biomedical reports containing multiple sentences that may or may not contain the entities. We preprocessed them by considering each statement as a sample without rejecting any sentence. Thus we bring all the datasets into a common format. Each sample in our preprocessed data contains the id(indicating the dataset: the sample origin), text, answers, span, number of answers present in the text, entity type, question context, definition of entities, and top ten frequently occurring examples with counts. We grouped similar entity types to form entity groups and add entity group definitions that can be used for further research.

2.3.4 Rule-based Template Creation

We use the following rules to create contexts for each knowledge type.

Entity: The first and the most specific context is the Entity type name (e_k) itself. We use this knowledge since it is the minimum text required to differentiate between multiple entities present in each dataset.

Question: We create a knowledge context Question (Q_k), using simple rules, like:

Q_k = What are the [e_k] mentioned in the text ?

We use this knowledge to see how much the query-guided format of knowledge with entity type helps the recognition task. This format simulates the recognition task as a question-answering task.

Definition: To get knowledge context (D_k), we find the corresponding scientific definition of each entity type from UMLS Meta-thesaurus (Bodenreider, 2004) by considering the entity type as a concept. Other sources include a definition of an entity type in the challenge dataset and online resources (Wikipedia). Few entity types are not directly available as concepts or have significantly vague or extensive definitions. To be precise and be under the limit of the maximum sequence length of 512, we search for other sources such as Wikipedia.

$D_k \in \{\text{UMLS} \mid \text{Challenge} \mid \text{Online Resource}\}$

This knowledge context is used since the definition helps us understand each entity type. We use definition as knowledge to help the NER model by instruction-based guidance. We use definitions like:

A **cell type** is a classification used to distinguish between morphologically or phenotypically distinct cell forms within a species. **Cancer** is a group of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body. **Proteins** are large biomolecules, or macromolecules, consisting of one or more long chains of amino acid residues.

Examples: To determine representative examples of an entity type, we find the top ten most frequent entities for each type from the entire training dataset. We concatenate the entity type ($[e_k]$) with these ten entities to generate the final knowledge context (Eg_k) and prepend this in front of the text.

We use this knowledge to provide an example-based guidance to the model.

All: is just the concatenation of all of the above knowledge contexts (e_k, Q_k, D_k, and Eg_k).

The motivation behind using the entity type and definition as knowledge context is that the neural model can leverage the information present to attend to correct entities. If we use the question as a context, the task becomes a multi-answer question-answering task. We use examples as knowledge with the hypothesis that our model will choose the entities that can belong to the same categories as the examples understanding its associations. We combine them to see if we can leverage the benefit for each of the four knowledge contexts together.

The distribution of each of the entities across each dataset for Training, Validation, and Test splits (both positive and negative samples) and more details about the dataset preparation can be found in Table 2.3 and 2.4.

We treat each sentence in a medical document or paragraph as an individual sample. If a sentence has an entity corresponding to a context, we consider it a positive sample for that context. Similarly, we treat a sentence that does not have an entity for a corresponding context as a negative sample for that context. However, these sentences can contain entities for other entity types. Since many datasets do not provide a validation split, we randomly sample from the train split to create our validation data. Overall, our dataset has 398495 train, 148166 dev, and 502306 test samples.

2.3.5 Knowledge Guided NER Model

We choose the BERT-base cased version (Devlin *et al.*, 2019) as our base model. In our approach, given a text C, we create a knowledge context K_i for each context type. We need to find the spans of entities S_{start} and S_{end}. So, we define the input to the BERT model as follows, the knowledge context tokens $K_i = \{k_{ij}\}$ are prepended to the text tokens, $C = \{c_j\}$. The sequence of tokens, $\{[CLS], k_{i1}, ..k_{im}, [SEP], c_1, ..c_n, [SEP]\}$ is given as input to the BERT model where m is the size of knowledge context K_i

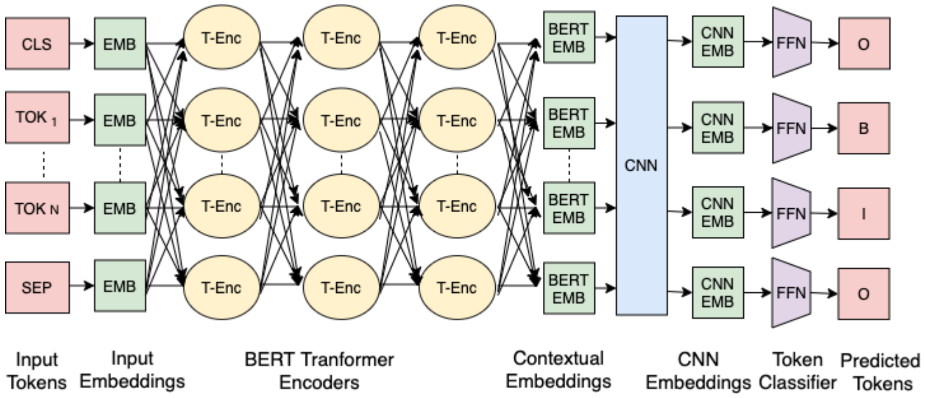


Figure 2.2: BERT-CNN for Multi-Answer KGQA

and n is the size of text C. For each token in our baseline model, we predict B, I, and O using a feed-forward layer.

2.3.6 Re-contextualization

We modify the BERT-base model by adding a re-contextualization layer consisting of a two-dimensional convolution layer. This layer aims to leverage information from adjacent or local token embeddings and help in the better start and end tag prediction of the entities. As BERT uses multiple layers of attention that jointly focus on all the tokens, we add this CNN layer with a window of $W < 5$ to focus on neighbor tokens only. We take the CNN layer's outputs and feed them to the final feed-forward layer to predict the start and end tags. Figure 2.2 represents our BERT-CNN model's end-to-end architecture.

2.3.7 Training and Testing

During training, the context, X (combination of knowledge, K_i and given text, C) has gold annotations (y_i) of B, I and O for each token (x_i). We calculate the cross-entropy loss for each token x_i as:

$$L_{token} = -\sum_{c=1}^{M} y_{x_i,c} log(P_{x_i,c})$$

where M is the total number of classes (B, I, O), $y_{x_i,c}$ is a binary indicator of whether the label c is the correct classification of a token x_i, $P_{x_i,c}$ is the predicted probability of x_i belonging to class c.

The model is trained end-to-end with the above loss. During inference, we consider the tokens x_i present only in the text C. Text chunks that start from label B and continue till the last I tag are predicted as entities. We feed the text with a different context and text input for each entity type. We train our model jointly on a processed, combined dataset of 18 biomedical datasets and compare the performance when trained individually.

2.4 Experiments

2.4.1 Experimental Setup and Training Parameters

We use a batch size of 256 and a learning rate of 5e-5 for all our experiments. The maximum sequence length of 128/256 depends on the 99th percentile of the input token lengths. We train using 4 NVIDIA V100 16GB GPUs, with a patience of 5 epochs. We report the mean F1 scores for three random seeds; the deviation is also reported. We apply a two-dimensional convolution layer on top of BERT contextual token embeddings for the BERT-CNN model. The convolution layer uses a 5×5 size kernel. The stride size is (1,2), where one is across sentence dimension, and two is across word embedding dimension. We also perform circular padding.

We use the HuggingFace (Wolf *et al.*, 2019), and PyTorch Deep learning framework (Paszke *et al.*, 2019). We train the model with the following hyperparameters, learning rates in the range [1e-6,5e-5], batch sizes of [16,32,48,64, 256], linear weight-decay

DATASET	F-MEASURE				
	T	Q	D	E	A
ANATEM	88.49 ± 0.52	89.50 ± 0.91	88.44 ± 0.94	89.97 ± 0.41	**90.94** ± 0.31
BC2GM	82.11 ± 0.32	83.01 ± 0.49	81.65 ± 0.77	82.25 ± 0.22	**83.47** ± 0.27
BC4CHEMD	90.02 ± 0.56	91.84 ± 0.79	89.44 ± 0.79	90.43 ± 0.22	**92.39** ± 0.13
BC5CDR	87.92 ± 0.54	90.02 ± 0.99	87.74 ± 0.84	88.42 ± 0.33	**90.50** ± 0.24
BIONLP09	89.16 ± 0.29	91.51 ± 0.55	58.21 ± 0.42	89.39 ± 0.41	**91.94** ± 0.32
BIONLP11EPI	85.82 ± 0.69	87.96 ± 1.03	80.84 ± 0.53	86.32 ± 0.45	**88.66** ± 0.27
BIONLP11ID	83.93 ± 0.99	86.14 ± 1.63	83.09 ± 1.24	84.12 ± 0.69	**87.36** ± 0.43
BIONLP13CG	85.32 ± 0.89	88.51 ± 0.57	84.90 ± 1.45	87.82 ± 0.79	**90.16** ± 0.21
BIONLP13GE	82.05 ± 0.79	85.40 ± 0.82	74.86 ± 0.87	82.54 ± 0.70	**85.81** ± 0.64
BIONLP13PC	89.36 ± 0.37	90.97 ± 0.54	88.63 ± 0.75	90.11 ± 0.25	**91.65** ± 0.27
CRAFT	85.81 ± 0.43	88.27 ± 0.65	83.97 ± 0.87	88.94 ± 0.21	**90.21** ± 0.19
EXPTM	84.19 ± 1.07	85.75 ± 1.07	78.76 ± 0.69	84.80 ± 0.77	**87.08** ± 0.36
JNLPBA	74.06 ± 0.28	**79.19** ± 0.64	67.87 ± 0.90	74.76 ± 0.34	78.88 ± 0.21
LINNAEUS	88.39 ± 0.69	89.71 ± 1.37	87.66 ± 1.13	90.70 ± 0.83	**92.63** ± 0.33
NCBIDISEASE	88.76 ± 0.51	**89.82** ± 1.08	88.67 ± 0.76	88.21 ± 0.56	89.72 ± 0.42
2010-i2b2	89.93 ± 0.11	92.63 ± 0.22	89.92 ± 0.29	90.22 ± 0.05	**92.67** ± 0.06
2011-i2b2	90.37 ± 0.18	92.67 ± 0.29	90.40 ± 0.32	91.97 ± 0.07	**93.68** ± 0.05
2012-i2b2	79.34 ± 0.86	**83.98** ± 1.03	79.48 ± 1.32	78.11 ± 1.21	83.12 ± 0.25

Table 2.1: F-measure of BERT-CNN Model Using Different Knowledge Types: Entity Type (T), Question (Q), Definition (D), Examples (E), and All of Them Together (A). The Best Scores Are in Bold, Second Best Is Underlined. The Mean of Ten Random Seed Runs Is Reported along with the Standard Deviations.

in the range [0.001,0.1] and warm-up steps in the range of [100,1000]. We use the BERT-base-cased version for all our models. The BERT-base-cased model has nearly 110M parameters.

2.4.2 Entity Distribution in the Dataset

The number of samples present in Table 2.3 and 2.4 is created directly from the train, validation, and test samples of 18 biomedical datasets. The i2b2 datasets do not have separate validation data splits. We use 30% of the samples from training data as validation data. The *Entity Mentions* represents the total number of entities present for the datasets in all train, validation, and test samples. Since each sample data

DATASET	PRECISION					RECALL				
	T	Q	D	E	A	T	Q	D	E	A
ANATEM	88.70 ± 0.47	89.72 ± 0.77	88.90 ± 0.66	90.13 ± 0.42	**91.15** ± 0.36	88.28 ± 0.82	89.28 ± 1.10	87.98 ± 1.27	89.80 ± 0.46	**90.74** ± 0.31
BC2GM	81.78 ± 0.28	82.55 ± 0.36	81.55 ± 0.62	82.00 ± 0.26	**83.12** ± 0.29	82.44 ± 0.56	83.48 ± 0.70	81.74 ± 0.95	82.49 ± 0.31	**83.82** ± 0.41
BC4CHEMD	90.53 ± 0.41	92.23 ± 0.62	90.08 ± 0.39	91.05 ± 0.19	**92.77** ± 0.18	89.52 ± 0.74	91.46 ± 0.99	88.80 ± 1.22	89.83 ± 0.26	**92.02** ± 0.27
BC5CDR	88.22 ± 0.29	90.48 ± 1.01	88.18 ± 0.59	88.86 ± 0.19	**90.96** ± 0.21	87.63 ± 0.87	89.57 ± 1.00	87.31 ± 1.14	87.99 ± 0.52	**90.04** ± 0.30
BIONLP09	88.95 ± 0.33	90.83 ± 0.53	50.04 ± 0.66	89.78 ± 0.35	**91.62** ± 0.25	89.36 ± 0.41	92.20 ± 0.64	69.58 ± 0.59	89.00 ± 0.64	**92.26** ± 0.52
BIONLP11EPI	87.46 ± 0.46	88.79 ± 0.87	78.74 ± 0.65	87.77 ± 0.36	**89.71** ± 0.35	84.25 ± 1.11	87.14 ± 1.22	83.06 ± 0.72	84.91 ± 0.80	**87.64** ± 0.43
BIONLP11ID	84.88 ± 1.43	86.74 ± 1.47	83.79 ± 1.89	85.00 ± 0.90	**87.91** ± 0.49	83.02 ± 1.16	85.56 ± 1.94	82.42 ± 1.33	83.26 ± 0.59	**86.81** ± 0.57
BIONLP13CG	87.11 ± 2.21	89.92 ± 0.82	86.61 ± 3.07	88.91 ± 1.66	**91.19** ± 0.19	83.64 ± 0.57	87.15 ± 0.59	83.31 ± 0.75	86.78 ± 0.37	**89.15** ± 0.30
BIONLP13GE	79.45 ± 0.98	83.03 ± 0.78	70.27 ± 0.99	79.98 ± 0.92	**83.33** ± 0.71	84.84 ± 1.23	87.91 ± 0.98	80.10 ± 1.29	85.28 ± 1.25	**88.44** ± 0.74
BIONLP13PC	89.13 ± 0.52	90.37 ± 0.73	88.42 ± 0.97	90.21 ± 0.51	**91.20** ± 0.31	89.60 ± 0.39	91.58 ± 0.55	88.84 ± 0.77	90.00 ± 0.35	**92.11** ± 0.31
CRAFT	86.08 ± 0.55	88.29 ± 0.47	82.94 ± 0.93	89.02 ± 0.47	**90.71** ± 0.16	85.55 ± 0.83	88.25 ± 0.95	85.03 ± 1.01	88.86 ± 0.39	**89.71** ± 0.37
EXPTM	83.86 ± 0.93	84.97 ± 0.96	74.99 ± 0.73	84.59 ± 0.62	**86.57** ± 0.36	84.53 ± 1.55	86.55 ± 1.34	82.93 ± 1.12	85.01 ± 1.15	**87.59** ± 0.53
JNLPBA	71.11 ± 0.53	**76.95** ± 0.66	66.36 ± 1.02	71.41 ± 0.83	76.87 ± 0.32	77.26 ± 0.35	**81.57** ± 0.67	69.48 ± 1.84	78.46 ± 0.33	81.00 ± 0.19
LINNAEUS	90.17 ± 1.30	90.62 ± 1.72	89.27 ± 1.71	92.70 ± 1.72	**93.51** ± 0.59	86.69 ± 0.63	88.75 ± 1.49	86.11 ± 1.04	88.80 ± 0.42	**91.76** ± 0.31
NCBIDISEASE	87.21 ± 0.74	**88.29** ± 1.17	87.18 ± 0.71	86.37 ± 0.70	87.98 ± 0.53	90.37 ± 0.37	91.42 ± 1.14	90.22 ± 1.05	90.13 ± 0.72	**91.52** ± 0.44
2010-i2b2	88.88 ± 0.10	91.75 ± 0.25	88.76 ± 0.31	89.21 ± 0.10	**91.90** ± 0.07	91.00 ± 0.21	91.52 ± 0.19	91.10 ± 0.35	91.24 ± 0.11	**93.44** ± 0.06
2011-i2b2	90.12 ± 0.12	92.39 ± 0.36	90.04 ± 0.30	91.85 ± 0.13	**93.45** ± 0.04	90.61 ± 0.28	92.96 ± 0.23	90.76 ± 0.39	92.10 ± 0.10	**93.89** ± 0.07
2012-i2b2	78.38 ± 1.87	**84.84** ± 1.48	78.69 ± 2.70	76.87 ± 2.69	83.74 ± 0.50	80.33 ± 0.29	**82.41** ± 0.61	80.29 ± 0.45	79.41 ± 1.27	82.34 ± 0.17

Table 2.2: Precision and Recall of BERT-CNN model using different knowledge types: Entity Type (T), Question (Q), Definition (D), Examples (E), and all of them together (A). The best scores are in bold, second best is underlined. The mean of ten random seed runs is reported along with the standard deviations.

can have multiple entities, the number is higher than the total positive and negative samples for the dataset.

2.4.3 Baseline Models

We consider the following models as strong baselines for our work. The first set of baselines is the BERT models pre-trained on biomedical text BioBERT (Lee *et al.*, 2020) and MimicBERT (Si *et al.*, 2019) finetuned using traditional NER task. BioBERT and MimicBERT are the current state-of-the-art (SOTA) models for NER on multiple biomedical datasets. The second set of baselines is BioBERT, MimicBERT, and BERT-MRC finetuned on the knowledge-guided NER task. BERT-MRC is initialized with BioBERT base weights, the same as our BERT-CNN model. BERT-MRC model is another concurrent query-driven NER model (Li *et al.*, 2020), that models the task as a machine reading comprehension task. It predicts all possible start and end positions and predicts valid start-end spans through another feed-forward layer that takes input from the predicted start-ends. This model shows considerable improvements in general domain query-based NER tasks. The above baselines are trained on individual datasets as each dataset has a separate set of entities. Furthermore, we train another baseline using BioBERT, TradBioBERT, to combine all the entities to utilize the joint dataset but follow the traditional NER task. More details about this baseline are present in section 2.5.2.

2.5 Results and Discussion

2.5.1 Biomedical NER

Table 2.5 compares our method with our baselines on the 18 biomedical NER datasets. Our methods use the best knowledge context identified on the validation set performance. Current state-of-the-art for AnatEM and Linnaeus use specific lexicons and entity-specific rules that do not generalize and are not directly comparable to neural models, although our methods approach their performance. On BC4CHEMD and BC5CDR, the state-of-the-art methods are BERT-Base models fine-tuned specifically on chemical and other science corpora, whereas our methods use BioBERT as the backbone. Still, our models are within the 1% F1 score. 2011-i2b2 does not have a task specific to NER. Therefore, it does not have current state-of-the-art methods but still has annotations for the named entities we use for joint training. On the remaining 11 datasets, we achieve state-of-the-art using BERT-Base and beat methods that use BERT-Large. The small difference in performance on JNLPBA and NCBI-Disease datasets with the state-of-the-art is not statistically significant to be considered an improvement. The state-of-the-art scores are F1 values from multiple works (Crichton *et al.*, 2017; Si *et al.*, 2019; Lee *et al.*, 2020; Beltagy *et al.*, 2019).

Our task formulation gives a significant boost in performance, which is observed in the improvements made by our BioBERT and MimicBERT base models compared to the baseline models following traditional NER formulation. Our BERT-CNN model further improves performance over the BioBERT knowledge-guided QA model on 12 tasks with a margin of 0.5 to 2.2% (173 - 1934 samples). When it under-performs, it is within a margin of 0.5% (less than 100 samples).

2.5.2 Ablation Studies and Analysis

DATASET	**P**	**ΔP**	**R**	**ΔR**	**F**	**ΔF**
ANATEM	87.34	**-2.38**	89.31	0.03	88.31	**-1.19**
BC2GM	79.91	**-2.64**	81.63	**-1.85**	80.76	**-2.25**
BC4CHEMD	92.56	0.33	91.10	-0.36	91.82	-0.02
BC5CDR	87.67	**-2.81**	90.13	0.56	88.88	**-1.14**
BIONLP09	86.92	**-3.91**	84.91	**-7.29**	85.90	**-5.61**
BIONLP11EPI	84.57	**-4.22**	86.97	-0.17	85.75	**-2.21**
BIONLP11ID	87.98	1.24	84.64	-0.92	86.27	0.13
BIONLP13CG	84.01	**-5.91**	80.84	**-6.31**	82.39	**-6.12**
BIONLP13GE	72.33	**-10.7**	86.44	**-1.47**	78.76	**-6.64**
BIONLP13PC	86.40	**-3.97**	87.21	**-4.37**	86.8	**-4.17**
CRAFT	86.35	**-1.94**	85.65	**-2.60**	86.00	**-2.27**
EXPTM	75.28	**-9.69**	81.51	**-5.04**	78.27	**-7.48**
JNLPBA	76.85	-0.10	81.79	0.22	79.24	0.05
LINNAEUS	91.28	0.66	86.15	**-2.60**	88.64	**-1.07**
NCBIDISEASE	83.86	**-4.43**	87.25	**-4.17**	85.52	**-4.30**
2010-i2b2	89.87	**-5.19**	90.75	**-5.06**	90.31	**-5.12**
2011-i2b2	91.49	**-2.64**	92.25	**-2.02**	91.87	**-2.33**
2012-i2b2	82.05	**-3.00**	82.31	**-3.35**	82.18	**-3.17**

Table 2.6: Change in Performance When BERT-CNN Model Is Trained Individually on Respective Datasets with **Question** Context. Negative Δ for Precision (P), Recall (R), and F-measure (F) Indicates Multi-task Is Better. $\Delta < 1$ (Significant Difference) Are Marked Bold.

Effect of different knowledge contexts: Since we incorporate five different knowledge contexts to help in the NER task, we identify which knowledge context is better across all the 18 datasets. The BERT-CNN model's performance with the knowledge contexts across the test set is shown in Table 2.1 and 2.2. The values are

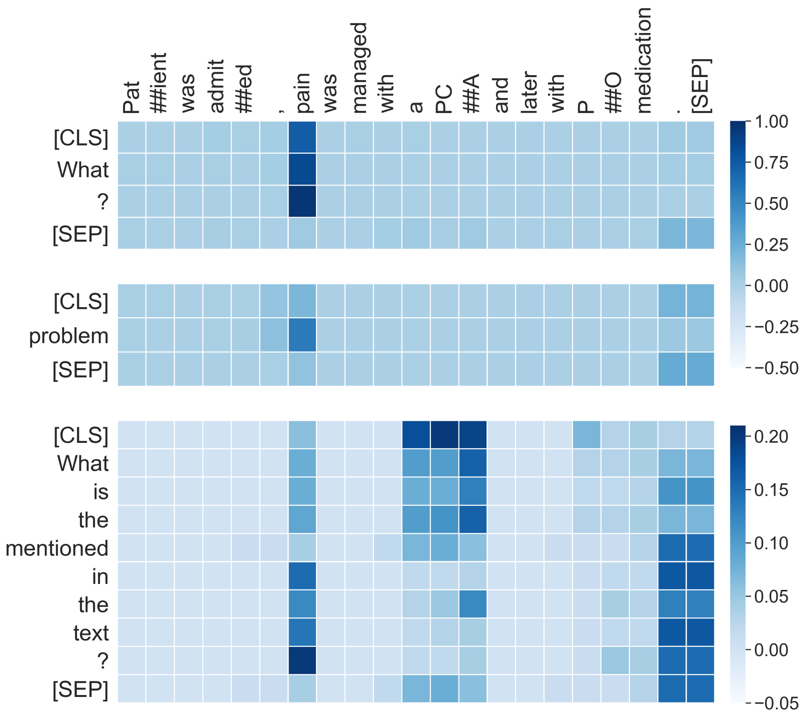


Figure 2.3: Attention Scores of Our BERT-CNN Model Trained with *Question* Context for Three Knowledge Context Probes, "What ?", "problem" and a Context Where We Remove the Entity Type. For the Last, the Model Attends to All Three Entities Present **pain** (*problem*), **PCA**, and **PO medication** (*treatment*).

entity precision, recall, and exact match F1 scores. We observe *Question* and *All* contexts perform consistently better on all the datasets. We believe this is because of the "what" token that helps the model find entities much better than just a text. To verify our hypothesis, we probe our BERT-CNN model trained with *Question* context with multiple probes like "what problem?", the complete question, and "problem" for 20 samples and observe the change in attention scores. As the model is trained with a template, the best prediction is observed on the complete question, with the least scores for only the entity type. Attention scores for "what" were consistently high. Figure 2.3 shows examples of such probes. We can observe our model identifies all the

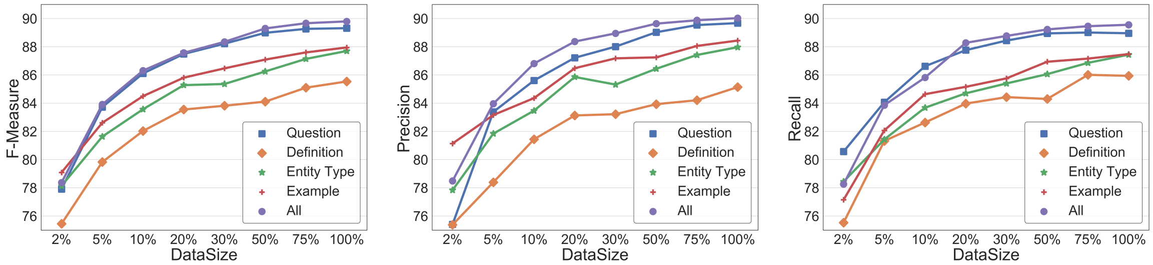


Figure 2.4: Effect of Train Data Size on Validation Data Using BERT-CNN Model Trained and Validated Individually with Five Contexts.

entities and the entity-type acts as a filter. *Definition* and *Examples* under-perform, we believe the definition might be too generic for an entity type and examples, although representative of class might not be comprehensive.

Contributions of Question words: To further understand the contributions of "what", we train the BERT-CNN model with different versions of *Question* context like "What <entity_type> ?" and "What <entity_type>". We observe that the F1 performance for these two variations is similar (within 1%), whereas they underperform by 1% to 5% when compared to original question contexts. This experiment shows that our well-formed question as knowledge context enhanced the model's performance. We also observe that certain words are repeated for each training sample only for the question context, i.e., "What are the" and "in the text?". In all other contexts, including only entity type, the context words vary given a sample. For example, the definition and examples have a different set of words unique to that entity. We hypothesize the difference in performance might be related to the presence of such common repeated words and the model is storing discriminative information between different entity types in the embeddings of these repeated words, and given the entity type can retrieve the relevant information from the weights leading to better performance. This effect is similar to the [CLS] token effect observed in BERT-based transformer encoders (Devlin *et al.*, 2019), where the discriminatory classification features are stored in the [CLS] token. This can be observed with our experiments with only "<entity_type>", "What <entity_type>", "What <entity_type> ?" and the full question with a pattern of overall increased performance. It can also be observed in Figure 2.5 where all the common words attend strongly to the predicted entity type, whereas the word "mentioned", which is not used for the question template for a few entities such as "Person" does not attend strongly to the entities but only to the separator token.

Effect of Multi-task training: To analyze how much the multi-task training strategy affects the performance, we define an experiment where we keep the model (BERT-CNN) and the knowledge context *Question* the same, but train on each individual dataset and compare with joint training. Table 2.6 shows the results of our experiments. Individual dataset training improves F1 scores marginally (max 0.13%) on only two datasets, whereas joint training substantially improves performance (0.02% - 7.48%) on the 16 remaining datasets. This result empirically validates our hypothesis that training on combined huge biomedical datasets helps. The datasets like Bionlp11ID and JNLPBA, with unique entities, do not show much difference on further analysis. The datasets that improve significantly have common entity types like Gene/Protein, Chemical, Disease, Problem, Treatment, and Test that helps the model generalize well and learn better representations. We noticed that an entity's definition was consistent when it was present in multiple datasets. It ensured that the model was not confused by different definitions. On the other hand, some identical entities had different names in different datasets but cannot be normalized into the same entity group. For example, the disease was called Problem in i2B2, Disease in NCBI-Disease, and by a more specific name Cancer in Bionlp13CG, but some entities are problems but not specifically diseases.

Effect of Knowledge Context compared to Individual entity training: In this experiment, we study the effect of knowledge context on our task formulation. We compare our single BERT-CNN model trained with *Question* context to six different BERT-CNN models, each trained only for one specific entity type detection without any context. The entity types are selected such that they are present in multiple datasets and have the most significant number of samples. Table 2.7 summarizes the results. The results show that knowledge context and training jointly with multiple entity types help improve all entity types' performance compared to individual entity-specific models. The performance improvement for Gene/Protein is the most significant.

ENTITY↓	**PRECISION**		**RECALL**		**F-MEASURE**	
	No-K	**K**	**No-K**	**K**	**No-K**	**K**
Gene/Protein	67.33	**84.19**	74.01	**86.73**	70.51	**85.44**
Chemical	89.04	**91.84**	88.42	**91.15**	88.73	**91.49**
Disease	83.17	**83.48**	86.62	**88.00**	84.86	**85.68**
Problem	91.95	**93.28**	92.83	**94.18**	92.39	**93.73**
Treatment	91.78	**92.91**	91.98	**93.47**	91.88	**93.19**
Test	92.12	**94.09**	93.23	**94.67**	92.67	**94.38**

Table 2.7: Comparison of Entity Specific (No-K) BERT-CNN Model with Question Context Provided Multi-entity BERT-CNN Model (K). Better Values Are in Bold.

Effect of Train Set Size: In this experiment, we study how the train set size affects the model performance. We sample different percentages of training samples from the total train dataset as seen in Figure 2.4. We ensure a balanced sampled train set with an equal number of positive and negative samples and evaluate across all entities. The training samples are chosen from each of the datasets to ensure the model is not biased towards a dataset or entity type. We do not change any parameters of the model. It can be observed that the performance of the model increases rapidly and then tapers down for each of the context types. We can infer that the model can achieve quite a good performance (84%) with just 5% of training samples but needs much more samples to achieve state-of-the-art performance. As training samples go beyond 5%, the precision, recall, and the F1-scores for knowledge types *Question* and *All* separate themselves from the other contexts.

Transfer Learning: We also examine our system's transfer learning capability on six entity groups: gene/protein, chemical, disease, problem, test, and treatment, since they are present in multiple datasets. We map *chemical* and *simple chemical* to generic chemical group and *amino acid*, *gene*, *gene or gene product* and *protein* to generic protein group. For each group type, we select only those datasets where they appear. Out of the selected dataset, we test our model on samples of one dataset (target domain) for each of the six entity types while training and validating the remaining selected datasets (source domain). We only keep samples for these entity types in the train, dev, and test data. We choose the target domain for each entity to be the dataset that produces the best overall F1 score on the entire data with the BERT-CNN model. We consider Bionlp11ID, BC4CHEMD, and BC5CDR datasets as the target domain for gene/protein, chemical, and disease, respectively, and 2010-i2b2 for the problem, treatment, and test. Table 2.8 summarizes the results.

The results show a varied degree of transfer learning, losing only 0.5% F1 in some tasks and by as much as 20% in other tasks, which is in line with earlier observed performance loss (Bethard *et al.*, 2017).

The difference in source and target F1-scores is remarkably close for Problem, Treatment, and Test entities. Although the two domains (text style and topics) are close for these entities, they have different entities. Chemical shows a significant drop, but still, our model achieves 75% F1 despite the target domain containing many prior unseen entities. For Gene/Protein, the source and target domain are nearly the same set of entities.

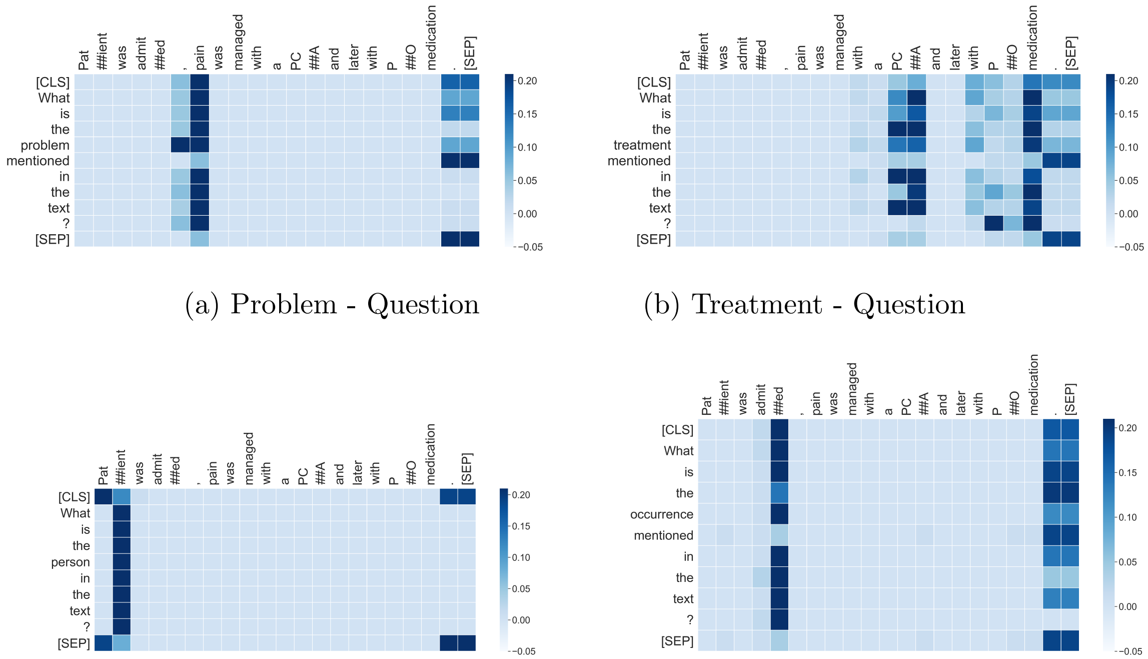
Training in Traditional NER approach with Combined Training Data: We also train Biobert (TradBioBERT in Table 2.5) in the traditional way of named entity recognition on the entire training data and without any external knowledge context. We use B_{Ent}, I_{Ent} tags for each of the 47 named entities present in the full dataset along with the O tag (others), resulting in a total of 95 target classes. It is trained as a multi-entity sequence classification task. The results are low as expected. We find the maximum F1-score of 78.22 and 75.25 for BC4CHEMD and NCBIDisease, respectively, followed by BIONLP11EPI, BC5CDR, ExPTM, BC2GM, 2010-i2b2, and 2011-i2b2 of 52.01 and below. The remaining datasets report a significantly lower F1-value of 13.89 and below. This low performance can be attributed to significantly increased confusion due to many target entity classes, leading to a rise in labeling errors. The poor results validate our approach of using generic B-ANS, I-ANS, and O instead of using BI tags for each entity type.

2.5.3 Nested Named Entity Recognition

We evaluate our KGQA model with a nested named entity dataset GENIA (Kim *et al.*, 2003). The results are present in Table 2.9. We can observe the *All* knowledge context outperforms previous methods, including specific graph-based methods that use state-of-the-art transformer encoders such as BERT. Moreover, our method uses the BERT-base version with 110M parameters, whereas all the previous methods use BERT-large with 340M parameters. This observation strongly supports our hypothesis that a simpler task formulation reduces labeling error and confusion. Table 2.10 shows predictions of our model with “all” as knowledge context.

2.5.4 Model Explanation with Attention Probing

We study our BERT-CNN model using attention-value heatmaps and try to explain how our model uses the knowledge contexts to extract specific entities from a given dataset. To show this, we choose a sample “*Patient* was *admited* , *pain* was managed with *a PCA* and later with *PO medication*.” where there are multiple entities **pain** (problem), **a PCA** and **PO medication** (treatment), **Patient** (person) and **admitted** (occurrence).



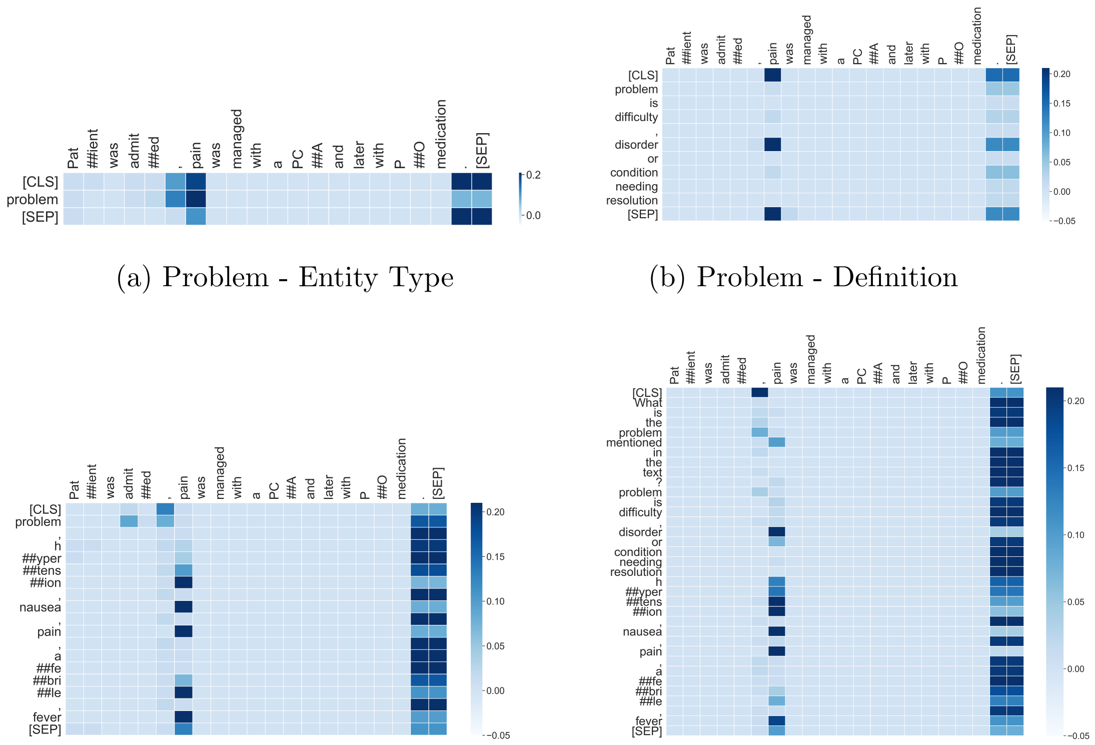
(a) Problem - Question

(b) Treatment - Question

(c) Person - Question

(d) Occurrence - Question

Figure 2.5: Attention Probes for Different Knowledge and Entity Types.



(a) Problem - Entity Type

(b) Problem - Definition

(c) Problem - Example

(d) Problem - All Knowledge

Figure 2.6: Attention Probes for Different Knowledge and Entity Types.

Using Question as Knowledge Context

From the Figures 2.5a, 2.5b, 2.5c and 2.5d it can be observed that when the knowledge context is in a question format then, each of the entity types present in the knowledge context guide the model to choose the correct entities. It can be observed from the higher attention values for those specific entities.

Using Entity Type, Definition, Example and "All" as Knowledge Context

We also probed our model to extract the attention weights for each of the other four knowledge contexts. Here we show this only for the problem entity type. In Figure 2.6a it can be seen that attention weight between *problem* in the knowledge context and *pain* in the text is highest. Figure 2.6b shows keywords like *disorder* in the definition representing the meaning of the entity type problem highly attends to the entity *pain*. On using the example as knowledge, it can be seen from Figure 2.6c, keywords like hypertension, pain, nausea, and fever highly attend to the entity pain in the text. This observation is in line with our hypothesis that providing similar entities belonging to the same entity group might help find the entity in a text. Finally, in Figure 2.6d, it can be seen that the keywords from the question, definition, and example all collectively help to predict the entity pain in the text.

2.5.5 Error Analysis

We also perform error analysis at a token level for each dataset for *all* knowledge context across the datasets with the BERT-CNN model. We notice that all the top five datasets where the model made maximum token prediction errors BioNLP13GE(20.98%), BC2GM (17.79%), ExPTM (17.07%), BioNLP11EPI (15.54%) and BioNLP09 (12.75%) have single entity type *Gene or protein*. It indicates the difficulty in predicting all the tokens of protein entities. For Linnaeus, the model made 30.27% of the errors due to the misclassification of tags (B instead of I and vice-versa). We found 54.41% of errors in BC4CHEMD are because of the model predicting more tokens than actually present, and 59.79% of errors of NCBIDisease are caused by the model predicting less number of tokens than actually present.

A few examples are shown in Table 2.11. In example 8, the model mistakenly identified VP22 and tegument protein as separate protein entities. In example 9, the model also considers the associated entities Lutz & Neiva. These errors indicate that span location identification still poses a significant challenge in a few cases and provides scope for further improvement.

2.6 Miscellaneous Experiments

2.6.1 Performance Comparison on Test Data

Table 2.12 shows the performance comparison of our best model with the state-of-the-art on test data of 18 datasets. The F-measures (bold) are the best performance on each dataset. The state-of-the-art for Linnaeus and AnatEM datasets use dictionaries developed without a clear train/test split; hence our scores are not directly comparable. Also, 2011-i2b2 does not have state-of-the-art concept extraction performance. Improvements in 10 datasets are significant as compared to the state-of-the-art.

Dataset	**Entities**	**SOTA F1**	**KGQA P**	**KGQA R**	**KGQA F1**	**Significant**	**KGQA ± Confidence Interval**
ANATEM	4616	**91.61**	90.29	89.43	89.85 ± 0.48	No	0.82
BC2GM	6322	81.69	82.89	83.39	**83.14 ± 0.54**	Yes	0.91
BC4CHEMD	25331	**92.36**	92.56	91.10	91.82 ± 0.58	No	0.32
BC5CDR	9808	**90.01**	90.09	89.62	89.62 ± 0.71	No	0.58
BIONLP09	3589	84.20	91.55	92.95	**92.25 ± 0.57**	Yes	0.89
BIONLP11EPI	5730	78.86	88.58	87.40	**87.99 ± 1.10**	Yes	0.82
BIONLP11ID	3810	81.73	87.98	84.64	**86.27 ± 1.80**	Yes	1.05
BIONLP13CG	7861	78.90	90.62	88.56	**89.58 ± 0.68**	Yes	0.65
BIONLP13GE	4354	78.58	83.77	88.01	**85.84 ± 0.93**	Yes	1.03
BIONLP13PC	5306	81.92	90.14	92.09	**91.11 ± 0.12**	Yes	0.74
CRAFT	18770	79.56	90.54	89.19	**89.86 ± 0.55**	Yes	0.42
EXPTM	2308	74.90	85.97	85.30	**85.64 ± 0.61**	Yes	1.36
JNLPBA	8673	78.58	76.85	81.79	**79.24 ± 0.45**	No	0.85
LINNAEUS	1428	**95.68**	90.69	90.53	90.61 ± 0.28	No	1.35
NCBIDISEASE	956	89.36	87.89	91.56	**89.69 ± 0.37**	No	1.92
2010-i2b2	30140	90.25	95.27	95.91	**95.59 ± 0.30**	Yes	0.29
2011-i2b2	25271	-	94.70	94.94	94.82 ± 0.41	-	0.30
2012-i2b2	15301	80.91	84.83	85.25	**85.04 ± 1.18**	Yes	0.59

Table 2.12: Precision(P), Recall(R), and F-measure(F1) Scores for Our Best Model Were Measured by Running with Ten Seed Values. The Significant Column Shows Whether Our F1 Scores Are Statistically Significantly Better than State-of-the-art F1 (95 % Confidence Interval Based on Wilson Score Intervals (Wilson, 1927)). Best F-measures Are in Bold.

2.6.2 Training with Balanced Dataset

We generated a sample for each text available in the source data. The text may or may not contain a particular entity. So we generate negative samples for each text and each available entity type of the dataset, making the datasets unbalanced. Table 2.13 shows that the negative samples do not impact our models' performance. The results are taken using the BERT-CNN model with *Question* as a context. Negative values in ΔP,ΔR, and ΔF means training on unbalanced data is better than on balanced data.

2.7 Conclusion

We reformulated the NER task as a knowledge-guided, context-driven QA task and showed that it leads to better F scores. Our models using query-text attention are more explainable and address some of the significant challenges faced by current NER systems. Our approach has achieved the above state-of-the-art F measures for 11 of the common public biomedical NER datasets. In the future, we plan to perform more experiments, such as few-shot learning between different entity groups, and adding specific loss functions and logical constraints for NER tasks.

REFERENCES

Abacha, A. B., A. G. S. de Herrera, K. Wang, L. R. Long, S. Antani and D. Demner-Fushman, "Named entity recognition in functional neuroimaging literature", in "2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)", pp. 2218–2220 (IEEE, 2017).

Amith, M., L. Cui, K. Roberts and C. Tao, "Towards an ontology-based medication conversational agent for PrEP and PEP", in "Proceedings of the First Workshop on Natural Language Processing for Medical Conversations", pp. 31–40 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.nlpmc-1.5`.

Bada, M., M. Eckert, D. Evans, K. Garcia, K. Shipley, D. Sitnikov, W. A. Baumgartner, K. B. Cohen, K. Verspoor, J. A. Blake *et al.*, "Concept annotation in the craft corpus", BMC bioinformatics **13**, 1, 161 (2012).

Beltagy, I., K. Lo and A. Cohan, "SciBERT: A pretrained language model for scientific text", 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)", pp. 3615–3620 (Association for Computational Linguistics, Hong Kong, China, 2019), URL `https://www.aclweb.org/anthology/D19-1371`.

Bethard, S., G. Savova, M. Palmer and J. Pustejovsky, "SemEval-2017 task 12: Clinical TempEval", in "Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)", pp. 565–572 (Association for Computational Linguistics, Vancouver, Canada, 2017), URL `https://www.aclweb.org/anthology/S17-2093`.

Bodenreider, O., "The unified medical language system (umls): integrating biomedical terminology", Nucleic acids research **32**, suppl_1, D267–D270 (2004).

Borthwick, A., J. Sterling, E. Agichtein and R. Grishman, "Exploiting diverse knowledge sources via maximum entropy in named entity recognition", in "Sixth Workshop on Very Large Corpora", (1998).

Chowdhuri, S., S. McCrea, D. D. Fushman and C. O. Taylor, "Extracting biomedical terms from postpartum depression online health communities", AMIA Summits on Translational Science Proceedings **2019**, 592 (2019).

Ciaramita, M. and Y. Altun, "Named-entity recognition in novel domains with external lexical knowledge", in "Proceedings of the NIPS Workshop on Advances in Structured Learning for Text and Speech Processing", vol. 2005 (2005).

Cohen, K. B. and L. Hunter, "Natural language processing and systems biology", in "Artificial intelligence methods and tools for systems biology", pp. 147–173 (Springer, 2004).

Crichton, G., S. Pyysalo, B. Chiu and A. Korhonen, “A neural network multi-task learning approach to biomedical named entity recognition”, BMC bioinformatics **18**, 1, 368 (2017).

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, “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)”, pp. 4171–4186 (Association for Computational Linguistics, Minneapolis, Minnesota, 2019), URL `https://www.aclweb.org/anthology/N19-1423`.

Doğan, R. I., R. Leaman and Z. Lu, “Ncbi disease corpus: a resource for disease name recognition and concept normalization”, Journal of biomedical informatics **47**, 1–10 (2014).

Gardner, M., J. Berant, H. Hajishirzi, A. Talmor and S. Min, “Question answering is a format; when is it useful?”, arXiv preprint arXiv:1909.11291 (2019).

Gerner, M., G. Nenadic and C. M. Bergman, “Linnaeus: a species name identification system for biomedical literature”, BMC bioinformatics **11**, 1, 85 (2010).

Giorgi, J. M. and G. D. Bader, “Towards reliable named entity recognition in the biomedical domain”, Bioinformatics **36**, 1, 280–286 (2020).

He, L., M. Lewis and L. Zettlemoyer, “Question-answer driven semantic role labeling: Using natural language to annotate natural language”, in “Proceedings of the 2015 conference on empirical methods in natural language processing”, pp. 643–653 (2015).

Katiyar, A. and C. Cardie, “Nested named entity recognition revisited”, in “Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)”, pp. 861–871 (2018).

Kazama, J. and K. Torisawa, “Exploiting wikipedia as external knowledge for named entity recognition”, in “Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)”, pp. 698–707 (2007).

Kim, J.-D., T. Ohta, S. Pyysalo, Y. Kano and J. Tsujii, “Overview of bionlp’09 shared task on event extraction”, in “Proceedings of the BioNLP 2009 workshop companion volume for shared task”, pp. 1–9 (2009).

Kim, J.-D., T. Ohta, Y. Tateisi and J. Tsujii, “Genia corpus—a semantically annotated corpus for bio-textmining”, Bioinformatics **19**, suppl_1, i180–i182 (2003).

Kim, J.-D., T. Ohta, Y. Tsuruoka, Y. Tateisi and N. Collier, “Introduction to the bio-entity recognition task at jnlpba”, in “Proceedings of the international joint workshop on natural language processing in biomedicine and its applications”, pp. 70–75 (Citeseer, 2004).

Krallinger, M., F. Leitner, O. Rabal, M. Vazquez, J. Oyarzabal and A. Valencia, "Chemdner: The drugs and chemical names extraction challenge", Journal of cheminformatics **7**, S1, S1 (2015).

Lai, G., Q. Xie, H. Liu, Y. Yang and E. Hovy, "Race: Large-scale reading comprehension dataset from examinations", in "Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing", pp. 785–794 (Association for Computational Linguistics, 2017), URL `http://aclweb.org/anthology/D17-1082`.

Lee, J., W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining", Bioinformatics **36**, 4, 1234–1240 (2020).

Li, X., J. Feng, Y. Meng, Q. Han, F. Wu and J. Li, "A unified MRC framework for named entity recognition", in "Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics", pp. 5849–5859 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.acl-main.519`.

Lin, H., Y. Lu, X. Han and L. Sun, "Sequence-to-nuggets: Nested entity mention detection via anchor-region networks", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 5182–5192 (2019).

Liu, H., Z.-Z. Hu, M. Torii, C. Wu and C. Friedman, "Quantitative assessment of dictionary-based protein named entity tagging", Journal of the American Medical Informatics Association **13**, 5, 497–507 (2006).

Liu, T., J.-G. Yao and C.-Y. Lin, "Towards improving neural named entity recognition with gazetteers", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 5301–5307 (Association for Computational Linguistics, Florence, Italy, 2019), URL `https://www.aclweb.org/anthology/P19-1524`.

Luan, Y., D. Wadden, L. He, A. Shah, M. Ostendorf and H. Hajishirzi, "A general framework for information extraction using dynamic span graphs", 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)", pp. 3036–3046 (2019).

McCann, B., N. S. Keskar, C. Xiong and R. Socher, "The natural language decathlon: Multitask learning as question answering", arXiv preprint arXiv:1806.08730 (2018).

Mihaylov, T., P. Clark, T. Khot and A. Sabharwal, "Can a suit of armor conduct electricity? a new dataset for open book question answering", in "EMNLP", (2018).

Nédellec, C., R. Bossy, J.-D. Kim, J.-J. Kim, T. Ohta, S. Pyysalo and P. Zweigenbaum, "Overview of bionlp shared task 2013", in "Proceedings of the BioNLP shared task 2013 workshop", pp. 1–7 (2013).

Nguyen, T. O.-R. B. N., J. T. J.-D. Kim and S. Pyysalo, “Overview of bionlp shared task 2011”, in “Proceedings of BioNLP Shared Task 2011 Workshop”, pp. 1–6 (2011).

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library”, in “Advances in Neural Information Processing Systems 32”, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, pp. 8024–8035 (Curran Associates, Inc., 2019), URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Pyysalo, S. and S. Ananiadou, “Anatomical entity mention recognition at literature scale”, Bioinformatics **30**, 6, 868–875 (2014).

Pyysalo, S., T. Ohta, M. Miwa and J. Tsujii, “Towards exhaustive protein modification event extraction”, in “Proceedings of BioNLP 2011 Workshop”, pp. 114–123 (2011).

Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, “Squad: 100,000+ questions for machine comprehension of text”, in “Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing”, pp. 2383–2392 (2016).

Sap, M., H. Rashkin, D. Chen, R. LeBras and Y. Choi, “Social iqa: Commonsense reasoning about social interactions”, in “EMNLP”, (2019).

Savery, M. E., W. J. Rogers, M. Pillai, J. G. Mork and D. Demner-Fushman, “Chemical entity recognition for medline indexing”, AMIA Summits on Translational Science Proceedings **2020**, 561 (2020).

Shibuya, T. and E. Hovy, “Nested named entity recognition via second-best sequence learning and decoding”, Transactions of the Association for Computational Linguistics **8**, 605–620 (2020).

Si, Y., J. Wang, H. Xu and K. Roberts, “Enhancing clinical concept extraction with contextual embeddings”, Journal of the American Medical Informatics Association **26**, 11, 1297–1304 (2019).

Smith, L., L. K. Tanabe, R. J. nee Ando, C.-J. Kuo, I.-F. Chung, C.-N. Hsu, Y.-S. Lin, R. Klinger, C. M. Friedrich, K. Ganchev *et al.*, “Overview of biocreative ii gene mention recognition”, Genome biology **9**, S2, S2 (2008).

Song, H.-J., B.-C. Jo, C.-Y. Park, J.-D. Kim and Y.-S. Kim, “Comparison of named entity recognition methodologies in biomedical documents”, Biomedical engineering online **17**, 2, 158 (2018).

Straková, J., M. Straka and J. Hajic, “Neural architectures for nested ner through linearization”, in “Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics”, pp. 5326–5331 (2019).

Sun, W., A. Rumshisky and O. Uzuner, "Evaluating temporal relations in clinical text: 2012 i2b2 challenge", Journal of the American Medical Informatics Association **20**, 5, 806–813 (2013).

Talmor, A., J. Herzig, N. Lourie and J. Berant, "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)", pp. 4149–4158 (Association for Computational Linguistics, Minneapolis, Minnesota, 2019), URL `https://www.aclweb.org/anthology/N19-1421`.

Uzuner, O., A. Bodnari, S. Shen, T. Forbush, J. Pestian and B. R. South, "Evaluating the state of the art in coreference resolution for electronic medical records", Journal of the American Medical Informatics Association **19**, 5, 786–791 (2012).

Uzuner, Ö., I. Solti and E. Cadag, "Extracting medication information from clinical text", Journal of the American Medical Informatics Association **17**, 5, 514–518 (2010).

Uzuner, Ö., B. R. South, S. Shen and S. L. DuVall, "2010 i2b2/va challenge on concepts, assertions, and relations in clinical text", Journal of the American Medical Informatics Association **18**, 5, 552–556 (2011).

Wang, X., J. Lyu, L. Dong and K. Xu, "Multitask learning for biomedical named entity recognition with cross-sharing structure", BMC bioinformatics **20**, 1, 427 (2019a).

Wang, X., Y. Zhang, Q. Li, X. Ren, J. Shang and J. Han, "Distantly supervised biomedical named entity recognition with dictionary expansion", in "2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)", pp. 496–503 (IEEE, 2019b).

Wang, X., Y. Zhang, X. Ren, Y. Zhang, M. Zitnik, J. Shang, C. Langlotz and J. Han, "Cross-type biomedical named entity recognition with deep multi-task learning", Bioinformatics **35**, 10, 1745–1752 (2018).

Wei, C.-H., Y. Peng, R. Leaman, A. P. Davis, C. J. Mattingly, J. Li, T. C. Wiegers and Z. Lu, "Overview of the biocreative v chemical disease relation (cdr) task", in "Proceedings of the fifth BioCreative challenge evaluation workshop", vol. 14 (2015).

Wilson, E. B., "Probable inference, the law of succession, and statistical inference", Journal of the American Statistical Association **22**, 158, 209–212 (1927).

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing", ArXiv **abs/1910.03771** (2019).

Wu, S., K. Roberts, S. Datta, J. Du, Z. Ji, Y. Si, S. Soni, Q. Wang, Q. Wei, Y. Xiang *et al.*, "Deep learning in clinical natural language processing: a methodical review", Journal of the American Medical Informatics Association **27**, 3, 457–470 (2020).

Yadav, V. and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models", in "Proceedings of the 27th International Conference on Computational Linguistics", pp. 2145–2158 (Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018), URL https://www.aclweb.org/anthology/C18-1182.

Yadav, V., R. Sharp and S. Bethard, "Deep affix features improve neural named entity recognizers", in "Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics", pp. 167–172 (2018).

Yoon, W., C. H. So, J. Lee and J. Kang, "Collabonet: collaboration of deep neural networks for biomedical named entity recognition", BMC bioinformatics **20**, 10, 249 (2019).

Yu, J., B. Bohnet and M. Poesio, "Named entity recognition as dependency parsing", in "Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics", pp. 6470–6476 (Association for Computational Linguistics, Online, 2020), URL https://www.aclweb.org/anthology/2020.acl-main.577.

Dataset	Entity	Entity Mentions	Train +	Train -	Dev +	Dev -	Test +	Test -
AnatEM	Anatomy	13701	3514	2169	1122	959	2308	1405
BC2GM	Gene (Gene/Protein)	24516	6404	6071	1283	1214	2568	2424
BC4CHEMD	Chemical	84249	14488	16002	14554	15909	12415	13738
BC5CDR	Chemical	14913	2951	1595	3017	1551	3090	1688
	Disease	12852	2658	1888	2727	1841	2842	1936
BioNLP09	Protein (Gene/Protein)	14963	4711	2716	1014	433	1700	739
BioNLP11EPI	Protein (Gene/Protein)	15881	3797	1896	1241	714	2836	1282
BioNLP11ID	Protein (Gene/Protein)	6551	1255	1193	446	265	955	977
	Organism	3469	1120	1328	270	441	779	1153
	Chemical	973	334	2114	77	634	151	1781
	Regulon-Operon	87	9	2439	19	692	43	1889
BioNLP13CG	Gene or Gene Product (Gene/Protein)	7908	1956	1077	393	610	1185	721
	Cell	4061	1388	1645	399	604	714	1192
	Simple Chemical (Chemical)	2270	645	2388	274	729	431	1475
	Cancer	2582	908	2125	324	679	665	1241
	Organ	2517	919	2114	305	698	565	1341
	Organism	2093	827	2206	267	736	486	1420
	Tissue	587	259	2774	77	926	153	1753
	Amino Acid	135	38	2995	17	986	34	1872
	Cellular Component	569	247	2786	78	925	138	1768
	Organism Substance	283	131	2902	33	970	81	1825
	Pathological Formation	228	91	2952	35	968	73	1833
	Anatomical System	41	16	3017	3	1000	17	1889
	Immaterial Anatomical	102	47	2986	18	985	29	1877
	Organism Subdivision	98	42	2991	12	991	35	1871
	Multi-Tissue Structure	857	345	2688	114	889	236	1670
	Developing Anatomical Structure	35	13	3020	5	998	17	1889
BioNLP13GE	Protein (Gene/Protein)	12031	1499	901	1655	1010	1936	1376
BioNLP13PC	Gene or Gene Product (Gene/Protein)	10891	2153	346	723	134	1396	298
	Complex	1502	542	1957	178	679	398	1296
	Simple Chemical (Chemical)	2487	596	1903	244	613	450	1244
	Cellular/ Component	1013	373	2126	144	713	263	1431
CRAFT	GGP (Gene/Protein)	16108	4458	5539	1358	2105	3140	3634
	Taxon (Taxonomy)	6835	2511	7486	994	2469	1710	5064
	CHEBI (Chemical)	6018	1908	8089	586	2877	1344	5430
	CL (Cell Line)	5487	2058	7939	540	2923	1257	5517
	SO (Sequence Ontology)	18856	4303	5694	1711	1752	3023	3751
	GO (Gene Ontology)	4166	1499	8498	336	3127	1344	5430
EXPTM	Protein (Gene/Protein)	4698	857	520	279	158	1160	679

Table 2.3: Data Distribution, with Counts of Entities, Number of Positive Samples with at Least One Entity Mentions, and Negative Samples with No Target Entity.

Dataset	Entity	Entity Mentions	Train +	Train -	Dev +	Dev -	Test +	Test -
JNLPBA	DNA	10550	4670	12146	553	1218	624	3226
	RNA	1061	713	16103	89	1682	102	3748
	Cell Line	4315	2591	14225	285	1486	378	3472
	Cell Type	8584	4735	12081	415	1356	1403	2447
	Protein (Gene/Protein)	35234	11840	4976	1137	634	2368	1482
Linnaeus	Species	4242	1546	9173	520	3300	1029	5381
NCBI-Disease	Disease	6871	2921	2473	489	434	538	398
2010-i2b2	Problem	18979	4213	4226	-	-	5802	6590
	Treatment	13809	3126	4226	-	-	7234	6590
	Test	13576	2426	4226	-	-	4591	6590
2011-i2b2	Person	17744	7207	3990	-	-	4715	2971
	Problem	18869	7003	3990	-	-	4384	2971
	Treatment	17708	5300	3990	-	-	3565	2971
	Test	13514	4191	3990	-	-	2786	2971
2012-i2b2	Problem	4754	2832	3597	-	-	2326	2683
	Treatment	7076	2341	4088	-	-	1976	3033
	Test	4754	1786	4643	-	-	1465	3544
	Occurrence	5126	2086	4343	-	-	1677	3332
	Clinical-Department Event	1716	852	5577	-	-	655	4354
	Evidential Event	1334	706	5723	-	-	560	4449
	Date	1201	1493	4936	-	-	1076	3933
	Frequency	249	207	6222	-	-	159	4850
	Duration	405	381	6048	-	-	310	4699
	Time	69	68	6361	-	-	53	4956
	Discharge	176	176	0	-	-	109	4699
	Admission	177	177	0	-	-	116	4956

Table 2.4: Data Distribution, with Counts of Entities, Number of Positive Samples with at Least One Entity Mentions, and Negative Samples with No Target Entity.

		ANATEM			BC2GM			BC4CHEMD			BC5CDR			BIONLP09			BIONLP11EPI		
BASE	BioBERT	89.63	89.32	89.47	82.45	83.83	83.13	90.97	89.59	90.27	87.19	90.59	88.84	89.65	88.60	89.13	85.23	85.60	85.41
	MimicBERT	86.23	85.72	85.98	79.04	80.32	79.68	88.77	85.41	87.06	84.01	86.86	85.39	87.71	84.15	85.89	79.37	77.78	78.57
	BERT-MRC	72.24	75.09	73.64	73.80	74.59	74.19	86.47	85.52	85.99	71.22	73.68	72.43	74.62	70.69	72.60	77.81	67.01	72.01
	TradBioBERT	4.10	1.37	2.05	54.28	47.74	50.80	79.12	77.34	78.22	42.81	58.93	49.60	7.22	3.23	4.46	54.77	46.07	50.04
	SOTA	-	-	**91.61***	-	-	81.69*	-	-	92.36	-	-	90.01	-	-	84.20*	-	-	78.86*
KGQA	BioBERT	90.29	89.43	89.85	82.47	83.36	82.91	91.93	91.11	91.52	89.63	88.80	89.21	91.35	92.21	91.78	88.26	86.77	87.51
	MimicBERT	87.05	86.50	86.80	81.22	81.40	81.31	89.47	88.86	89.16	88.25	86.78	87.51	89.19	91.41	90.29	88.01	82.19	85.00
	BERT-CNN	91.15	90.74	90.94	83.12	83.82	**83.47**†	92.77	92.02	**92.39**	90.96	90.04	**90.50**	91.62	92.26	**91.94**†	89.71	87.64	**88.66**†

		BIONLP11ID			BIONLP13CG			BIONLP13GE			BIONLP13PC			CRAFT			EXPTM		
BASE	BioBERT	84.23	85.77	84.70	84.82	86.42	85.56	72.92	85.42	78.68	87.64	90.56	89.06	84.92	86.56	85.70	76.12	79.81	77.92
	MimicBERT	83.93	81.84	82.35	77.52	80.32	78.71	64.72	65.53	65.12	81.59	85.45	83.43	81.27	79.08	80.04	66.74	67.29	67.01
	BERT-MRC	80.25	73.26	76.60	74.57	69.25	71.81	77.79	75.54	76.65	76.51	76.95	76.73	75.79	72.25	73.98	76.61	76.93	76.77
	TradBioBERT	4.67	2.29	3.07	0.00	0.00	0.00	53.92	31.52	39.78	0.00	0.00	0.00	0.01	0.01	0.01	45.52	47.05	46.27
	SOTA	-	-	81.73*	-	-	78.90*	-	-	78.58*	-	-	81.92*	-	-	79.56*	-	-	74.90*
KGQA	BioBERT	86.34	85.58	85.96	87.18	87.28	87.23	82.28	86.58	84.38	90.14	92.09	91.11	88.18	88.61	88.39	85.97	85.30	85.64
	MimicBERT	83.12	81.78	82.45	85.08	85.37	85.23	81.61	86.28	83.88	87.62	89.63	88.61	85.01	87.14	86.06	84.09	81.34	82.69
	BERT-CNN	87.91	86.81	**87.36**†	91.19	89.15	**90.16**†	83.33	88.44	**85.81**†	91.20	92.11	**91.65**†	90.71	89.71	**90.21**†	86.57	87.59	**87.08**†

		JNLPBA			LINNAEUS			NCBIDISEASE			2010-i2b2			2011-i2b2			2012-i2b2		
BASE	BioBERT	69.96	78.19	73.63	92.30	86.42	89.27	86.67	89.38	88.00	85.32	83.23	84.26	91.24	90.32	90.78	79.31	75.89	77.56
	MimicBERT	67.99	76.32	71.66	91.69	81.81	86.46	84.04	88.23	86.08	90.37	88.29	89.32	92.83	91.22	92.02	79.78	81.01	80.39
	BERT-MRC	70.52	69.38	69.95	74.13	73.56	73.84	77.25	73.23	75.19	75.32	73.23	74.26	81.24	80.32	80.78	69.31	65.89	67.56
	TradBioBERT	2.63	1.51	1.92	12.3	15.95	13.89	75.05	75.44	75.25	58.24	46.98	52.01	64.9	34.39	44.95	6.73	22.13	10.32
	SOTA	-	-	78.58*	-	-	**95.68***	-	-	89.36	-	-	90.25#	-	-	-	-	-	80.91#
KGQA	BioBERT	76.12	82.15	79.02	90.32	89.88	90.10	87.50	90.67	89.05	91.19	92.63	91.82	92.32	91.88	92.12	73.53	83.21	78.07
	MimicBERT	74.97	80.79	77.77	86.31	85.10	85.70	86.82	88.80	87.80	91.28	92.96	92.11	92.19	92.30	92.24	81.57	84.76	83.13
	BERT-CNN	76.95	81.57	**79.19**	93.51	91.76	92.63	88.29	91.42	**89.82**	91.90	93.44	**92.67**†	93.45	93.89	**93.68**	84.84	82.41	**83.98**†

Table 2.5: Precision, Recall, and F-measure (in Order) for 18 Datasets Compared with Multiple Models. * Tagged Scores Are Non-BERT Systems, # BERT-large, and Rest Are BERT-base Systems. Our Models Use Knowledge Type All or Question, Whichever Is Observed Best on Validation Accuracy. TradBioBERT Is the BioBERT Model Trained Traditionally (B-e_k, I-e_k, O Tags for Each Entity) on Combined Training Data. Biobert and BERT-CNN Models Use 110 Million Parameters, and MimicBERT Uses 355 Million Parameters. Best F1 Scores Are in Bold. Underlined Are Our Best Scores, Where Our Models Are Not State-of-the-art. † Tagged Scores Are Statistically Significantly Better than State-of-the-art (Model Is Better Including 95% Confidence Intervals Based on Wilson Score Intervals (Wilson, 1927) over Strong Baselines). JNLPBA, NCBIDisease, and 2012-i2b2 BERT-CNN Results Are with "Question" Knowledge, and the Rest Are with "All" Knowledge. Dataset Statistics Are in 2.4. Confidence Interval Bounds Are in 2.12.

MODEL →	**BIOBERT(KGQA)**		**MimicBERT(KGQA)**		**BERT-CNN(KGQA)**	
ENTITY ↓	**SRC F1**	**TGT F1**	**SRC F1**	**TGT F1**	**SRC F1**	**TGT F1**
Gene/Protein	84.83	83.27	82.46	80.75	84.99	**85.63**
Chemical	91.35	**76.13**	85.60	66.63	89.83	75.92
Disease	86.46	68.01	84.13	60.54	88.74	**70.00**
Problem	94.42	90.29	93.72	89.67	94.43	**90.90**
Treatment	94.01	89.67	93.76	89.99	94.16	**90.22**
Test	94.99	**91.36**	94.85	90.91	94.85	91.11

Table 2.8: Transfer Learning Experiment Results (Trained and Tested with Question Context). The Metric Is the Exact Match F1 for the Source (SRC) and Target (TGT) Domain. Bold Across Each of the Entities Are the Best, and Underlined Are the Second Best.

Method	**Precision**	**Recall**	**F-Measure**
Hyper-Graph (Katiyar and Cardie, 2018)	70.60	70.40	71.50
ARN (Lin *et al.*, 2019)	75.80	73.90	74.80
Path-BERT (Shibuya and Hovy, 2020)	78.07	76.45	77.25
DYGIE (Luan *et al.*, 2019)	-	-	76.20
Seq2Seq-BERT (Straková *et al.*, 2019)	-	-	78.31
KGQA+ENTITY TYPE	77.66	74.34	75.96
KGQA+QUESTION	77.48	74.08	75.74
KGQA+DEFINITION	73.86	64.64	68.94
KGQA+EXAMPLE	79.62	77.56	78.57
KGQA+ALL	**81.91**	**78.80**	**80.32**

Table 2.9: Performance of BERT-CNN with Five Knowledge Contexts for GENIA (Kim *et al.*, 2003) Dataset. The Best Results Are in Bold.

TEXT	NESTED ENTITY PAIRS
1. Tissue-specific regulation of the rabbit 15 - lipoxygenase gene in erythroid cells by a transcriptional silencer .	15 - lipoxygenase **(Protein)**, rabbit 15 - lipoxygenase gene **(DNA)**
2. Here, we demonstrate that the human GM - CSF receptor alpha promoter directs reporter gene activity in a tissue-specific fashion in myelomonocytic cells, which correlates with its expression pattern as analyzed by reverse transcription PCR.	GM - CSF **(Protein)**, human GM - CSF receptor alpha promoter **(DNA)**
3. In unstimulated cells which do not secrete IL - 2, only Sp1 binds to this region, while in stimulated IL - 2 secreting cells the inducible EGR - 1 protein recognizes this element.	IL - 2 **(Protein)**, stimulated IL - 2 secreting cells **(Cell Line)**

Table 2.10: Nested Entity Predictions Using the All Knowledge Contexts

Text	**Gold**	**Predictions**
1. The position of the magnetic bead is measured using an inverted microscope placed beneath the flow cell .	**Anatomy**: cell	flow cell
2. Expression of the HI - viral structure proteins is driven by a natural 5 ' LTR promoter and a 3 ' polyadenylation signal, whereas the envelope expression out of the shuttle vector is achieved by an IRES .	**Anatomy**: LTR, envelope	5 ' LTR, envelope
3. SAP is upregulated in AD and protects amyloid fibrils from proteolysis in vitro [140 , 141] .	**Anatomy**: amyloid fibril	fibril
4. Serotonin is present in mammalian iris - ciliary body complex (ICB) at higher concentration that in non - mammalian species [5 , 45 , 73 , 129 , 137] .	**Anatomy**: ICB, iris - ciliary body complex	ICB, iris, ciliary body
5. He had a gastric pull up in 1992 of the stomach into the left thorax and he has a transverse colostomy and a sinus tract on the abdomen .	**Problems**: a gastric pull up, a transverse colostomy	a gastric pull up, a transverse colostomy, a sinus
6. His gastrointestinal bleeding issues were investigated with an upper endoscopy which revealed multiple superficial gastric ulcerations consistent with an non-steroidal anti-inflammatory drugs gastropathy .	**Problems**: his gastrointestinal bleeding issues, gastric ulcerations, an non-steroidal anti-inflammatory drugs gastropathy; **Test**: an upper endoscopy	All Predictions correct.

7. The functional effects of these interactions are that CBP and p300 , but not P/CAF , enhance EKLF\'s transcriptional activation of the beta-globin promoter in erythroid cells .	**DNA**: beta-globin, promoter;
	Cell-Type: erythroid cells;
	Proteins: P/CAF, CBP, EKLF, p300;
		All Predictions correct.

8. Expression cloning with this region of DNA now shows that tegument protein VP22 and the viral dUTPase, encoded by genes UL49 and UL50 , respectively , are T-cell antigens .	**Proteins**: tegument protein VP22, T-cell antigens, viral dUTPase	VP22, tegument protein, T-cell antigens, viral
9. The Phlebotominae sand flies Lutzomyia intermedia Lutz & Neiva 1912 and Lutzomyia whitmani Antunes & Coutinho 1912 are vectors of cutaneous leishmaniasis in Brazil .	**Species**: Lutzomyia whitmani, Lutzomyia intermedia	Lutzomyia intermedia Lutz & Neiva, Lutzomyia whitmani

Table 2.11: Examples of Errors and Correct Predictions Made by Our KGQA BERT-CNN with All Knowledge Context. Entity Types Are Mentioned in the Gold Targets.

DATASET	**P**	**ΔP**	**R**	**ΔR**	**F**	**ΔF**
ANATEM	88.88	**-0.07**	88.23	+0.89	88.55	+0.41
BC2GM	82.93	**-0.04**	82.84	+0.55	82.88	+0.26
BC4CHEMD	91.64	+0.43	91.03	**-0.02**	91.33	+0.21
BC5CDR	89.61	+0.48	88.37	+0.79	88.98	+0.64
BIONLP09	90.76	**-0.60**	92.33	**-0.81**	91.54	**-0.71**
BIONLP11EPI	87.86	+0.72	86.29	+1.11	87.07	+0.92
BIONLP11ID	85.39	+1.21	85.19	+0.16	85.29	+0.68
BIONLP13CG	88.61	**-0.63**	87.54	**-0.27**	88.07	**-0.45**
BIONLP13GE	81.94	**-0.12**	88.77	**-2.51**	85.22	**-1.24**
BIONLP13PC	89.48	**-0.45**	90.73	+1.14	90.10	+0.33
CRAFT	87.43	+0.64	88.12	+0.07	87.78	+0.35
EXPTM	85.12	**-1.40**	85.99	**-0.25**	85.55	**-0.84**
JNLPBA	75.85	+0.19	82.37	**-0.74**	78.98	**-0.25**
LINNAEUS	88.16	+0.31	87.91	+0.56	88.03	+0.44
NCBI-DISEASE	88.00	**-1.34**	90.46	+0.42	89.22	**-0.50**
2010-i2b2	94.96	+0.31	95.71	+0.20	95.33	+0.26
2011-i2b2	94.01	+0.41	94.09	+0.28	94.05	+0.35
2012-i2b2	82.97	**-1.64**	86.65	**-2.13**	84.77	**-1.88**

Table 2.13: Precision (P), Recall (R) and F-measure (F) Using BERT-CNN Model Trained on **Balanced Dataset** with **Question** as Knowledge. Δp, Δr, Δf Represent a Change in Performance Compared to Training Our Model on Full Datasets. A Negative Value Indicates Training on the Unbalanced Dataset Is Better, While a Positive Value Indicates Balanced Dataset Training Produces Better Performance. Negative Values Are in Bold.

Chapter 3

CONSTRUCTING FLOW GRAPHS FROM PROCEDURAL CYBERSECURITY TEXTS

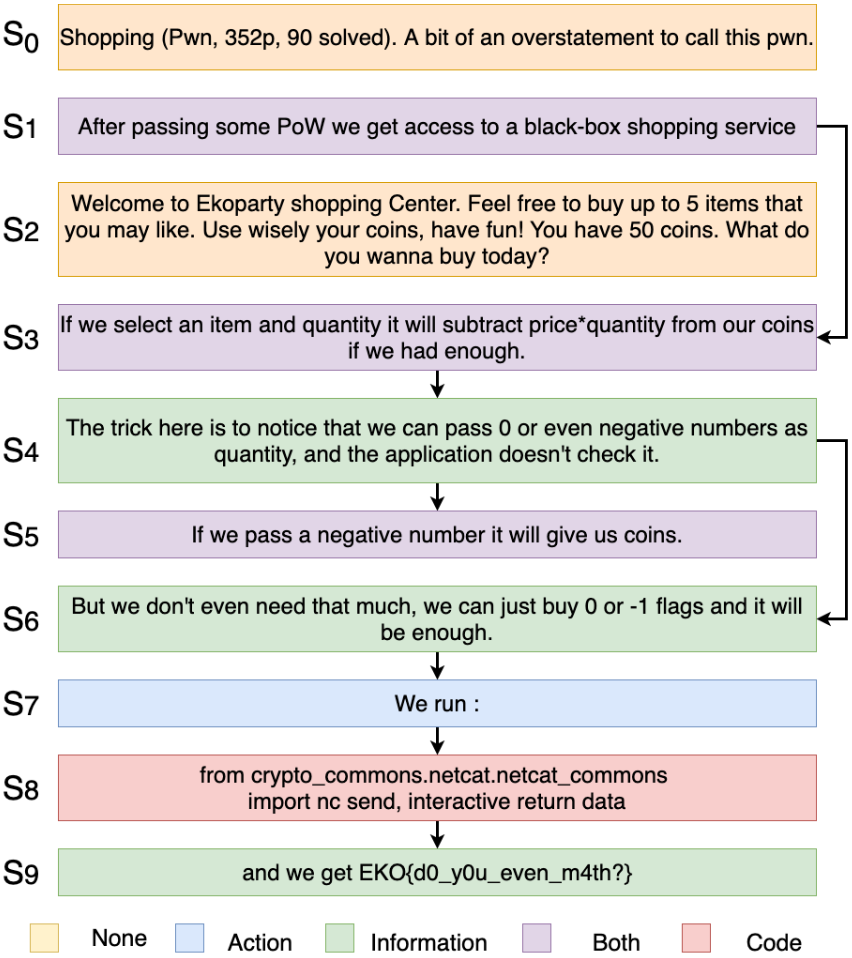
ABSTRACT

Following procedural texts written in natural languages is challenging. We must read the whole text to identify the relevant information or identify the instruction flows to complete a task, which is prone to failure. If such texts are structured, we can readily visualize instruction flows, reason or infer a particular step, or even build automated systems to help novice agents achieve a goal. However, this structure recovery task is a challenge because of such texts' diverse nature. This paper proposes to identify relevant information from such texts and generate information flows between sentences. We built a large annotated procedural text dataset (CTFW) in the cybersecurity domain (3154 documents). This dataset contains valuable instructions regarding software vulnerability analysis experiences. We performed extensive experiments on CTFW with our LM-GNN model variants in multiple settings. To show the generalizability of both this task and our method, we also experimented with procedural texts from two other domains (Maintenance Manual and Cooking), which are substantially different from cybersecurity. Our experiments show that Graph Convolution Network with BERT sentence embeddings outperforms BERT in all three domains.

3.1 Introduction

Many texts in the real world contain valuable instructions. These instructions define individual steps of a process and help users achieve a goal (and corresponding sub-goals). Documents including such instructions are called *procedural texts*, ranging from simple cooking recipes to complex instruction manuals. Additionally, *discussion in a shared forum or social media platform, teaching books, medical notes, sets of advice about social behavior, directions for use, do-it-yourself notices, itinerary guides can all be considered as procedural texts* (Delpech and Saint-Dizier, 2008). Most of these texts are in the form of natural languages and thus, lack structures. We define *structure* as sentence-level dependencies that lead to a goal. These dependencies can vary based on the text domain. Some examples of such dependencies are action traces, effects of an action, information leading to the action, and instruction order. Constructing structured flow graphs out of procedural texts is the foundation for natural language understanding and summarization, question-answering (QA) beyond factoid QA, automated workflow visualization, and the recovery of causal relationships between two statements. By flow-graph we mean both information and action flows in a text. However, the lack of structures in such texts makes them challenging to follow, visualize, extract inferences, or track the states of an object or a sub-task, which ultimately makes constructing their flow graphs an insurmountable task.

Procedural texts are common in cybersecurity, where security analysts document how to discover, exploit, and mitigate security vulnerabilities in articles, blog posts, and technical reports, which are usually referred to as *security write-ups.* Practitioners in cybersecurity often use write-ups as educational and research materials. Constructing structured flow graphs from security write-ups may help with automated vulnerability discovery and mitigation, exploit generation, and security education in



S0 Shopping (Pwn, 352p, 90 solved). A bit of an overstatement to call this pwn.

S1 After passing some PoW we get access to a black-box shopping service

S2 Welcome to Ekoparty shopping Center. Feel free to buy up to 5 items that you may like. Use wisely your coins, have fun! You have 50 coins. What do you wanna buy today?

S3 If we select an item and quantity it will subtract price*quantity from our coins if we had enough.

S4 The trick here is to notice that we can pass 0 or even negative numbers as quantity, and the application doesn't check it.

S5 If we pass a negative number it will give us coins.

S6 But we don't even need that much, we can just buy 0 or -1 flags and it will be enough.

S7 We run :

S8
```
from crypto_commons.netcat.netcat_commons
import nc send, interactive return data
```

S9 and we get EKO{d0_y0u_even_m4th?}

Figure 3.1: An Example Flow Graph from the CTFW. Sentences in S_2 Are Merged into One Block for Clarity.

general. However, automatically analyzing and extracting information from security write-ups are extremely difficult since they lack structure.

Figure 3.1 illustrates the core of a security write-up (broken into sentences) that carries instructions for exploiting a vulnerability in an online shopping service. S_1, S_3, and S_4 are the author's observations about the service's nature. Based on this information, S_5 and S_6 are two possible paths of action. The author chose S_6 and ran the Python code in S_8 to exploit the service. S_0 and S_2 are irrelevant to the author's goal of exploiting this service.

Here we propose a novel approach to extract action paths out of structure-less, natural language texts by identifying actions and information flows embedded in and between sentences and constructing action flow graphs. Specifically, our focus is on procedural texts in the cybersecurity domain. We also show that constructing flow graphs helps extract paths of actions in domains besides cybersecurity, such as cooking and maintenance manuals.

Most previous works (Mori *et al.*, 2014; Kiddon *et al.*, 2015; Malmaud *et al.*, 2014; Maeta *et al.*, 2015; Xu *et al.*, 2020; Mysore *et al.*, 2019; Song *et al.*, 2011) focus on fine-grained knowledge extraction from procedural texts in diverse domains. There are also a handful of works (Delpech and Saint-Dizier, 2008; Fontan and Saint-Dizier, 2008; Jermsurawong and Habash, 2015) that study the structure of natural language texts. Different from previous works, we extract structures and construct flow graphs from natural texts at the sentence level. This is because fine-grained domain-entity extraction tasks require a large amount of annotated data from people with specific in-depth domain knowledge, whereas text structures can be generalized.

Dataset. We built a dataset from security write-ups that are generated from past Capture The Flag competitions (CTFs). CTFs are computer security competitions that are usually open to everyone in the world. Players are expected to find and exploit security vulnerabilities in a given set of software services, and through exploiting vulnerabilities, obtain a *flag*—a unique string indicating a successful attempt—for each exploited service. Once the game is over, many players publish security write-ups that detail how they exploited services during the game. While these write-ups are a valuable educational resource for students and security professionals, they are usually unstructured and lacking in clarity. We collected 3617 CTF write-ups from the Internet, created a procedural text dataset, and invited domain experts to label each sentence for the purpose of constructing flow graphs and identifying action paths. To the best of our knowledge, this is the first attempt to use the knowledge embedded in security write-ups for automated analysis. The data and the code is publicly available [1] for future research.

This paper makes the following contributions:

- We built a new procedural text dataset, CTFW, in the cybersecurity domain. To the best of our knowledge, CTFW is the first dataset that contains valuable information regarding vulnerability analysis from CTF write-ups.
- We proposed a new NLU task of generating flow graphs from natural language procedural texts at the sentence level without identifying fine-grained named entities.
- We proposed four variations of a graph neural network-based model (LM-GNN) to learn a neighbor-aware representation of each sentence in a procedural text and predict the presence of edges between any pair of sentences.
- We evaluated our models on CTFW. To the best of our knowledge, this is the first attempt in automated extraction of information from security write-ups. We also evaluated our models across three datasets in different domains and showed the generalizability of our approach.

3.2 Our Approach

We map each sentence of a procedural text as a node in a graph, and the action or information flows as edges. The task is then simplified into an edge prediction task: Given a pair of nodes, find if there is an edge between them. We learn feature representations of nodes using language models like BERT/RoBERTa (Devlin *et al.*,

[1] https://github.com/kuntalkumarpal/FlowGraph

Dataset Statistics	**COR**	**MAM**	**CTFW**
# Documents	297	575	3154
Avg size of document	9.52	8.12	17.11
Avg length of sentence	65.46	34.81	92.87
# Edges ($\|e^{+}\|$)	2670	5043	54539
$\|e^{+}\| : (\|e^{+}\| + \|e^{-}\|)$	0.18	0.12	0.07
Avg degree of node	1.83	1.76	1.88

Table 3.1: Dataset Statistics. $|e^{+}|$ Is the Total Number of Actual Edges, and $|e^{+}| + |e^{-}|$ Is the Total Number of Edges Possible. The In-degree of the Starting Node and Out-degree of the End Node Are Both .

2019; Liu *et al.*, 2019). Then, to make the nodes aware of their neighboring sentences, we use Graph Neural Network (GNN) to update the node representations. We check for the edge between every pair of nodes in a graph and reduce the task to a binary classification during inference. This formulation enables us to predict any kind of structure from a document.

3.3 Dataset Creation

In this section, we present how we created three datasets on which we evaluated our approach. Table 3.1 shows the statistics for each dataset used.

3.3.1 CTF Write-ups Dataset (CTFW)

Each CTF competition has multiple challenges or tasks. Each task may have multiple write-ups by different authors. We crawled 3617 such write-ups from GitHub and CTFTime (CTFTime, 2021). Write-ups are unique and diverse but have common inherent principles. For each write-up, we provide two kinds of annotations: *sentence type* and *flow structure*. The writing style is informal with embedded code snippets and often contains irrelevant information.

Part of the annotations was provided as an optional, extra-credit assignment for the Information Assurance course. These CTF write-ups were directly related to the course content, where students were required to read existing CTF write-ups and write write-ups for other security challenges they worked on during the course. Then students were given the option of voluntarily annotating CTF write-ups they read for extra credits in the course. For this task, we followed all the existing annotation guidelines and practices. We also ensured that (1) The volunteers were aware of the fact that their annotations would be used for a research project (2) They were aware that no PII was involved or would be used in the research project (3) They were aware that extra credits were entirely optional, and they could refrain from submitting at any point of time without any consequences (4) Each volunteer was assigned only 10-15 write-ups based on a pilot study we did ahead of time, annotating an average-length CTF write-up took about two minutes (maximum ten mins).

The remaining annotations were performed by the Teaching Assistants (TA) of the course. These annotations were done as part of the course preparation process, which was part of their work contract. All the TAs were paid bi-weekly compensation by the university or by research funding. It was also ensured that the TAs knew these annotations would be used for a research project, their PII was not involved and annotations were to be anonymized before use. We verified the annotations by randomly selecting write-ups from the set. Figure 3.1 shows a sample annotation.

Sentence Type Annotations. We split the documents into sentences using natural language rules. We then ask the volunteers to annotate the type of each statement as either Action (A), Information (I), Both (A/I), Codes (C), or irrelevant (None).

Action sentences are those where the author specifies actions taken by them, whereas, *Information* statements mention observations of the author, the reasons and effects of their action. Sentences containing *codes* are assigned as C, and those which can be considered as both information and actions are marked as *Both* (A/I).

Flow structure Annotations. The second level of annotations is regarding the write-up structure. Each volunteer is given a CSV file for each document with a set of sentence IDs and text for each write-up. They are asked to annotate the flow of information in the document by annotating the sentence id of some next possible sentences, which indicate the flow. We filter those write-ups which are irrelevant and those which did not have much detail (single line of valuable information). We call a write-up irrelevant if it has no action-information annotations or if it has direct codes without any natural language description of steps to detect vulnerabilities. We only keep write-ups written in the English language for this work. Finally, we have 3154 write-ups with *sentence type* and *structure annotations.*

CTFTime website states that the write-ups are copyrighted by the authors who posted them and it is practically impossible to contact each author. Such data is also allowed to use for academic research purposes(Copyright Office, 2016; European Union, 2020). Thus, we follow the previous work using data from CTFTime (Švábenskỳ *et al.*, 2021), and share only the URLs of those write-ups which we use. We do not provide the scraper script since it would create a local copy of the write-up files unauthorized by the users. Interested readers can replicate the simple scraper script from the instructions provided and use it after reviewing the conditions under which it is permissible to use it. We, however, share our annotations for those write-up files.

3.3.2 Cooking Recipe Flow Corpus (COR)

This corpus (Yamakata *et al.*, 2020) provides 300 recipes with annotated recipe Named Entities and fine-grained interactions between each entity and their sequencing steps. Since we attempt to generate action flow graphs without explicitly identifying each named entity, we aggregate the fine-grained interactions between recipe Named Entities to generate sentence-level flows for each recipe. We reject three single-sentence recipes.

3.3.3 Maintenance Manuals Dataset (MAM)

This dataset (Qian *et al.*, 2020) provides multi-grained process model extraction corpora for the task of extracting process models from texts. It has over 160 Maintenance Manuals. Each manual has fine-grained interactions between each entity and its sequencing steps. We use the annotations from sentence-level classification data and semantic recognition data for generating annotations of sentence-level flows for each process. Here also, we reject single sentence processes.

3.3.4 Extraction and Processing of Write-ups:

The extraction of CTF Write-up involved the following three phases.

Writeup URL extraction : We loop through all the write-up pages on CTFTime website from pages numbered 1 to 25500). We use a simple Python scraper to scrape the content of each page using the Python requests (Reitz, 2020) library. We look for the keyword "*Original write-ups*" and extracted the href component if present. These URLs are stored for each writeup indexed with the page numbers.

Write-up Content extraction : We use these URLs and extract the contents of the write-ups using Python libraries requests and BeautifulSoup (Richardson, 2007).

We extract all the text lines ignoring contents in HTML tags like style, scripts, head, and title. The contents are stored in a text file named with the same page ids as the URLs.

Processing of Write-up : We processed and filter out sentences that do not have any verb forms using spacy (spaCy, 2017) POS-Tagger. We cleaned and removed unnecessary spaces and split them into sentences. The processing script is available in the GitHub.

3.3.5 CTFW Data Statistics

In CTFW, there are write-ups for 2236 unique tasks. Only four out of those have more than 5 write-ups each. 72% of the tasks have a single write-up. The write-ups are from 311 unique competitions, ranging from years 2012-2019. A task having multiple write-ups vary in content. In CTFW, only 3% of the tasks have more than three write-ups, and 9% have more than two.

3.4 CTFW STC Label Statistics

Table 3.2 shows the label distributions of Sentence Type Classification data.

Label	**Train**	**Val**	**Test**
A	11143	1499	3321
I	23279	3075	6882
A/I	2931	380	826
C	1386	185	338
NONE	82012	12192	22896

Table 3.2: CTFW Sentence Type Classification

3.5 Model Description

Our goal is to find paths or traces of actions or information between texts. This needs an understanding of each sentence's interconnection. Hence, we modeled the problem into an edge prediction task in a graph using GNNs. We represent each sentence as a node and directed edges as information flows. Since this is procedural text (unidirectional nature) of instructions, we consider only the directed edges from one sentence S_n to any of its next sentences S_{n+i}. The node representations are learned using language models (LM) and GNNs.

3.5.1 Document to Sentence Pre-processing

Given a natural language document, first, we split the document into sentences based on simple rules and heuristics. COR and MAM datasets already have documents split into separate sentences. In the flow graph creation task, we filter out irrelevant sentences for the CTFW dataset based on the sentence type annotations. After this pre-processing task, each document (D_i) is converted into a series of sentences (S_j) where n is the number of valid sentences in a document.

$$D_i = \{S_0, S_1, S_2 ... S_{n-1}\}$$

3.5.2 Document to Graph Representation

A graph $(G = (V, E))$ is formally represented as a set of nodes $(V = \{v_0, v_1, ..\})$ connected by edges $(E = \{e_0, e_1, ..\}$ where $e_i = \{v_m, v_n\})$. We consider the sentences (S_j) of any document (D_i) as nodes of a directed graph (G_i). We experiment with two graph structure types for learning better node representation using GNN. First, we form local windows (W_N, where $N = 3, 4, 5, all$ sentences) for each sentence and

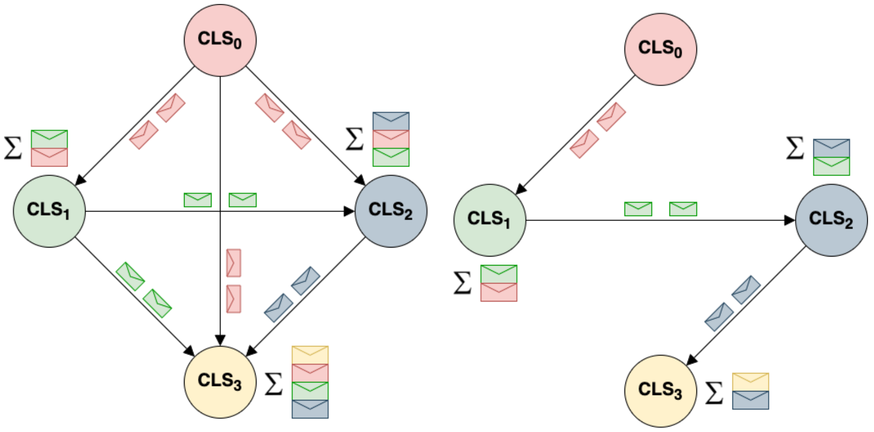


Figure 3.2: *Node Representation Learning for a Document with Four Sentences in Single-layer GNN.* Left: *Semi-Complete* Structure, Right: *Linear* Structure. During Training, the Sentence Representation (CLS_i) Is Enriched Using Appropriate Message-passing Techniques from the Connected 1-hop Neighbors.

allow the model to learn from all of the previous sentences in that window. We form the document graph by connecting each sentence with every other sentence in that window, with directed edges only from S_i to S_j where $i < j$. We do this since procedural languages are directional. We call this configuration *Semi-Complete*. Second, we consider connecting the nodes linearly where every S_i is connected to S_{i+1} except the last node. We call this *Linear* setting. Figure 3.2 shows the settings. We use LMs like BERT and RoBERTa to generate initial sentence representations. For each sentence (S_i), we extract the pooled sentence representation (CLS_{S_i}) of contextual BERT/RoBERTa embeddings (h_{S_i}). We use CLS_{S_i} as node features for the graph (G_i).

$$h_{S_i} = BERT\left([CLS]s_0s_1...s_{n-1}[SEP]\right)$$

3.5.3 Neighbor Aware Node Feature Learning

Since the LM sentence vectors are generated individually for each sentence in the document, they are not aware of other local sentences. So, through the *semi-complete* graph connection, the model can learn a global understanding of the document. However, the *linear* connection helps it learn better node representation conditioned selectively on its predecessor. We call the connected nodes the neighbor nodes. We use Graph Convolutional Network (GCN) (Kipf and Welling, 2016) and Graph Attention Network (GAT) (Veličković *et al.*, 2017) to aggregate the neighbor information for each node following the generic graph learning function (3.1)

$$\mathbf{H}^{l+1} = f(\mathbf{H}^{l}, \mathbf{A}) \tag{3.1}$$

where $\mathbf{A}$ is the adjacency matrix of the graph, $\mathbf{H}^l$ and $\mathbf{H}^{(l+1)}$ are the node representations at lth and $(l+1)$th layer of the network and f is the message aggregation function. In GCN, each node i, aggregates the representations of all of its neighbors $N(i)$ based on $\mathbf{A}$ and itself at layer l and computes the enriched representation $\mathbf{h}_i^{l+1}$ based on the weight matrix $\mathbf{\Theta}$ of the layer normalized by degrees of source $d(i)$ and its connected node $d(j)$ as per (3.2). In GAT, messages are aggregated based on multi-headed attention weights (α) learned from the neighbor node representations $\mathbf{h}_j^l$ following (3.3).

$$\mathbf{h}_i^{l+1} = \mathbf{\Theta} \sum_{j \in N(i) \cup \{i\}} \frac{1}{\sqrt{d(i)d(j)}} \mathbf{h}_j^l \tag{3.2}$$

$$\mathbf{h}_i^{l+1} = \alpha_{ii} \mathbf{\Theta} \mathbf{h}_i^l \quad + \sum_{j \in N(i)} \alpha_{ij} \mathbf{\Theta} \mathbf{h}_j^l \tag{3.3}$$

3.5.4 Projection

We concatenate the neighbor-aware node representations of each pair of nodes $(\mathbf{h}_i;\mathbf{h}_j)$ from a graph and pass it through two projection layers with a GELU (Hendrycks and Gimpel, 2016) non-linearity in between. We use the same non-linearity functions used by the BERT layers for consistency. We steadily decrease the parameters of each projection layer by half. During testing, given a document, we are unaware of which two sentences are connected. So, we compare each pair of nodes. This leads to an unbalanced number of existing (1) and non-existing (0) edge labels. Hence, we use the weighted cross-entropy loss function as in equation (3.4) and (3.5), where L is the weighted cross-entropy loss, w_c is the weight for class c, i is the data in each mini-batch.

$$L(x,c) = w_c\Big(- x_c + log\Big(\sum_j exp(x_j) \Big) \Big) \tag{3.4}$$

$$L = \frac{\sum_{i=1}^{N} L(i, c_i)}{\sum_{i=1}^{N} w_{c_i}} \tag{3.5}$$

3.5.5 Training and Inference

Our training data comprises a set of sentences and the connections as an adjacency matrix for each document. Batching is done based on the number of graphs. GCN/GAT updates the sentence representations. A pair of node representations are assigned a label of 1 if there is an edge between them; otherwise, we assign them 0. Thus, we model it as a binary classification task as in equation (3.6) where f is the projection function, g is the softmax function, and y is the binary class output. Depending on the weighted cross-entropy loss, the node representations get updated after each epoch. During inference, the model generates node representations of each sentence in a test document, and we predict whether an edge exists between any two nodes in a given document graph.

$$y_c = \arg\max_k g(f(\mathbf{h}_i; \mathbf{h}_j), k) \quad c \in \{0, 1\} \tag{3.6}$$

3.6 Experiments

	Models	CTFW		COR		MAM	
		PRAUC	F1	PRAUC	F1	PRAUC	F1
Baselines	Random	-	50.49	-	42.78	-	47.82
	Weighted Random	-	37.81	-	39.13	-	44.10
	BERT-NS	0.5751	26.12	0.5638	43.14	0.5873	29.73
	RoBERTa-NS	0.5968	32.44	0.5244	42.99	0.6236	39.65
Ours	BERT-GCN	0.7075	69.26	**0.6312**	58.13	**0.6888**	63.75
	RoBERTa-GCN	**0.7221**	69.04	0.6233	61.44	0.6802	65.73
	BERT-GAT	0.5585	61.93	0.4553	41.93	0.4568	62.18
	RoBERTa-GAT	0.5692	64.51	0.4358	24.74	0.4585	59.55

Table 3.3: Comparison with Baselines on Best Test Area under Precision-recall Curve (PRAUC) and Its Corresponding F1 for **CTFW** (CTFwrite-up), **COR** (Cooking), **MAM** (Maintenance) Datasets. NS Is the next Sentence-based Prediction Approach. Our Best Model Performance Is Bold, While the Maximum Baseline Performance Is Underlined.

3.6.1 Datasets and Tasks

Each dataset is split into the train, validation, and test sets in a 70:10:20 ratio. The first task is identifying relevant information from raw CTF write-ups by classifying the type of each sentence. The second task is identifying information flows between sentences by predicting edges between sentence pairs, if any.

3.6.2 Metrics

We use *accuracy* as the evaluation metric for the Sentence Type classification task on CTFW. For the second task, because of the label imbalance, we compare based on the *area under Precision-Recall curve* (PRAUC) and also report the corresponding *F1-score*. Hence do not report the area under the ROC curve or accuracy.

We consider four settings for this task. The *no window* setting (W_{all}) checks whether there is an edge between any two statements in the given document. The comparisons required in this setting are directly proportional to the document's size. In CTFW, the size of each write-up is quite large. So, to reduce unnecessary comparisons, we apply simple heuristics that instructions in procedural text, in general, do not have longer *direct* dependencies. Thus, using the windows, we can control each sentence's number of comparisons (node). To understand how the performances change we evaluate with a *sliding windows of N sentences* (W_N) where $N = 3, 4, 5$. The comparisons are only made with the next N sentences from a given sentence. For example, in case of W_5, for first sentence (S_1) we check for edges with S_2, S_3, S_4, S_5, S_6 and not S_7 on-wards. However, to have a fair comparison, we keep labeled out-of-window gold edges, if any. The ratios of existing and total edges in CTFW are 0.07 (W_{all}), 0.24 (W_5), 0.29 (W_4), and 0.38 (W_3).

3.6.3 Training Details

We use Pytorch Geometric (Fey and Lenssen, 2019) for GNN and transformers (Wolf *et al.*, 2020) for LM implementations. Training is done with AdamW (Loshchilov and Hutter, 2017) optimizer along with linear warmup scheduler on 4 Tesla V100 16GB GPUs. We use BERT-base-uncased, BERT-large-uncased, RoBERTa-base, and RoBERTa-large versions as the base model. We store the model with the best PRAUC score. Batch size of {4,8,16} and learning rates of {1e-5,5e-6} are used. Maximum sequence length varies between {64, 80, 128}. GNN depths are kept at 128 (layer 1) and 64 (layer 2). We use a dropout of 0.4 in selected layers. For GAT, we keep four attention heads in layer 1.

The correct set of hyperparameters is found by running three trials. We run for {50, 100} epochs and store the model with the best PRAUC score. Each training with evaluation takes around 1-3 hours for the base version of models and around 6 hours for larger versions depending upon the dataset used. The model parameters are directly proportional to the model parameters of language models since the GNN only allows a few more parameters as compared to the LMs.

3.7 Results And Discussion

3.7.1 Sentence Type Classification (STC)

We use large and base versions of BERT and RoBERTa for this task to predict the type of sentences in a given text to establish a baseline for this task. This task helps to identify relevant and irrelevant sentences in a document. Each sentence is classified into any of Action, Information, Both, Code, and None. These fine-grained annotations can be used in later works for creating automated agents for vulnerability analysis. The processed data consists of 120751 samples for training, 17331 for validation, and 34263 for testing. Table 3.4 shows that RoBERTa-large performs better than the rest.

Model	**Val**	**Test**
BERT-Base	78.48±0.25	77.42±0.10
BERT-Large	78.19±0.48	77.13±0.20
RoBERTa-Base	78.85±0.25	77.37±0.11
RoBERTa-Large	79.02±0.16	**77.66±0.12**

Table 3.4: Sentence Type Classification (Mean Accuracy from Three Seed Values). Best Performance in Bold.

3.7.2 Flow Structure Prediction

Here we present the performance results for the flow structure prediction.

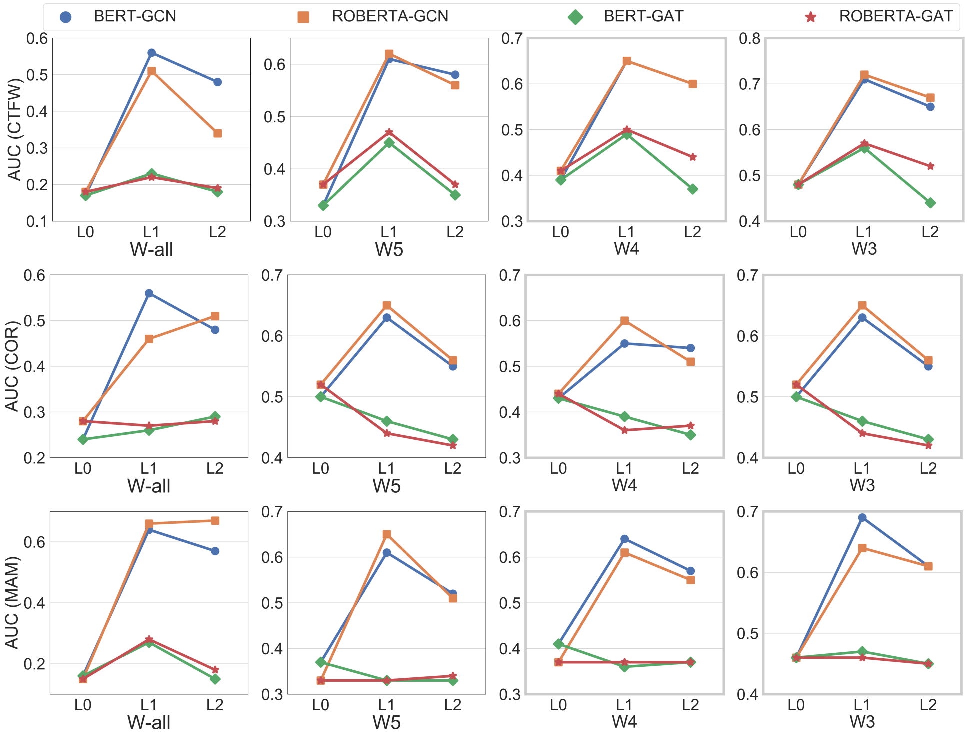


Figure 3.3: Effect of GNN Layers (L_0, L_1, L_2) on Performance (PRAUC) of the Models for W_{all}, W_5, W_4, W_3 Settings on the Three Datasets

Random Baseline:

In the *Random* baseline, for every pair of nodes in each document we randomly select 0 (no-edge) or 1 (edge). For *Weighted Random* baseline, we choose randomly, based on the percentage of edge present in the train set. We only report F1 since there is no probability calculation.

Next Sentence-based Prediction (NS) Baseline:

We use LMs like BERT and RoBERTa in a next-sentence prediction setting to get the baselines. Each pair of sentences is concatenated using [SEP] token and passed through these language models. Using the pooled LM representation, we classify whether an edge exists between them or not. We show the maximum PRAUC and its corresponding F1 for each dataset from the results of each of our window settings (W_3, W_4, W_5, W_{all}).

Next Sentence-based Prediction With Weighted Cross-Entropy

Table 3.5 shows the PRAUC values when we use weighted cross-entropy with the base version of BERT on unbalanced data during training. The results are not much different than the Next-Sentence baseline shown previously.

Dataset	W_3	W_4	W_5	W_{all}
CTFW	0.4613	0.4397	0.2546	0.3681
COR	0.4724	0.4748	0.4837	0.4761
MAM	0.5318	0.2318	0.2297	0.4724

Table 3.5: BERT-base-uncased Performance with NS Prediction When Weighted Cross-entropy Used with Unbalanced Training Data

Our Models

We compare four variants of our LM-GNN models both with baseline and among each other in Table 3.3. The scores are overall best scores across single and double layers GNN (GCN/GAT) and LM (BERT/RoBERTa) after experiments with both base and large versions, trained with pre-trained and randomly initialized weights.

We see that the best LM-GCN models outperform the best baseline model by 0.12, 0.07, and 0.06 in PRAUC for CTFW, COR, and MAM datasets, respectively. However, the best LM-GAT scores fall short of the baselines indicating that the graph attentions on LM *sentence representations* cannot learn robust representation to perform this edge prediction task. Another thing to notice here is that the best BERT-GCN models

	W_3	W_4	W_5	W_{all}
CTFW-SC	0.6630	0.5985	0.5733	**0.5590**
CTFW-L	**0.7221**	**0.6520**	**0.6150**	0.3962
CTFW-EP	0.3700	0.2900	0.2400	0.0700
COR-SC	0.5639	0.5129	0.4731	**0.5580**
COR-L	**0.6456**	**0.6012**	**0.5274**	0.4034
COR-EP	0.3700	0.3100	0.2600	0.1700
MAM-SC	0.6528	0.6219	0.6091	**0.6718**
MAM-L	**0.6888**	**0.6362**	**0.6137**	0.4161
MAM-EP	0.4500	0.3700	0.3200	0.1500

Table 3.6: Effect of Semi-Complete(SC) and Linear(L) Graph Connection on 3 Datasets in Area under Precision-Recall Curve (PRAUC). We Also Keep Edge Percent (EP) in Four Window Settings for Comparison.

perform better than RoBERTa-GCN for COR and MAM datasets while performing poorly in the CTFW dataset. We hypothesize that this is because, the CTFW dataset has ten times more data than COR and six times more than MAM, which helps the RoBERTa model correctly predict the edges.

3.7.3 Analysis

Effect of Graph Connection Type:

Table 3.6 shows how the models behave with semi-complete (SC) and linear (L) graph connections. For each dataset, we compare the PRAUC results for each window to draw more granular insight into the effect of neighbor-aware representation learning. When we restrict graph learning by creating small windows (W_3, W_4, W_5), the linear model works better because of its selective learning conditioned on its predecessor. On the other hand, the semi-complete connection helps to learn a global awareness and works best in the W_{all} setting. It is important to note that each model performs better than the average PRAUC performance, which is the percentage of edges in the data indicating that the model is able to learn using the graph connections.

Effect of Graph Layers:

We study how the depth of the GNNs affects the performance. We compare PRAUC across all four variations of the model in No-Window (W_{all}), W_5, W_4, W_3 settings in Figure 3.3. We experimented with no (L_0), single (L_1), and double (L_2) GNN layers. In all three datasets, we find the performance improves when we use a single layer and degrades beyond that for each of the windows with GCN-based models. We do not go beyond two layers because of this observation and the graph connection types we use. We believe the reason for this drop (0.03-0.08 PRAUC) is that information from 2-hop neighbors might hinder the learning of the current node and confuse the model to predict wrongly. The GAT-based models mostly remain unaffected with the graph layers for both COR and MAM while showing some improvement in CTFW for one layer setting.

Effect of Pre-trained LM Weights:

We study the impact of pre-trained weights of BERT and RoBERTa on the performance in Figure 3.4. We notice, for the three datasets, the performance slightly decreases when the pre-trained model weights are used. This observation may be because the texts' nature is quite different from the type of texts these LMs have been pre-trained on. The CTFW data often contains code fragments embedded in sentences, emoticons, or common conversational languages used in public forums.

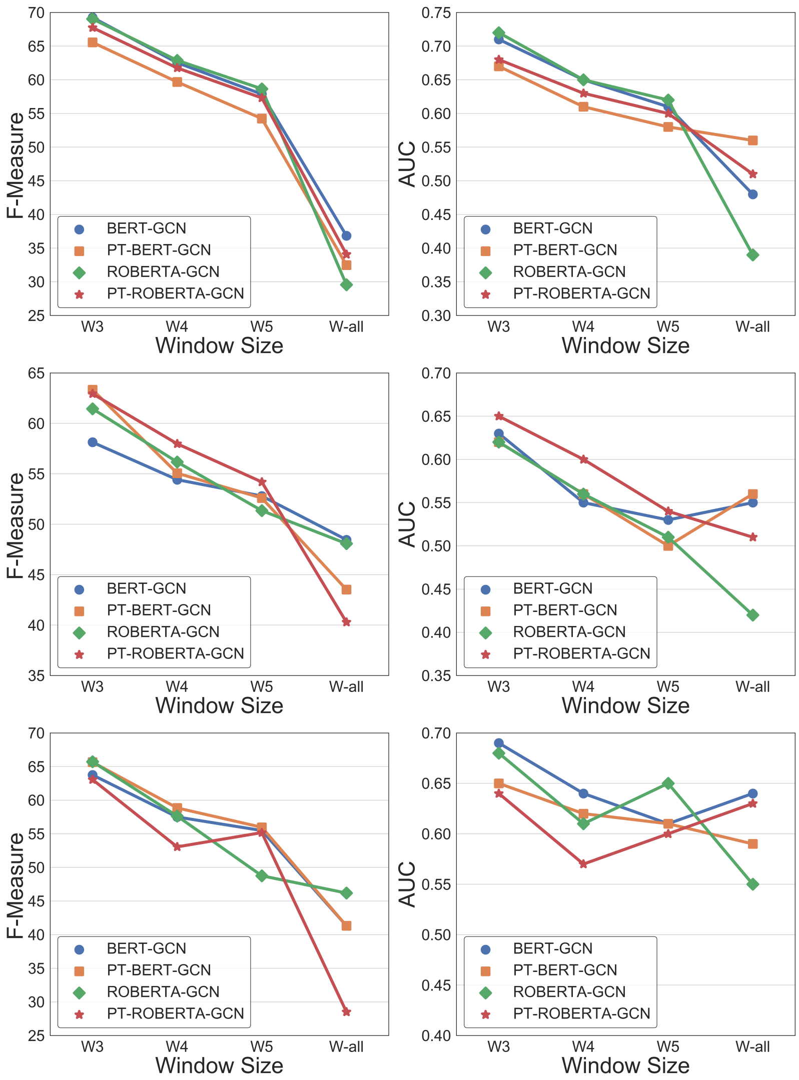


Figure 3.4: Performance for CTFW, COR, MAM Trained from scratch and Fine-tuned With Pre-trained Weights.

Effect of LM Size:

We also experimented with the size of sentence embeddings to see if that makes any difference to the performance. We use base and large versions of BERT and RoBERTa for the experiments across three datasets. We present the impact on F1 and PRAUC in Figure 3.5. The performance of the larger versions of the models drops in all three datasets. This drop, we believe, is because the sentences in these texts are relatively short and help the smaller versions of the models with lesser parameters to learn better.

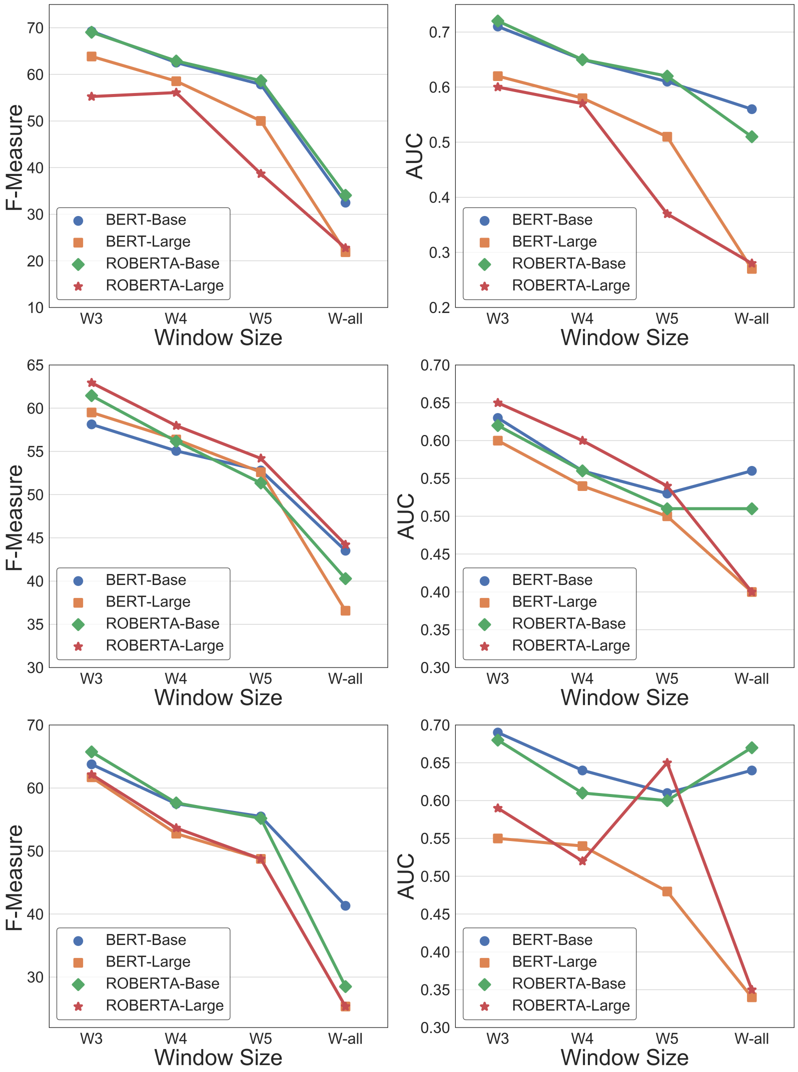


Figure 3.5: Performance on CTFW, COR, MAM Trained with base and large Version of the Model.

Number of Comparisons Reduction Using Windows

We can control the total number of comparisons required to predict the edges in a graph by using the windows (W_N where $N = 3, 4, 5, all$). The number of comparisons for each window is given by the equation 3.7. We can reduce the number of comparisons considerably for large documents using shorter windows of 3, 4, and 5 sentences. The number of comparison $\mathcal{C}$ is defined by

$$\mathcal{C} = \begin{cases} max\{(n-s), 0\}s + \frac{s(s-1)}{2} & n = 3, 4, 5 \\ \binom{n}{2} & n = all \end{cases} \tag{3.7}$$

Other experiments: We also experimented with modifications of other parts of the models like changing the number of projection layers, projection layer sizes, the number of attention heads in the GAT model, or dropout percent in selected layers and modes of message aggregation (add, max, mean). We do not report them since they do not significantly change PRAUC values.

3.8 Related Work

Procedural knowledge extraction: There are attempts to extract structured knowledge from cooking instructions in the form of named entities (Malmaud *et al.*, 2014), their sentence-level dependencies (Mori *et al.*, 2014; Maeta *et al.*, 2015; Xu *et al.*, 2020), and action-verb argument flow across sentences (Jermsurawong and Habash, 2015; Kiddon *et al.*, 2015; Pan *et al.*, 2020). In other domains, extraction of clinical steps from MEDLINE abstracts (Song *et al.*, 2011), extraction of material synthesis operations and its arguments in material science (Mysore *et al.*, 2019), providing structures to how-to procedures (Park and Motahari Nezhad, 2018), and action-argument retrieval from web design tutorials (Yang *et al.*, 2019) mostly focus on fine-grained entity extractions rather than action or information traces. The goal of our paper is to construct flow graphs from free-form, natural-language procedural texts without diverse domain knowledge. Hence, we refrain from training specialized named-entity recognizers for each domain to find specific entities. Our work is related to event or process discovery in process modeling tasks (Epure *et al.*, 2015; Honkisz *et al.*, 2018; Qian *et al.*, 2020; Hanga *et al.*, 2020), but our goal is not finding specific events or actions from procedural texts. In addition, recent research proposed a method to create the forum structures from an unstructured forum based on the contents of each post using BERT's Next Sentence Prediction (Kashihara *et al.*, 2020). However, we focus on building flow graphs for procedural texts using GNNs.

Graph Neural Networks: GNNs are important in reasoning with graph-structured data in three major tasks, node classification (Kipf and Welling, 2016; Hamilton *et al.*, 2017), link prediction (Schlichtkrull *et al.*, 2018), and graph classification (Ying *et al.*, 2018; Pan *et al.*, 2015, 2016; Zhang *et al.*, 2018). GNNs help learn better node representations in each task using neural message passing (Gilmer *et al.*, 2017) among connected neighbors. We consider two widely used GNNs, GCN (Graph Convolutional Network) (Kipf and Welling, 2016) and GAT (Graph Attention Networks) (Veličković *et al.*, 2017) to learn sentence representation to provide a better edge prediction.

Edge Prediction Task: Edge or link prediction tasks (Li *et al.*, 2018; Zhang and Chen, 2018; Pandey *et al.*, 2019; Haonan *et al.*, 2019; Bacciu *et al.*, 2019) work mainly on pre-existing networks or social graphs as inputs and predict the existence of future edges between nodes by extracting graph-specific features. Different from existing work, we modeled the task of generating a graph structure from a given natural-language text as an edge prediction task in a graph and learning representations of sentences considered as nodes.

Combinations of BERT and GCN: Recent works have used concatenation of BERT and GCN representations of texts or entities to improve the performance of tasks like commonsense knowledge-base completion (Malaviya *et al.*, 2019), text classification (Ye *et al.*, 2020; Lu *et al.*, 2020), multi-hop reasoning (Xiao *et al.*, 2019), citation recommendation (Jeong *et al.*, 2019), medication recommendation (Shang *et al.*, 2019), relation extraction (Zhao *et al.*, 2019). Graph-BERT (Zhang *et al.*, 2020) solely depends on the attention layers of BERT without using any message aggregation techniques. However, we differ from each of the previous methods in terms of model architecture, where we use BERT to learn initial sentence representations and GCN or GAT to improve them by learning representations from its neighboring connected sentences. BERT-GAT for MRC (Zheng *et al.*, 2020) created the graph structure from the well-structured Wikipedia data whereas we explore two predefined natures of graph structures because of the free-formed text nature without such well-defined text-sections, the presence of code-fragments, emoticons, and unrelated-token.

3.9 Conclusion and Future Work

We introduce a new procedural sentence flow extraction task from natural-language texts. This task is important for procedural texts in every domain. We create a sufficiently large procedural text dataset in the cybersecurity domain (CTFW) and construct structures from the natural form. We empirically show that this task can be generalized across multiple domains with different natures and styles of texts. In this paper, we only focus on English security write-ups. As part of future work, we plan to build automated agents in the cybersecurity domain to help and guide novices in performing software vulnerability analysis. We also plan to include non-English write-ups. We hope the CTFW dataset will facilitate other works in this research area. More details are available in the published work (Pal *et al.*, 2021).

Acknowledgement

The authors acknowledge support from the Defense Advanced Research Projects Agency (DARPA) grant number FA875019C0003 for this project.

Impact Statement

The dataset introduced here consists of write-ups written in public forums by students or security professionals from their personal experiences in the CTF challenges. The aggregated knowledge of such experiences is immense. This in-depth knowledge of the analysis tools and the approach to a problem is ideal for students working in software vulnerability analysis to learn from. Automated tutors built using such knowledge can reduce the efforts and time wasted in manually reading through a series of lengthy write-up documents.

CTFTime website states that the write-ups are copyrighted by the authors who posted them and it was practically impossible to contact each author. It is also allowed to use the data for research purposes (Copyright Office, 2016; European Union, 2020) Thus, we follow the previous work (Švábenskỳ *et al.*, 2021) using data from CTFTime and share only the URLs of those write-ups from the CTFTime website which we use. We do not provide the scraper script since it would create a local copy of the write-up files unauthorized by the users. Interested readers can replicate the simple scraper script from the instructions provided and use it after reviewing the conditions under which it is permissible to use it. We, however, share our annotations for those write-up files.

Part of the annotations was provided as an optional, extra-credit assignment for the Information Assurance course. These CTF write-ups were directly related to the course content, where students were required to read existing CTF write-ups and write write-ups for other security challenges they worked on during the course. Then students were given the option of voluntarily annotating CTF write-ups they read for extra credits in the course. For this task, we followed all the existing annotation guidelines and practices. We also ensured that

- The volunteers were aware of the fact that their annotations would be used for a research project.
- They were aware that no PII was involved or would be used in the research project.
- They were aware that extra credits were entirely optional, and they could refrain from submitting at any point in time without any consequences.
- Each volunteer was assigned only 10-15 write-ups based on a pilot study we did ahead of time, annotating an average-length CTF write-up took about two minutes (maximum ten mins).

The remaining annotations were performed by the Teaching Assistants (TA) of the course. These annotations were done as part of the course preparation process, which was part of their work contract. All the TAs were paid bi-weekly compensation by the university or by research funding. It was also ensured that the TAs knew these annotations would be used for a research project, their PII was not involved and annotations were to be anonymized before use.

REFERENCES

Bacciu, D., A. Micheli and M. Podda, "Graph generation by sequential edge prediction.", in "ESANN", (2019).

Copyright Office, U. S., "Copyright law of the united states, accessed: 2021-05-01", https://www.copyright.gov/title17/92chap1.html#107 (2016).

CTFTime, "CTFTime, accessed: 2021-05-01", https://ctftime.org (2021).

Delpech, E. and P. Saint-Dizier, "Investigating the structure of procedural texts for answering how-to questions", in "Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)", (European Language Resources Association (ELRA), Marrakech, Morocco, 2008), URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/20_paper.pdf.

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "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)", pp. 4171–4186 (Association for Computational Linguistics, Minneapolis, Minnesota, 2019), URL https://www.aclweb.org/anthology/N19-1423.

Epure, E. V., P. Martín-Rodilla, C. Hug, R. Deneckère and C. Salinesi, "Automatic process model discovery from textual methodologies", in "2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)", pp. 19–30 (IEEE, 2015).

European Union, ., "Copyright in the eu, accessed: 2021-05-01", https://europa.eu/youreurope/business/running-business/intellectual-property/copyright/index_en.htm (2020).

Fey, M. and J. E. Lenssen, "Fast graph representation learning with pytorch geometric", arXiv preprint arXiv:1903.02428 (2019).

Fontan, L. and P. Saint-Dizier, "Analyzing the explanation structure of procedural texts: Dealing with advice and warnings", in "Semantics in Text Processing. STEP 2008 Conference Proceedings", pp. 115–127 (2008).

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, "Neural message passing for quantum chemistry", arXiv preprint arXiv:1704.01212 (2017).

Hamilton, W., Z. Ying and J. Leskovec, "Inductive representation learning on large graphs", in "Advances in neural information processing systems", pp. 1024–1034 (2017).

Hanga, K. M., Y. Kovalchuk and M. M. Gaber, "A graph-based approach to interpreting recurrent neural networks in process mining", IEEE Access **8**, 172923–172938 (2020).

Haonan, L., S. H. Huang, T. Ye and G. Xiuyan, “Graph star net for generalized multi-task learning”, arXiv preprint arXiv:1906.12330 (2019).

Hendrycks, D. and K. Gimpel, “Gaussian error linear units (gelus)”, arXiv preprint arXiv:1606.08415 (2016).

Honkisz, K., K. Kluza and P. Wiśniewski, “A concept for generating business process models from natural language description”, in “International Conference on Knowledge Science, Engineering and Management”, pp. 91–103 (Springer, 2018).

Jeong, C., S. Jang, H. Shin, E. Park and S. Choi, “A context-aware citation recommendation model with bert and graph convolutional networks”, arXiv preprint arXiv:1903.06464 (2019).

Jermsurawong, J. and N. Habash, “Predicting the structure of cooking recipes”, in “Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing”, pp. 781–786 (2015).

Kashihara, K., J. Shakarian and C. Baral, “Social structure construction from the forums using interaction coherence”, in “Proceedings of the Future Technologies Conference”, pp. 830–843 (2020).

Kiddon, C., G. T. Ponnuraj, L. Zettlemoyer and Y. Choi, “Mise en place: Unsupervised interpretation of instructional recipes”, in “Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing”, pp. 982–992 (2015).

Kipf, T. N. and M. Welling, “Semi-supervised classification with graph convolutional networks”, arXiv preprint arXiv:1609.02907 (2016).

Li, J.-c., D.-l. Zhao, B.-F. Ge, K.-W. Yang and Y.-W. Chen, “A link prediction method for heterogeneous networks based on bp neural network”, Physica A: Statistical Mechanics and its Applications **495**, 1–17 (2018).

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach”, arXiv preprint arXiv:1907.11692 (2019).

Loshchilov, I. and F. Hutter, “Decoupled weight decay regularization”, arXiv preprint arXiv:1711.05101 (2017).

Lu, Z., P. Du and J.-Y. Nie, “Vgcn-bert: Augmenting bert with graph embedding for text classification”, in “European Conference on Information Retrieval”, pp. 369–382 (Springer, 2020).

Maeta, H., T. Sasada and S. Mori, “A framework for procedural text understanding”, in “Proceedings of the 14th International Conference on Parsing Technologies”, pp. 50–60 (2015).

Malaviya, C., C. Bhagavatula, A. Bosselut and Y. Choi, “Exploiting structural and semantic context for commonsense knowledge base completion”, arXiv preprint arXiv:1910.02915 (2019).

Malmaud, J., E. Wagner, N. Chang and K. Murphy, "Cooking with semantics", in "Proceedings of the ACL 2014 Workshop on Semantic Parsing", pp. 33–38 (2014).

Mori, S., H. Maeta, Y. Yamakata and T. Sasada, "Flow graph corpus from recipe texts.", in "LREC", pp. 2370–2377 (2014).

Mysore, S., Z. Jensen, E. Kim, K. Huang, H.-S. Chang, E. Strubell, J. Flanigan, A. McCallum and E. Olivetti, "The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures", arXiv preprint arXiv:1905.06939 (2019).

Pal, K. K., K. Kashihara, P. Banerjee, S. Mishra, R. Wang and C. Baral, "Constructing flow graphs from procedural cybersecurity texts", arXiv preprint arXiv:2105.14357 (2021).

Pan, L.-M., J. Chen, J. Wu, S. Liu, C.-W. Ngo, M.-Y. Kan, Y. Jiang and T.-S. Chua, "Multi-modal cooking workflow construction for food recipes", in "Proceedings of the 28th ACM International Conference on Multimedia", pp. 1132–1141 (2020).

Pan, S., J. Wu, X. Zhu, G. Long and C. Zhang, "Task sensitive feature exploration and learning for multitask graph classification", IEEE transactions on cybernetics **47**, 3, 744–758 (2016).

Pan, S., J. Wu, X. Zhu, C. Zhang and S. Y. Philip, "Joint structure feature exploration and regularization for multi-task graph classification", IEEE Transactions on Knowledge and Data Engineering **28**, 3, 715–728 (2015).

Pandey, B., P. K. Bhanodia, A. Khamparia and D. K. Pandey, "A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges", Expert Systems with Applications **124**, 164–181 (2019).

Park, H. and H. R. Motahari Nezhad, "Learning procedures from text: Codifying how-to procedures in deep neural networks", in "Companion Proceedings of the The Web Conference 2018", pp. 351–358 (2018).

Qian, C., L. Wen, A. Kumar, L. Lin, L. Lin, Z. Zong, J. Wang *et al.*, "An approach for process model extraction by multi-grained text classification", in "International Conference on Advanced Information Systems Engineering", pp. 268–282 (Springer, 2020).

Reitz, K., "Requests: Http for humans, accessed: 2020-10-23", `https://requests.readthedocs.io/en/master/` (2020).

Richardson, L., "Beautiful soup documentation", April (2007).

Schlichtkrull, M., T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov and M. Welling, "Modeling relational data with graph convolutional networks", in "European Semantic Web Conference", pp. 593–607 (Springer, 2018).

Shang, J., T. Ma, C. Xiao and J. Sun, "Pre-training of graph augmented transformers for medication recommendation", arXiv preprint arXiv:1906.00346 (2019).

Song, S.-k., H.-s. Oh, S. H. Myaeng, S.-P. Choi, H.-W. Chun, Y.-S. Choi and C.-H. Jeong, "Procedural knowledge extraction on medline abstracts", in "International Conference on Active Media Technology", pp. 345–354 (Springer, 2011).

spaCy, "spacy v2.0", `https://spacy.io/models/en#en_core_web_md` (2017).

Švábenskỳ, V., P. Čeleda, J. Vykopal and S. Brišáková, "Cybersecurity knowledge and skills taught in capture the flag challenges", Computers & Security **102**, 102154 (2021).

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph attention networks", arXiv preprint arXiv:1710.10903 (2017).

Wolf, T., J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer *et al.*, "Transformers: State-of-the-art natural language processing", in "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations", pp. 38–45 (2020).

Xiao, Y., Y. Qu, L. Qiu, H. Zhou, L. Li, W. Zhang and Y. Yu, "Dynamically fused graph network for multi-hop reasoning", arXiv preprint arXiv:1905.06933 (2019).

Xu, F. F., L. Ji, B. Shi, J. Du, G. Neubig, Y. Bisk and N. Duan, "A benchmark for structured procedural knowledge extraction from cooking videos", arXiv preprint arXiv:2005.00706 (2020).

Yamakata, Y., S. Mori and J. A. Carroll, "English recipe flow graph corpus", in "Proceedings of The 12th Language Resources and Evaluation Conference", pp. 5187–5194 (2020).

Yang, L., C. Fang, H. Jin, W. Chang and D. Estrin, "Creative procedural-knowledge extraction from web design tutorials", arXiv preprint arXiv:1904.08587 (2019).

Ye, Z., G. Jiang, Y. Liu, Z. Li and J. Yuan, "Document and word representations generated by graph convolutional network and bert for short text classification", in "ECAI 2020", pp. 2275–2281 (IOS Press, 2020).

Ying, Z., J. You, C. Morris, X. Ren, W. Hamilton and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling", in "Advances in neural information processing systems", pp. 4800–4810 (2018).

Zhang, J., H. Zhang, L. Sun and C. Xia, "Graph-bert: Only attention is needed for learning graph representations", arXiv preprint arXiv:2001.05140 (2020).

Zhang, M. and Y. Chen, "Link prediction based on graph neural networks", in "Advances in Neural Information Processing Systems", pp. 5165–5175 (2018).

Zhang, M., Z. Cui, M. Neumann and Y. Chen, "An end-to-end deep learning architecture for graph classification", in "Thirty-Second AAAI Conference on Artificial Intelligence", (2018).

Zhao, Y., H. Wan, J. Gao and Y. Lin, “Improving relation classification by entity pair graph”, in “Asian Conference on Machine Learning”, pp. 1156–1171 (2019).

Zheng, B., H. Wen, Y. Liang, N. Duan, W. Che, D. Jiang, M. Zhou and T. Liu, “Document modeling with graph attention networks for multi-grained machine reading comprehension”, in “Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics”, pp. 6708–6718 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.acl-main.599`.

Chapter 4

"LEN OR INDEX OR COUNT, ANYTHING BUT V1": PREDICTING VARIABLE NAMES IN DECOMPILATION OUTPUT WITH TRANSFER LEARNING

4.1 Introduction

Code Compilation, which transforms high-level source code into low-level machine code, is fundamentally a *lossy* procedure. Much semantic information, including control flow structures, function names, variable locations, variable names, variable types, and comments, is discarded during compilation because the target machine does not need such information. For example, a CPU does not understand the notions of variables, types, or loops (relying only on registers, memory, bytes, and branch statements), so the compiled output does not need these concepts.

This phenomenon of compilation-induced information loss makes it more difficult for human analysts to understand binary programs ("binaries") than to understand source code (Yakdan *et al.*, 2016), despite the fact that a compiler-generated binary encodes the same logic as the corresponding source code. To aid humans in such understanding, and support a number of downstream security tasks, researchers have developed a number of *de*compilation techniques, which take as input binary code, recover the lost semantic information from the binary, and derive roughly equivalent source code (or pseudocode, which generally is in an approximated version of C). State-of-the-art decompilers, e.g., IDA Pro's Hex-Rays decompiler (Hex-Rays, 2013), Ghidra (decompiler, 2022), and Binary Ninja (Ninja, 2022), are widely used in academia and industry. Security applications for decompilation include malware analysis (Ďurfina *et al.*, 2011, 2013; Yakdan *et al.*, 2016), vulnerability discovery in binary code (Mantovani *et al.*, 2022b), patching software defects (Schwartz *et al.*, 2013), protocol reverse engineering (Kalle *et al.*, 2019), and code reuse discovery (Mirzaei *et al.*, 2021).

However, decompilation is far from perfect, and significant problems continue to be addressed by researchers, including the reconstruction of code structure (Yakdan *et al.*, 2015a), inferencing of variable types (Lee *et al.*, 2011; Noonan *et al.*, 2016; Zhang *et al.*, 2021b; Chen *et al.*, 2022), and even the recovery of meaningful variable names (Lacomis *et al.*, 2019; Chen *et al.*, 2022). Variable name recovery is important because developers strive to properly name variables to *embed semantic meaning* and to improve readability (and maintainability) of source code (Schankin *et al.*, 2018). Unfortunately, unless debug information is preserved during compilation, these carefully-chosen variable names are lost, contributing to the difficulty of binary code understanding. Preserving debug information would also not be a solution since it would lead to increase in the size of the binaries. While malware analysis techniques have advanced over the years, human analysts continue to play a vital role in understanding binaries (Mantovani *et al.*, 2022a). Hence inferring variable names still remains an important task.

Contributions. We summarize our contributions as follows:

- We built a new neural-network model, VarBERT, to predict variable names and variable origins in decompilation output using transfer learning from source code samples.

- We identified issues in existing datasets DIRE and DIRT and built improved versions, and re-evaluated DIRE and DIRTY on them.

- We used novel techniques to build a new data set, addressing the identified issues of our predecessors. We also show various analyses to show the superiority of our dataset.

- We evaluated VarBERT on our new data set, VarCorpus and showed that it achieves 59.33% and 59.84% prediction accuracy on our data set, for IDA and Ghidra respectively. Our evaluation shows that VarBERT is applicable to variable name and origin prediction on decompilation output of stripped, real-world software.

In the spirit of open science, we will release our research artifacts, including all data sets, the source code, and our models, upon the acceptance of our paper.

4.2 Cybersecurity Background

Predicting variable names is built atop many layers of foundations. In this section, we provide the necessary background knowledge on these layers.

4.2.1 Binary Reverse Engineering

Binary reverse engineering usually refers to the process of analyzing binary programs without or with only limited access to the original source code. The obscurity of binary programs makes analyzing malware (Ďurfina *et al.*, 2011, 2013; Yakdan *et al.*, 2016), finding software vulnerabilities (Mantovani *et al.*, 2022b), and mitigating software defects (Schwartz *et al.*, 2013) extremely difficult, and the goal of binary reverse engineering is to alleviate this problem.

4.2.2 Human Binary Reverse Engineering

Ethical human analysts use binary reverse engineering techniques in malware analysis (Engineering, 2021), deobfuscation (Yadegari *et al.*, 2015), binary diffing (Bourquin *et al.*, 2013; Duan *et al.*, 2020), inferring data structures (Slowinska *et al.*, 2011; Lee *et al.*, 2011; Noonan *et al.*, 2016; Zhang *et al.*, 2021b), manually finding vulnerabilities (Votipka *et al.*, 2020), assisting automated vulnerability discovery (Shoshitaishvili *et al.*, 2015, 2017; Babić *et al.*, 2019; Jung *et al.*, 2021), and patching vulnerabilities in binary code (Wang *et al.*, 2017). Reverse engineering binary programs usually requires significant expertise and the use of sophisticated, sometimes very expensive, tools. Popular binary reverse engineering frameworks include IDA Pro (Hex-Rays, 2013), Ghidra (decompiler, 2022), Binary Ninja (Ninja, 2022), Hopper (Hopper, 2022), and angr (Shoshitaishvili *et al.*, 2016).

4.2.3 Binary Decompilation

First, binary reverse engineering tools must *disassemble* the binary code (essentially reversing the assembly process). Disassembling refers to the process of transforming machine code in a binary program to their corresponding human-readable instructions, and optionally recovers data, function boundaries, and control-flow graphs (CFGs).

Next, binary reverse engineering tools can attempt to decompile binary code (usually in response to a user's request). Decompilation is an intuitive solution for making binary code less obscure by recovering high-level source code (such as C code) from binary code (Schwartz *et al.*, 2013). Since the publication of the first paper on decompiling binary code (Cifuentes, 1994), modern decompilers have advanced in regards to both completeness and soundness. Human analysts are now using decompilation techniques to find bugs in software (Mantovani *et al.*, 2022b), discover reused code (Mirzaei *et al.*, 2021), analyze malware behaviors (Ďurfina *et al.*, 2011, 2013; Yakdan *et al.*, 2016), and patch software bugs (Reiter *et al.*, 2022). Due to the lossyness inherent in the compilation process, binary decompilation must attempt the task of recovering variables. Even this simple task can be complex and create decompilation artifacts: With only one variable in the original code, the decompiler may infer many variables. These could be due to operations of the binary code (e.g., storing and retrieving the same variable from the stack), to compilation operations (e.g., creating temporary variables), or to compiler optimizations (e.g., duplicating code or variables to improve performance). Therefore, the variables that exist in decompilation can be either *human-created* (i.e., in the original source code) or *extraneous* (i.e., not in the original source code).

Additionally, two supporting pillars for binary decompilation are (control-flow) structural analysis (Yakdan *et al.*, 2015b; Gussoni *et al.*, 2020), which attempts to identify high-level control-flow structures (e.g., if-else, for loops, and do-while loops), and variable type inferencing (Noonan *et al.*, 2016; Zhang *et al.*, 2021b; Chen *et al.*, 2022), which infers high-level types of variables (e.g., `int`, `char`, `enum`, pointers, and members of a `struct`). By enabling certain compilation flags (e.g., `-g` for GCC and Clang), compilers may preserve *debug information* either in the binary or as a separate file for debugging purposes.

4.2.4 Predicting Variable Names in Decompiled Code

The quality of decompilation output will be significantly lower than the original source code due to information discarded during compilation. A critical category of lost information is variable names. While modern decompilers attempt to infer some variable names in decompilation output in a rule-based manner (e.g., arguments passed to known library functions), they still leave a large portion of variables unnamed. Human analysts must understand the decompilation output, which is tedious, and rename unnamed variables one by one. Because variable names are critical in assisting with understanding the source code, the lack of such information in decompilation output severely hampers its readability.

4.3 Corpora Generation

Building a good corpus that is diverse and also representative of the target task is critical for training and evaluating any ML model.

4.3.1 Existing Datasets

Earlier researchers working in variable name inferencing have developed two datasets DIRE (Lacomis *et al.*, 2019) and DIRT (Chen *et al.*, 2022) which are publicly available. General statistics of these datasets are available in Table 4.1. These datasets are curated from various GitHub libraries. However, while attempting to use the data sets from DIRE (Lacomis *et al.*, 2019) and DIRT (Chen *et al.*, 2022), we identified several issues that we believe necessitate a new data set.

4.3.2 Issues With Existing Datasets

While investigating these data sets, we noticed several issues that we believe make them not suitable for future research in variable name prediction for decompiled code. **Significant Overlap Between Test and Training Sets:** In DIRE and DIRT, the test and training sets exhibit a significant overlap. Approximately 79.9% of functions in DIRE's test set exist in the training set. Therefore, the overall accuracy (74.3%) that DIRE reported does not reflect the true performance of their model as a variable name prediction solution. Instead, it only demonstrates *how well DIRE identifies functions that were known to their model during training.*

DIRE authors were aware of this issue and reported an accuracy of 35.3% for a body-not-in-train test set, where they eliminated from the test set all functions that exist in the training set. Similar is the case for DIRTY: The overlap in their test and train set is 65.5% (Chen *et al.*, 2022, Table 11) with variable name prediction accuracy of 35.1% for body-not-in-train functions (DIRTY, 2022). These numbers reflect the real performance for DIRE and DIRTY to generalize to new decompiled functions.

High number of duplicated functions. Another issue we discovered in DIRE and DIRT is the high number of duplicated functions. 683K (68.6%) functions among DIRE's 1M functions that are used for training DIRE are duplicates. Similarly, 1M (56.9%) out of 1.8M are duplicated in the training set of DIRTY.

The nature of neural network models will cause models trained on these data sets to learn more from frequently appearing functions and ignore those less frequently appearing functions. We do not believe the distribution in DIRE and DIRT reflect the real distribution of functions for binaries that users will decompile in the real world.

High failure rate of variable matching. Another issue with DIRE and DIRT is the high failure rates of variable matching. We understand that decompilers are unable to recover all of the variable names from source code, however oftentimes the variable matching algorithm of DIRE and DIRT's corpus creation is unable to match the original human-created variable to the decompiler-assigned variable. We performed analysis on a sample of functions from each corpus in Section 4.3.6. Incorrect matching or missing matches between human-created variable names and decompiler-generated names impacts the ground truth of the data set.

Single Decompiler Generated Dataset: A minor issue that might hamper the generalizability of models trained on DIRE and DIRT is that they are IDA-only corpora. This is because of drawbacks to their corpora generation: DIRE requires legitimate IDA-generated ASTs for training, while DIRT could not match variables in Ghidra's decompilation against DWARF information *post-mortem*. Also, decompiled codes generated using each decompiler are drastically different. Therefore, future techniques built on DIRE and DIRT might overfit to IDA's decompilation output and cannot demonstrate their applicability to other decompilers.

In conclusion, we believe DIRE and DIRT are unsuitable for future research on predicting variable names in decompiled code (but the data sets may still be useful for other purposes). We built a new corpus for VarBERT with these weaknesses and limitations in mind.

4.3.3 Improving Existing Datasets

To better understand the impact of duplicated functions on neural models, we attempted to address the duplication problem in the DIRE and DIRT corpra by fully de-duplicating functions to create fixed data sets that we call DIRE-dedup and DIRT-dedup.

We deduplicated individually each of the training, validation, and test set of DIRE and DIRT. As a result, the number of functions in the training data gets reduced from 1M to 327K approximately in DIRE-dedup and from 1.6M to 717K approximately in DIRT-dedup. In Section 4.7.1, we will evaluate VarBERT on these two new corpora.

4.3.4 Building Our Human-Source-Code (HSC) Dataset

The first type of corpora comprises annotated source code functions that human developers author and are used for pre-training. We collected C source code files from the Debian APT repository, then parsed and pre-processed these files. The goal of pre-processing is to make source code resemble decompilation output, as required by transfer learning. Pre-processing includes comment removal, macro expansion, invalid identifier removal, etc. Finally, we annotated these files to indicate the variables in each function. This resulted in a total of 5,235,792 C functions. For the rest of the paper, we refer to this corpus as the *Human-Source-Code (HSC) corpus.*

4.3.5 Building Our VarCorpus Dataset

Compiling Packages: The second type of corpora is decompilation output which is generated by decompilers and will be used as input for fine-tuning. We collected C packages from the Gentoo package repository, and built them targeting x86-64 for three compiler optimizations: `-O0` (no optimization), `-O1`, and `-O2` with debug symbols preserved (`-g`). We stopped at `-O2` because we found that decompilers failed a significant amount of time on `-O3` binaries. We had a total number of 62,650 deduplicated binary executables or libraries prior to decompilation. This number includes binary executables for all three compiler optimizations.

Decompiling Binaries & Matching Variable Names: We processed the debug symbols for each binary, stripping their type information, and leaving only human-created variable names. This is important because it eliminates the impact that debug symbols (especially type information) have on decompilation output. Then, we decompiled each binary with two decompilers, IDA and Ghidra, and collected the decompilation output per function. We chose IDA and Ghidra because, out of all the binary decompilers we tested, only IDA and Ghidra could generate C-style pseudocode with acceptable quality. Other decompilers either failed to decompile many functions, did not support debug information, or could not generate C-style pseudocode. We decompiled the binaries (compiled keeping the debug symbols preserved) and binaries without type or debug information. Then we were able to reliably map the decompiler-generated variables to the human-created variables at function-level. This also helps us to identify whether a variable is decompiler-generated (extraneous) or human-created which we call *variable origin*. This way, we eliminate the need for variable matching heuristics used by prior work (which was only about 59% accurate) (Jaffe *et al.*, 2018; Lacomis *et al.*, 2019).

Data set	C.O.	Unique Variables	Functions	Binaries
HSC	N/A	3,561,537	5,235,792	N/A
VarCorpus (IDA)	O0	849,192	2,608,873	19,815
	O1	239,312	655,376	14,015
	O2	187,065	527,646	13,347
VarCorpus (Ghidra)	O0	445,332	1,994,190	18,008
	O1	168,005	803,974	14,826
	O2	158,701	731,666	15,325
DIRE (Lacomis *et al.*, 2019, RQ4)*	O0	92,082	1,259,935	164,632
DIRE-Dedup	O0	92,082	463,238	N/A
DIRT (Chen *et al.*, 2022, Table 11)*	O0	237,928	2,075,762	75,656
DIRT-Dedup	O0	237,928	995,418	N/A

Table 4.1: Summary of all data sets, including numbers of functions, unique variable names, and numbers of binaries. "C.O." means Compiler Optimization.

We also removed irrelevant functions (e.g., PLT and glibc stubs) from the collection, then annotated the remaining functions to indicate the locations of variable names in each function. We call this corpus *VarCorpus*. VarCorpus-O0 is a corpus created from binaries compiled with -O0 compiler optimization and so on. We provide details of VarCorpus in Table 4.1.

4.3.6 Evaluating VarCorpus Quality:

Here we show that VarCorpus is reasonable and improves over the state-of-the-art corpora.

Binaries: An essential difference between VarCorpus and DIRT (also DIRE) is the definition of "binaries." DIRT and DIRE use compiled object files whereas VarCorpus only includes binary executables or libraries. Since each executable may be linked

	C.O.	Min	Max	Average
VarCorpus (IDA)	O0	4	11,142	36.09
	O1	4	7,693	36.54
	O2	4	8,999	36.48
VarCorpus (Ghidra)	O0	5	11,759	31.58
	O1	5	186,22	29.92
	O2	5	10,434	31.23
DIRE	O0	5	1,295	24.12
DIRE-Dedup	O0	5	1,285	27.56

Table 4.2: The Distribution of Function Lengths in VarCorpus And DIRE. "C.O." Means Compiler Optimization.

from tens, hundreds, or thousands of object files, the numbers of binaries in VarCorpus are not comparable to the ones in DIRE and DIRT.

Corpora Sizes and Duplicated Functions: Table 4.1 shows the summary of all data sets. DIRE and DIRT both contain a high number of duplicated functions in their training sets (and test sets). VarCorpus has no duplicate functions. Additionally, VarCorpus is more diverse. For O0 data sets, VarCorpus contains over 1.6x more unique functions than DIRT-Dedup and 4.6x more unique functions than DIRE-Dedup. Finally, the number of unique variables in VarCorpus-O0 is 3.57x of the number of unique variables in DIRT.

Overlap Between Test and Training Sets: The test set of DIRE has an overlap of 79.9% and the test set of DIRT has an overlap of 65.5% with their training sets. Because there is no function duplication in VarCorpus, our test and training sets also do not overlap.

Function Lengths: Table 4.2 shows the distribution of function lengths (lines of

	C.O.	Entropy	
		Train	Test
HSC	NA	14.56	14.66
VarCorpus (IDA)	O0	12.59	12.45
DIRE	O0	11.00	10.74
DIRT	O0	9.06	9.20

Table 4.3: Shannon's Entropy of Variables on Four Corpora.

code) in VarCorpus and DIRE. We can see that while VarCorpus and DIRE have similar minimum lines of code, VarCorpus includes roughly 10 times larger functions than DIRE, showing better diversity. We cannot get the line count from DIRT because only split tokens (without newlines) are provided in the data set.

Distribution of variable in terms of Shannon Entropy. To better understand the variable diversity, we use Shannon's entropy on variable names as in prior work (Dramko *et al.*, 2022). The higher Shannon's entropy is, the higher diversity there is among all variables in a corpus, and the less likely the trained model will skew towards common variable names. Table 4.3 shows Shannon's entropy.

Our HSC data set is the biggest corpus with the most number of unique variable names. Naturally, the HSC data set has the highest entropy of 14.56 which seconds the richness and diversity of variable names. Similarly, VarCorpus-O0 (IDA) has an entropy of 12.59, much higher than DIRT (11.00) and DIRE (9.06).

Variable name matching. Finally, to understand the performance of variable matching in DIRE, DIRT, and VarCorpus-O0 (IDA), we randomly sampled 5,000 functions in each corpus, and measured the ratio between developer-assigned variable names and the total number of variable names. Assuming IDA injects on average similar amount of extraneous variables during decompilation for binaries built with the same optimization level, the ratios on both corpora should be similar. However, through random sampling, we measured a ratio of 56% on DIRE, 59% on DIRT, and 74% on VarCorpus (IDA). This shows that we correctly mapped more developer-assigned variable names to variables in decompilation output than DIRE and DIRT.

Based on the above data, we believe that VarCorpus represents a significant improvement over the prior variable name prediction data sets, and should serve as a benchmark for future research in this area.

4.4 Our Approach

We draw intuition of our approach of variable name inferencing from the transfer learning approach of modern AI systems. Researchers in NLP and computer vision proposed (and demonstrated the success of) *transfer learning* (Conneau *et al.*, 2017; McCann *et al.*, 2017; Deng *et al.*, 2009; Yosinski *et al.*, 2014), wherein parameter values originally trained for Task A can be re-used as initial parameter values when training a neural model for Task B if A and B are similar. Creating an initial model for Task A is called *pre-training*, and the second training run on Task B is called *fine-tuning*. The pre-training approach allows a system to tune it to he domain of data by learning the initial representation.

Given the obvious similar nature between source code and decompilation output, transfer learning (pre-training on source and fine-tuning on decompilation output) becomes a natural choice. It also reduces the need for clean the large task-specific data sets. Here, we use transfer learning to compensate for the difficulty of creating a large corpus of clean decompilation output: We pre-train on easy-to-obtain source code and then fine-tune on the decompilation task.

4.4.1 The VarBERT Model

We develop a variation of BERT model to customize for the task of learning the code representation. As a transformer-based model, basic parameters of VarBERT are the number of neural layers L, the number of self-attention heads A, and the hidden dimension H. Training a huge transformer-based model is computationally heavy and time-consuming. Limited by accessible computing resources, VarBERT has fewer layers than the original BERT. We take initial hyper-parameters from RoBERTa (Liu *et al.*, 2019b), which demonstrated an empirically optimal hyper-parameter configuration for BERT on NLP tasks. We hypothesized that for the task of variable name prediction a model might perform well, and created a model with $L = 6$, $A = 8$, and $H = 512$. Our model has 45 million trainable parameters, which is about 40% of the number of trainable parameters of BERT-Base (Liu *et al.*, 2019b).

4.4.2 Tokenization Scheme

The natural language BERT model is familiar with English word vocabulary. However, English is quite different from source code and decompiled code in terms of structure, syntax, and keywords. For the model to better understand these, we first learn a new source vocabulary using a Byte-Pair Encoding (BPE) tokenizer. Following the RoBERTa (an optimized version of the original BERT model) (Liu *et al.*, 2019b) tokenizer, we consider learning a source vocabulary of 50,265 most frequently occurring tokens from the training dataset of our human source code (HSC) corpus. We add a special mask token $\langle mask \rangle$, along with four usual tokens: start token$\langle s \rangle$, pad token $\langle pad \rangle$, end token $\langle /s \rangle$, and unknown token $\langle unk \rangle$. While learning the vocabulary of 50K we keep those frequent pairs which occurs at least 2 times in the text.

4.4.3 Pre-Training

We pretrain our VarBERT model on HSC corpus to learn the nature of text. In this process the model learns contextual representation of each token of the input text and gets ready for down-stream task of variable name prediction.

Masked Language Modeling: Here, our motivation is to make VarBERT familiar with the nature of input data. We first tokenize each function of the HSC corpus and generate a token stream using the learned vocabulary. Finally, we learn the representation of the code-tokens using BERT from scratch by Masked Language Modeling approach similar to the approach given in RoBERTa (Liu *et al.*, 2019b). We randomly mask tokens and ask the model to recover the masked token, and in the process, learn rich representations for the code-tokens. For MLM we use the Whole Word Masking (Devlin *et al.*, 2019a) technique so as to keep the variable names preserved and learn the correlation of each whole variables with the contexts.

Constrained Masked Language Modeling: Next, we formulate another pre-training task, Constrained MLM (a variation of MLM), which was proposed in prior work (Donahue *et al.*, 2020; Gu *et al.*, 2020), where tokens are not randomly masked. The motivation is to teach the model the task of selectively recovering specific code tokens. We define Constrained MLM as follows: Let $W_0, ...W_N$ be a sequence of tokens. Let $C = \{A_0, ..A_c\}$ be a set of tokens which we define as the constrained set of tokens. Then, in Constrained MLM, all tokens in $W_0, ...W_N$ which belong to C are masked, and C is a subset of V. We then train the model to predict the masked token with a cross-entropy loss:

$$\mathcal{L} = -\sum_{i=0}^{N}\sum_{c=1}^{M} y_{w_i,v_c} \log(p_{w_i,v_c}) \tag{4.1}$$

where M is the size of the vocabulary, w_i is the current token, v_c is the target token,

y is an indicator variable which is 1 if the target is v_c and 0 otherwise, and p is the probability that w_i is the same as v_c. Here, we select a vocabulary of 50K most frequently occurring human-authored variable names from the HSC corpus. Then, we selectively mask the variable names and train the model to predict them based on the vocabulary.

4.4.4 Fine-Tuning

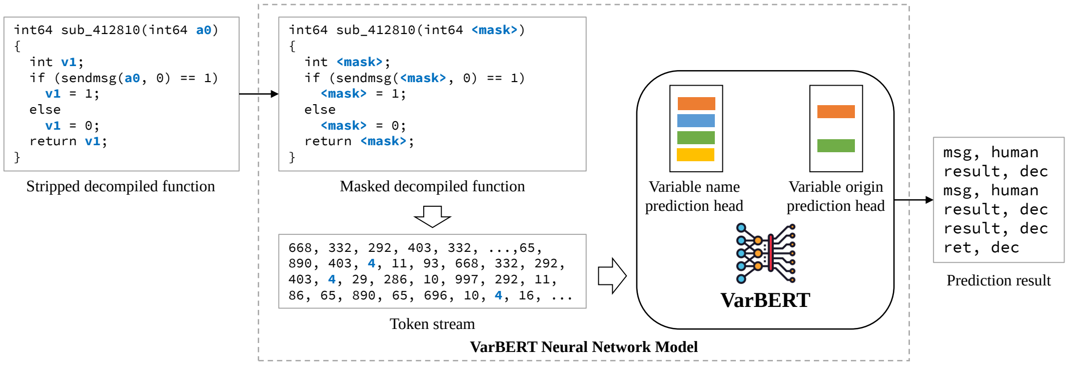
After pre-training on general source code, we fine-tuned our pre-trained models for the variable name prediction task and the variable origin prediction task. Here we again use the Constrained MLM task by masking variables in the training corpus for each decompiler. We first develop a target vocabulary of frequently occurring variable names from the training data along with earlier chosen 50K human-authored variable names. We optimized the model parameters by minimizing the overall joint cross-entropy loss $\mathcal{L}(\mathbf{X}, \mathbf{Y})$ by taking the mean loss of N mini-batches (4.2), where $\mathbf{X}$ and $\mathbf{Y}$ are the input and labels respectively.

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \frac{1}{N} \sum_{n=1}^{N} L_n \tag{4.2}$$

For the variable name prediction task, each mini-batch loss (L_n) is the sum of all variable-name prediction loss (l_v^t) in that mini-batch. For the joint prediction task, each mini-batch loss is the sum of joint loss of the predicted name (l_v^t) and the predicted origin (l_o^t) for each variable (t) in that mini-batch (n). Equation (4.3) formalizes this, where T_n is the total number of variables to predict in a particular mini-batch (n).

$$L_n = \sum_{t=1}^{T_n} (l_v^t + l_o^t) \tag{4.3}$$

Our VarBERT approach is shown through the end-to-end diagram 4.1. We further



```
int64 sub_412810(int64 a0)
{
  int v1;
  if (sendmsg(a0, 0) == 1)
    v1 = 1;
  else
    v1 = 0;
  return v1;
}
```

Stripped decompiled function

```
int64 sub_412810(int64 <mask>)
{
  int <mask>;
  if (sendmsg(<mask>, 0) == 1)
    <mask> = 1;
  else
    <mask> = 0;
  return <mask>;
}
```

Masked decompiled function

```
668, 332, 292, 403, 332, ...,65,
890, 403, 4, 11, 93, 668, 332, 292,
403, 4, 29, 286, 10, 997, 292, 11,
86, 65, 890, 65, 696, 10, 4, 16, ...
```

Token stream
Variable name prediction head
Variable origin prediction head
VarBERT
VarBERT Neural Network Model

```
msg, human
result, dec
msg, human
result, dec
result, dec
ret, dec
```

Figure 4.1: The Prediction Pipeline of VarBERT. The decompiled Code is Variable-masked And Tokenized Into a Token Stream, Which Is Used As Input To The Model For Prediction. VarBERT Predicts Both Variable Name And Origin For Each Masked Location

formalize both cross-entropy loss functions in Equations (4.4) and (4.5) where C_1 and C_2 are vocabulary lengths of the variable name prediction task and the variable origin prediction task, respectively.

$$l_v^t(\mathbf{x}, \mathbf{y}) = \log \frac{exp(x_{ty_t})}{\sum_{c=0}^{C_1-1} exp(x_{tc})} \tag{4.4}$$

$$l_o^t(\mathbf{x}, \mathbf{y}) = log \frac{exp(x_{ty_t})}{\sum_{c=0}^{C_2-1} exp(x_{tc})} \tag{4.5}$$

Handling Long Functions Although we increased the maximum input size of our VarBERT to 800 tokens (from BERT's 512), we still encounter many functions that have more tokens. To predict variable names for functions with more than 800 tokens, VarBERT splits these function bodies into 800-token chunks and considers them as separate samples during prediction.

4.5 Experiments:

4.6 Implementation

We implement VarBERT using Python. For decompilation when building the fine-tuning corpora, we used the Hex-Rays decompiler (in IDA Pro 7.6) and Ghidra 10.1. For building and training the neural model, we used PyTorch (Paszke *et al.*, 2019) 1.9.0, HuggingFace Transformers (Wolf *et al.*, 2020b) 4.10.0, and Facebook FairSeq (Ott *et al.*, 2019) 0.10.0.

4.6.1 Hyper Parameters

We optimized our models using BERTAdam (Devlin *et al.*, 2019a; Kingma and Ba, 2014) with the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e - 6$, and L_2 weight decay of 0.01. We warmed up over the first 10,000 steps to a peak value of $1e - 4$ and then linearly decayed. We set the dropout to 0.1 on all layers and attention weights. Our activation function was GELU (Hendrycks and Gimpel, 2016). We trained all our models for 30 epochs, except for experiments about evaluating DIRE on DIRE where we trained for 60 epochs (so that our results are comparable to the ones in the DIRE paper, where models were trained for 60 epochs). We performed all training and experiments on four 81GB Nvidia A100 GPUs with a training batch size of 16.

4.6.2 Training

4.7 Results And Analysis

Throughout this evaluation, we seek to measure the performance of variable name prediction systems in predicting variable names in previously unforeseen functions. Therefore, unlike prior research like DIRE and DIRTY, we will only present the results from data sets where there is no duplication of functions between the training and testing sets. In almost all experiments, we only consider top-1 accuracy—that is, does the model predict with the highest confidence the same variable name that the developer used? This is to showcase the lower-bound usefulness of variable name prediction systems, even "close" guesses such as `buf` for `buffer` would be considered incorrect. We leave it to future research to address the problem of evaluating the correctness and usefulness of variable name prediction results.

In this evaluation, we attempt to answer the following research questions:

RQ1. How Do VarBERT Compare With Prior Works On DIRE And DIRT?

RQ2. How Effective Is VarBERT On VarCorpus?

RQ3. How Do Different Aspects of VarBERT Impact Its Effectiveness?

Data set	Model	Top-1 Accuracy (%)
DIRE	DIRE (Lacomis *et al.*, 2019, Table 1)*	35.3
	DIRTY (Chen *et al.*, 2022, Table 4)*	42.8
	DIRECT (Nitin *et al.*, 2021, Table 1)*	42.8
	VarBERT (no PT)	51.24
	VarBERT	**61.49**
DIRE-dedup	DIRE	38.29
	VarBERT	**61.73**
DIRT	DIRE (Lacomis *et al.*, 2019, Table 1)*	31.8
	DIRTY (Chen *et al.*, 2022, Table 4)*	36.9
	VarBERT (no PT)	47.11
	VarBERT	**51.28**
DIRT-dedup	DIRTY	41.25
	VarBERT	**51.02**

Table 4.4: Results of DIRE, DIRTY, DIRECT, and VarBERT on DIRE, the fixed DIRE-dedup, DIRT, and the fixed DIRT-dedup. To understand the impact of pre-training, we also ran VarBERT without pre-training on HSC, indicated as VarBERT (no PT). To save computation resources we did not re-run experimental results for directly comparable results, and results of models marked with an asterisk* are taken from the indicated paper.

4.7.1 RQ1: How Do VarBERT Compare With Prior Works On DIRE And DIRT?

In this first experiment, we compare VarBERT, DIRE (Lacomis *et al.*, 2019), DIRECT (Nitin *et al.*, 2021), and DIRTY (Chen *et al.*, 2022)'s performance on the original DIRE, the fixed DIRE-dedup, original DIRT, and the fixed DIRT-dedup data sets. In addition, we also evaluate VarBERT without pre-training on HSC, indicated as VarBERT (no PT). The goal of this experiment is to demonstrate how much of VarBERT's performance gain is due to the transfer learning on HSC.

Result. Table 4.4 shows the result of this experiment. Note that to save computational resources in cases where results from papers were directly comparable (on the same dataset) we included the results from prior papers.

The results show that, on the original DIRE, VarBERT *without pre-training* outperformed prior work: 35.3% top-1 accuracy for DIRE, 42.8% for DIRTY, 42.8% for DIRECT, and 51.24% for VarBERT without pre-training. However, VarBERT *with* pre-training increased the top-1 accuracy to 61.49%. This same result also held for the original DIRT dataset: 31.8% top-1 accuracy for DIRE, 36.9% for DIRTY, and 47.11% for VarBERT without pre-training. And VarBERT *with* pre-training increased the top-1 accuracy to 51.28%.

To measure the impact of duplication in the DIRE and DIRT datasets, we re-ran DIRE and VarBERT on DIRE-dedup and DIRTY and VarBERT on DIRT-dedup. The results in Table 4.4 show that DIRE's accuracy improves on the fixed DIRE-dedup from 35.3% to 38.29%, while VarBERT maintains a high accuracy of 61.49%. Likewise, DIRTY's accuracy improves from 36.9% to 41.25% on the fixed DIRT-dedup, while VarBERT maintained a high accuracy of 51.02%.

In conclusion, VarBERT outperforms all prior models: even without any pre-training VarBERT outperforms the state-of-the-art, DIRTY's model, by 10.21%. This means that VarBERT is a strict improvement over the state-of-the-art model. Therefore, in the following evaluations, we focus only on evaluating VarBERT on our improved data set VarCorpus.

	C.O.	Split	Variable Name				Variable Origin	
			Top-1	Top-3	Top-5	Top-10	Accuracy	F1
VarCorpus (IDA)	O0	Function	59.33	68.03	70.89	74.14	90.44	89.07
		Binary	45.12	53.24	56.16	59.71	89.46	87.91
	O1	Function	57.79	66.82	69.84	73.31	83.86	83.48
		Binary	44.47	52.11	54.96	58.53	82.21	81.70
	O2	Function	58.39	67.70	69.89	73.21	81.24	81.22
		Binary	44.78	52.21	54.94	58.27	79.69	79.66
VarCorpus (Ghidra)	O0	Function	59.84	69.11	72.13	75.59	87.45	87.44
		Binary	55.51	62.86	65.42	68.52	87.86	87.86
	O1	Function	62.22	70.03	72.47	75.34	89.11	87.78
		Binary	46.70	53.47	55.87	58.80	87.70	86.11
	O2	Function	62.28	69.99	72.50	75.40	89.54	87.42
		Binary	50.54	57.26	59.67	62.57	88.37	85.72

Table 4.5: Evaluation of VarBERT's variable-name-prediction task fine-tuned on VarCorpus for different corpus optimization levels (C.O.) and either a function-level split or a binary-level split. Values in Top-N columns are accuracy rates of human-created variable names in percentage: Top-N means the correctly predicted variable is present among the first N predicted variable names.

4.7.2 RQ2: How Effective Is VarBERT On VarCorpus?

Given the performance of VarBERT on corpora from prior work, which we demonstrated in Section 4.3.2 are flawed, and the quality of VarCorpus that we evaluated in Section 4.3.6, we now focus on evaluating VarBERT on VarCorpus to answer RQ2: How Effective Is VarBERT On VarCorpus?

We took six corpora from VarCorpus, three for IDA (the IDA corpus), and another three for Ghidra (the Ghidra corpus) across three different compiler optimizations: `-O0`, `-O1`, and `-O2`. In a ratio of 80:20, we split each corpus into training and test sets.

We also experimented with two data splitting approaches: (a) randomly splitting data by function (per-function) and (b) randomly splitting data by binary (per-binary). Please note that, as discussed in Section 4.3.6, VarCorpus per-binary split (where "binaries" are executables) is different from DIRT and DIRE (where "binaries" are object files). This split is important because the per-function split might have functions that are from the same source project in both testing and training (although not the same functions in testing and training, as there is no duplication), while with the per-binary split, this will not occur (although, it is possible for two binaries to be from different projects and include similar library functions, even if not identical, such as different versions of a library). In this sense, we expect it to be more difficult to predict when using the per-binary split than the per-function split.

We pre-trained VarBERT on human source code (HSC) and then fine-tuned it separately on each corpus: IDA corpus and the Ghidra corpus.

Finally, we evaluated VarBERT on two testing sets for two tasks: Predicting the origin of each variable (extraneous versus human-created) and predicting the name for each human-created variable.

Variable origins. The right two columns of Table 4.5 show the results of the variable-origin-prediction task. VarBERT can predict whether a variable is human-created or extraneous roughly 80% to 90% of the time. We believe that these results can help decompilers to produce *improved* decompilation results that are closer to the source code (an orthogonal benefit to variable name prediction).

Variable names. For the variable-name-prediction task, we only consider how well VarBERT performs. Table 4.5 shows the results of the variable-name-prediction task on the IDA and Ghidra corpora.

Because both VarCorpus-O0 (IDA) and DIRTY are generated from O0 binaries, we can compare their results. VarBERT achieved top-1 accuracy of 59.33% on IDA corpus and 59.84% on Ghidra courpus, when split on a per-function basis. We believe these numbers are comparable to DIRTY's accuracy on DIRT's not-in-train test set, which was 36.9% as reported in their paper (Chen *et al.*, 2022).

This result shows that VarBERT learns variables' semantics better from their context and generalizes better to functions on which the model was not trained (because there is no function duplication in VarCorpus).

Variable names on optimized binaries. VarCorpus allows the evaluation of VarBERT's performance on binaries compiled with optimizations enabled. Much to our surprise, we do not observe any significant drop in accuracy when predicting variable names for O1 and O2 binaries using per-function splits, and this finding is consistent across both decompilers. This result clearly shows the applicability of VarBERT for predicting variable names in real-world binaries, which are often built with optimizations enabled.

Variable names on per-binary splits. As we anticipated, the accuracy of VarBERT when working on per-binary splits is decreased compared to its accuracy on per-function splits. The difference with the same decompiler-optimization pair is usually between 10% and 15%. Through preliminary analysis, we believe the root problem is that our vocabulary was created using development and training sets. In per-binary splits, due to the obvious Out-Of-Distribution issue, any variable names that only appear in test tests cannot be predicted. Because there are many more unique variables in IDA-O0 than in Ghidra-O0 (849k versus 445k), IDA-O0 suffered more heavily. We believe this highlights an interesting research direction for future work to address.

Non-Exact Match Performance: In Tables 4.4 and 4.5, we evaluate VarBERT using the hardest metric: *Exact-Match (EM)*, where we consider the model prediction to be correct only if the prediction exactly matches the reference variable names. For example, even if VarBERT predicts `tmp` or `tmp2` in place of `tmp0`, we consider these predictions incorrect even though they carry the same semantic meaning and can assist humans in binary understanding. So we introduce two non-exact-match metrics like *Average Edit Distance* (**AED**) and *Average Character Error Rate* (**ACER**) and show the performance of VarBERT on VarCorpus, DIRE and DIRT in Table 4.6.

We find that, for both the compilers, with more optimization, the AED increase indicating that VarBERT finds it difficult to learn and predict the accurate variable names. The ACER, however, does not change significantly which we believe is because the model was able to find similar names on most of the occasions irrespective of the optimizations. Another observation is that both the metrics are higher in the binary-level split than function-level split, which is because the variables in the binary-level split are much more difficult to predict than function-level split simply because here there is no opportunity of learning the commonalities of the nature of the intra-binary functions.

	C.O.	Split	AED	ACER
VarCorpus (IDA)	O0	Function	1.79	0.36
		Binary	2.31	0.50
	O1	Function	2.07	0.41
		Binary	2.48	0.53
	O2	Function	2.11	0.41
		Binary	2.57	0.55
VarCorpus (Ghidra)	O0	Function	1.93	0.40
		Binary	1.95	0.42
	O1	Function	2.16	0.41
		Binary	2.76	0.57
	O2	Function	2.44	0.53
		Binary	2.73	0.52
DIRE	O0	Original	1.94	0.41
		Dedup	1.68	0.36
DIRT	O0	Original	1.88	0.44
		Dedup	1.87	0.44

Table 4.6: Evaluation Of VarBERT's Variable-Name-Prediction Task On Each Of The Datasets: VarCorpus, DIRE And DIRT Using Two Non-Exact-Match Metrics: *Average Edit Distance* (**AED**) and *Average Character Error Rate* (**ACER**).

Comparison With Simple Baselines: Since our formulation of Variable Name Inference is a prediction task across a large number of variable classes (150K for VarCorpus and DIRT and around 100K for DIRE), we show the *Most Frequent (MF) Baseline* performance in Table 4.7. We choose the most frequently occurring variable from the training data of each of the datasets and consider that as the model prediction. It can be seen that for each of the datasets, the MF top-1 exact-match accuracy is less

	C.O.	Split	MF Top-1 Accuracy (%)
VarCorpus (IDA)	O0	Function	3.15
		Binary	2.96
	O1	Function	2.30
		Binary	2.34
	O2	Function	2.53
		Binary	2.52
VarCorpus (Ghidra)	O0	Function	5.90
		Binary	6.17
	O1	Function	1.47
		Binary	1.32
	O2	Function	1.44
		Binary	1.21
DIRE	O0	Original	6.93
DIRT	O0	Original	3.85

Table 4.7: Most Frequent (MF) Baseline: Selecting Most Frequently Occurring Variable In The Training Data

than 7%. Apart from that, the *Random Baseline* selecting a random variable from training data as model prediction is 0%. *This shows that our model performance is not biased on the training data variable names.*

4.7.3 RQ3: How Do Different Aspects of VarBERT Impact Its Effectiveness? An Ablation Study.

The Impact of Pre-training: As shown in Table 4.4, we measured the impact of pre-training by evaluating VarBERT and VarBERT (no PT) on DIRE. The improvement of 10 percentage points (61.49% versus 51.24%) shows the necessity of pre-training.

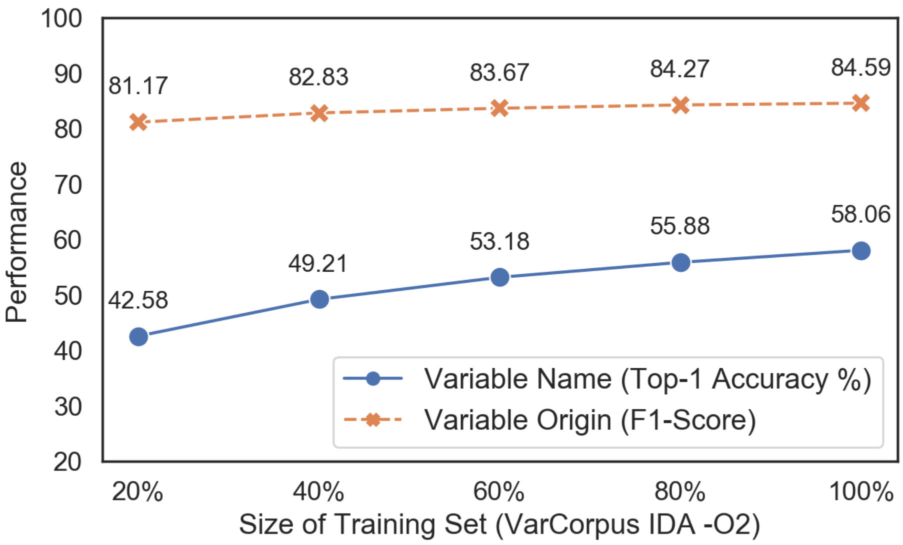
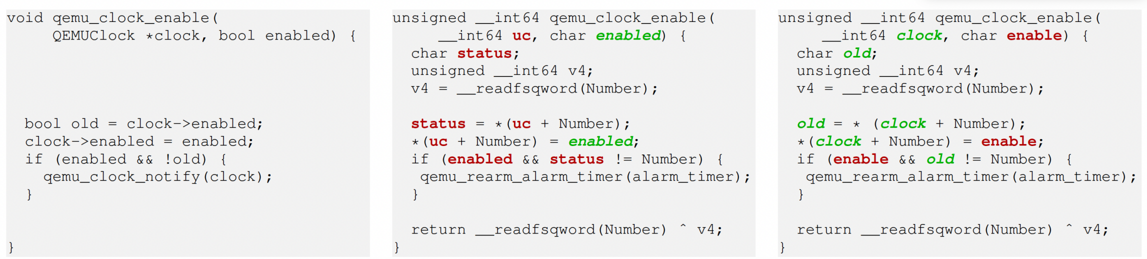


Figure 4.2: The Impact Of Corpora Sizes On VarBERT's Performance.

The Performance of VarBERT on CMLM pre-training Task: This constrained MLM task tunes the model to the task of retrieving the missing variable for a masked-out location on a source code. This task shows around 4% and 1% improvements in DIRE and DIRT datasets. VarBERT shows 54.12%, 66.87%, 71.18% and 75.99% in top-1, top-3, top-5, top-10 in HSC-Test dataset.

The Impact of Corpus Size: To study the impact of corpus size on the accuracy of VarBERT, we train VarBERT separately on 20%, 40%, 60%, and 80% of VarCorpus-O2 (IDA). Figure 4.2 shows the result. With 40% training data, our model achieved good performance on both the variable name prediction task (49.21%) and variable origin prediction (82.83 F1 score). With more data, the accuracy of VarBERT on the variable name prediction task increases steadily, which shows its potential to improve further with more training data.



```
void qemu_clock_enable(
     QEMUClock *clock, bool enabled) {



  bool old = clock->enabled;
  clock->enabled = enabled;
  if (enabled && !old) {
    qemu_clock_notify(clock);
  }


}
```

(a) Original source code.

```
unsigned __int64 qemu_clock_enable(
     __int64 uc, char enabled) {
  char status;
  unsigned __int64 v4;
  v4 = __readfsqword(Number);

  status = *(uc + Number);
  *(uc + Number) = enabled;
  if (enabled && status != Number) {
   qemu_rearm_alarm_timer(alarm_timer);
  }

  return __readfsqword(Number) ^ v4;
}
```

(b) DIRTY predicted variable names.

```
unsigned __int64 qemu_clock_enable(
     __int64 clock, char enable) {
  char old;
  unsigned __int64 v4;
  v4 = __readfsqword(Number);

  old = * (clock + Number);
  *(clock + Number) = enable;
  if (enable && old != Number) {
   qemu_rearm_alarm_timer(alarm_timer);
  }

  return __readfsqword(Number) ^ v4;
}
```

(c) VarBERT predicted variable names.

Figure 4.3: Example Case Study On Function `qemu_clock_enable` From The DIRT Dataset. Figure (a) Shows the Actual Source Code For Reference. (b) Shows DIRTY's Prediction (c) Shows VarBERT Prediction. Correctly Predicted Variables Are In ***Italic Green*** While Incorrect Predictions Are In Styled **Red**. Variable `v4` is Extraneous Variable And Hence Not Predicted By Either Of The Systems

4.8 Case Studies

4.8.1 DIRTY and VarBERT Comparison on DIRT

We show an example of variable name prediction results to examine here as a case study. We chose the function `qemu_clock_enable` from the DIRT data set and compared the variable name prediction results of DIRTY and VarBERT on it.

To assist in the comparison, Listing 4.3(a) shows the original source code of the `qemu_clock_enable` function (which we were able to find using the decompilation output). Next to Listing 4.3(a), we show the decompilation output with the variable name prediction results of DIRTY in Listing 4.3(b) and VarBERT in Listing 4.3(c). In the decompilation output, we removed type casts (which is a keyboard shortcut in IDA) for easier comparison between the outputs. In both the decompilation outputs, variables that are predicted correctly are styled ***italic green*** while variables that are predicted incorrectly are styled **red**.

The function in Listing 4.3(a) has three developer-intended variables: `clock`, enabled, and old. DIRTY in Listing 4.3(b) was able to correctly predict enabled, but failed for the others, predicting clock as uc and old as status. However VarBERT in Listing 4.3(c) correctly predicted clock and old but was incorrect in predicting enable for enabled. Note that even though enable is very semantically similar to enabled, the strict design of our evaluation counts this as a failure when considering top-1 accuracy.

4.8.2 Mispredictions

VarBERT and DIRTY both consider a prediction correct if it is a strict match. But some of these mispredictions are as valuable as predictions. We looked into VarBERT's prediction results when running on VarCorpus-O0 (IDA) to understand its mispredictions. Table 4.8 shows examples of developer-intended variables, the percentage of time that the correct variable was predicted along with the top-3 incorrect predictions. For instance, the original variable name substring_n was only predicted correctly 50% of the time, and the other incorrect predictions were substring1 and substring. For a human, the semantic meaning of these predictions is quite similar, however, we count these predictions as mispredictions.

We also noticed that the model seemed to pick up on nuances of the English language. For example the VarBERT prediction from Section 4.8.1 in Listing 4.3(c): VarBERT predicted enabled as enable. These two words share the same root, but this is not a correct prediction. We see a similar misprediction trend with the variable buffer, where buf and buffer are essentially the same word.

4.9 Discussion

While VarBERT improves the feasibility of variable name prediction on real-world decompiled code by a sizable margin, we envision that this research challenge will still

	Correct Predictions	**Top-3 Incorrect Predictions**		
`buffer`	`buffer` (63.55%)	`buf` (8.95%)	`UNK` (5.35%)	`data` (2.43%)
`len`	`len` (73.57%)	`UNK` (5.65%)	`size` (2.80%)	`n` (1.84%)
`tmp1`	`tmp1` (48.56%)	`tmp2` (12.83%)	`tmp0` (11.55%)	`UNK` (5.56%)
`srcsize`	`srcsize` (42.86%)	`len` (28.57%)	`length` (14.29%)	`size` (7.14%)
`substring_n`	`substring_n` (50.00%)	`substring1` (25.00%)	`substring` (25.00%)	-

Table 4.8: Common variables and uncommon variables, their probabilities of getting correctly predicted, and the probabilities of the top-3 incorrect predictions of VarBERT on VarCorpus-O0 (IDA).

remain unsolved for some time. We discuss in this section the limitations of VarBERT as well as potential future research directions that may lead to the ultimate solution of readable and useful decompilation.

4.9.1 Threats to Validity

Insufficient Decompilation Quality: In the course of conducting this research, we notice that modern binary code decompilers usually fail to generate satisfactory results for C++ binaries, or C binaries that were compiled with optimizations enabled. This is because binary decompilation is its own research area with many unsolved research problems. Variable name prediction can only build upon the correct identification of variables in decompilation output. When decompilers fail to yield sufficiently good results, especially when many human-created variables are not identified, VarBERT (and other solutions) cannot predict variable names for variables that do not exist in decompiled code.

Unrepresentative Corpora: A key assumption underpinning statistical machine learning is that the training set must be of an independent and identical distribution as real-world samples. As we showed in Section 4.3.2, the most obvious difference between VarCorpus, DIRE and DIRT is the distribution of functions: DIRE and DIRT contain much more duplicated functions.

Based on how we created our corpora (using binary executables instead of compiled object files), we believe our distribution is closer to what a human reverse engineer may encounter when decompiling binary programs. However, we built our corpora by collecting C packages in package repositories of two major Linux distributions. The corpora may not be representative for executables on other platforms, such as Windows or MacOS. Additionally, most (if not all) of the variables in the corpus are named in English, which means VarCorpus is not representative for binaries whose variables were originally named in a non-English language.

4.9.2 Future Research

An obvious improvement for variable name prediction is considering the surrounding functions and calling context during training and testing. This will be very helpful for small utility functions that are called at different locations. Additionally, future research may consider incorporating run-time values in variable name prediction, which has been suggested by StateFormer (Pei *et al.*, 2021a).

4.10 Related Work

Binary Decompilation: Decompilation was originally referred to as "reverse compilation" by Cifuentes in a seminal thesis (Cifuentes, 1994). Two critical problems that binary code decompilation must solve are (control-flow) structural analysis and variable type inference. On structural analysis, Phoenix first proposed semantics-preserving structural analysis (Schwartz *et al.*, 2013). The Dream decompiler implemented a goto-free structural recovery algorithm (Yakdan *et al.*, 2015a). Recently, Gussoni et al.

proposed an approach that structures binary-level control flow graphs with zero goto statements (Gussoni *et al.*, 2020). Regarding variable type inference, TIE (Lee *et al.*, 2011), retypd (Noonan *et al.*, 2016), and Osprey (Zhang *et al.*, 2021b) are recent work for inferencing variable types on binary programs. Advances in binary decompilation benefit VarBERT by providing higher-quality decompilation output that resembles human-developed source code.

Neural Models in Binary Decompilation: Recently, researchers have been applying deep learning in decompiling binary code or improving the decompilation result. Katz et al. used recurrent neural networks to decompile binary code snippets into C code (Katz *et al.*, 2018). Coda is a recent end-to-end neural-based approach to decompilation (Fu *et al.*, 2019). While they have achieved promising results, it is still too early to decompile reasonably sized binary programs purely with neural models.

More research projects aim to use neural networks to improve the quality of binary reverse engineering results or decompilation output. Debin is the first attempt in using machine learning to predict debug information for stripped binary code (He *et al.*, 2018). DIRE focuses on predicting variable names in decompilation output (Lacomis *et al.*, 2019). NFRE predicts function names on stripped binaries by learning from a large corpus of stripped binaries (Gao *et al.*, 2021). DIRECT improved upon DIRE by not requiring an AST for the decompilation output and only relying on text tokens, and it was evaluated on DIRE's data set (Nitin *et al.*, 2021). DIRTY advances the field by predicting both variable names and types on decompiled code (Chen *et al.*, 2022). VarBERT directly improves on DIRE by first introducing a transfer-learning-based model, which addresses the fundamental challenge in data set building. VarBERT also proposes a new data set VarCorpus that alleviates many key issues in DIRE and DIRTY's original training and test corpus. Finally, VarBERT expands variable name prediction to multiple decompilers (IDA and Ghidra), while DIRE and DIRTY only

		DIRE	**DIRECT**	**DIRTY**	**VarBERT**
Dataset	New Datasets	✓	✗	✓(x1.65)	✓(x2.07, x4.16)
	# New Datasets	1	✗	1	7
	Optimization level	O0	–	O0	O0, O1, O2
	Split	Binary	–	Binary	Function, Binary
	Duplicates	✓(68.6%)	–	✓(56.9%)	✗
Approach	Pretraining	✗	✓(MLM)	✗	✓(MLM, CMLM)
	Extra inputs	AST	AST	✗	✗
	Origin Prediction	✗	✗	✗	✓
	Inference	Generation	Generation	Generation	Prediction

Table 4.9: Our Contributions Compared To Existing Approaches

support IDA.

4.11 Conclusion

We propose a new solution VarBERT for predicting meaningful variable names in decompilation output. VarBERT is based on transfer learning, which learns knowledge on a large corpus of human-developed source code and fine-tunes for the task of variable name prediction and variable origin prediction on decompilation output from two decompilers (IDA and Ghidra). During our research, we found major issues with the state-of-the-art dataset and built a new data set that is, to the best of our knowledge, free of these issues. We demonstrate that VarBERT outperforms both DIRE and DIRTY on their original data set, *even without pre-training on human source code* and achieves acceptable prediction accuracy on our new corpora VarCorpus. Our contributions as compared to the existing approaches can be seen from the table 4.9. Our research corrects the existing understanding of the research progress on the topic of variable name prediction in decompiled code, establishes new baselines, and, by releasing our research artifacts to the public, will foster new research on this topic.

Chapter 5

CAREFUL SELECTION OF KNOWLEDGE TO SOLVE OPEN BOOK QUESTION ANSWERING

ABSTRACT

Open book question answering is a type of natural language-based QA (NLQA) where questions are expected to be answered with respect to a given set of open book facts, and common knowledge about a topic. Recently a challenge involving such QA, OpenBookQA, has been proposed. Unlike most other NLQA tasks that focus on linguistic understanding, OpenBookQA requires deeper reasoning involving linguistic understanding as well as reasoning with common knowledge. In this paper, we address QA with respect to the OpenBookQA dataset and combine state-of-the-art language models with abductive information retrieval (IR), information gain-based re-ranking, passage selection, and weighted scoring to achieve 72.0% accuracy, an 11.6% improvement over the current state of the art.

5.1 Introduction

Natural language-based question answering (NLQA) not only involves linguistic understanding but often involves reasoning with various kinds of knowledge. In recent years, many NLQA datasets and challenges have been proposed, for example, SQuAD (Rajpurkar *et al.*, 2016), TriviaQA (Joshi *et al.*, 2017) and MultiRC (Khashabi *et al.*, 2018), and each of them have their own focus, sometimes by design and other times by virtue of their development methodology. Many of these datasets and challenges try to mimic human question-answering settings. One such setting is open book question answering where humans are asked to answer questions in a setup where they can refer to books and other materials related to their questions. In such a setting, the focus is not on memorization but, as mentioned in (Mihaylov *et al.*, 2018), on "*deeper understanding of the materials and its application to new situations (Jenkins, 1995; Landsberger, 1996).*" In (Mihaylov *et al.*, 2018), they propose the OpenBookQA dataset mimicking this setting.

Question: *A tool used to identify the percent chance of a trait being passed down has how many squares?* (A) Two squares (B) **Four squares** (C) Six squares (D) Eight squares
Extracted from OpenBook: a punnett square is used to identify the percent chance of a trait being passed down from a parent to its offspring.
Retrieved Missing Knowledge: Two squares is four. The Punnett square is made up of 4 squares and 2 of them are blue and 2 of them are brown, this means you have a 50% chance of having blue or brown eyes.

Table 5.1: An Example of Distracting Retrieved Knowledge

The OpenBookQA dataset has a collection of questions and four answer choices for each question. The dataset comes with 1326 facts representing an open book. It is expected that answering each question requires at least one of these facts. In addition, it requires common knowledge. To obtain relevant common knowledge we use an IR system (Clark *et al.*, 2016) front end to a set of knowledge-rich sentences. Compared to the reading comprehension-based QA (RCQA) setup where the answers to a question are usually found in the given small paragraph, in the OpenBookQA setup the open book part is much larger (than a small paragraph) and is not complete as additional common knowledge may be required. This leads to multiple challenges. First, finding the relevant facts in an open book (which is much bigger than the small paragraphs in the RCQA setting) is a challenge. Then, finding the relevant common knowledge using the IR front end is an even bigger challenge, especially since standard

IR approaches can be misled by distractions. For example, Table 5.1 shows a sample question from the OpenBookQA dataset. We can see the retrieved missing knowledge contains words that overlap with both answer options A and B. Introduction of such knowledge sentences increases confusion for the question answering model. Finally, reasoning involving both facts from open book, and common knowledge leads to multi-hop reasoning with respect to natural language text, which is also a challenge.

We address the first two challenges and make the following contributions in this paper: (a) We improve on knowledge extraction from the OpenBook present in the dataset. We use semantic textual similarity models that are trained with different datasets for this task; (b) We propose natural language abduction to generate queries for retrieving missing knowledge; (c) We show how to use Information Gain based Re-ranking to reduce distractions and remove redundant information; (d) We provide an analysis of the dataset and the limitations of BERT Large model for such a question answering task.

The current best model on the leaderboard of OpenBookQA is the BERT Large model (Devlin *et al.*, 2019). It has an accuracy of 60.4% and does not use external knowledge. Our knowledge selection and retrieval techniques achieve an accuracy of 72%, with a margin of 11.6% on the current state of the art. We study how the accuracy of the BERT Large model varies with varying numbers of knowledge facts extracted from the OpenBook and through IR.

5.2 Related Work

In recent years, several datasets have been proposed for natural language question answering (Rajpurkar *et al.*, 2016; Joshi *et al.*, 2017; Khashabi *et al.*, 2018; Richardson *et al.*, 2013; Lai *et al.*, 2017; Reddy *et al.*, 2018; Choi *et al.*, 2018; Tafjord *et al.*, 2018; Mitra *et al.*, 2019) and many attempts have been made to solve these challenges

(Devlin *et al.*, 2019; Vaswani *et al.*, 2017; Seo *et al.*, 2016).

Among these, the closest to our work is the work in (Devlin *et al.*, 2019) which perform QA using fine-tuned language model, and the works of (Sun *et al.*, 2018; Zhang *et al.*, 2018) which performs QA using external knowledge.

Related to our work for extracting missing knowledge are the works of (Ni *et al.*, 2018; Musa *et al.*, 2018; Khashabi *et al.*, 2017) which respectively generate a query either by extracting key terms from a question and an answer option or by classifying key terms or by Seq2Seq models to generate key terms. In comparison, we generate queries using the question, an answer option, and an extracted fact using natural language abduction.

The task of natural language abduction for natural language understanding has been studied for a long time (Norvig, 1983, 1987; Hobbs, 2004; Hobbs *et al.*, 1993; Wilensky, 1983; Wilensky *et al.*, 2000; Charniak and Goldman, 1988, 1989). However, such works transform the natural language text to a logical form and then use formal reasoning to perform the abduction. On the contrary, our system performs abduction over natural language text without translating the texts to a logical form.

5.3 Approach

Our approach involves six main modules: *Hypothesis Generation*, *OpenBook Knowledge Extraction*, *Abductive Information Retrieval*, *Information Gain based Reranking*, *Passage Selection* and *Question Answering.* A key aspect of our approach is to accurately hunt the needed knowledge facts from the OpenBook knowledge corpus and hunt missing common knowledge using IR. We explain our approach in the example given in Table 5.2.

Question: *A red-tailed hawk is searching for prey. It is most likely to swoop down on what?* (A) **a gecko**
Generated Hypothesis : H : A red-tailed hawk is searching for prey. It is most likely to swoop down on a gecko.
Retrieved Fact from OpenBook: F : hawks eat lizards
Abduced Query to find missing knowledge: K : gecko is lizard
Retrieved Missing Knowledge using IR: K : Every gecko is a lizard.

Table 5.2: Our Approach with an Example for the Correct Option

In *Hypothesis Generation*, our system generates a hypothesis $\mathbf{H_{ij}}$ for the ith question and jth answer option, where $j \in \{1, 2, 3, 4\}$. In *OpenBook Knowledge Extraction*, our system retrieves appropriate knowledge $\mathbf{F_{ij}}$ for a given hypothesis $\mathbf{H_{ij}}$ using semantic textual similarity, from the OpenBook knowledge corpus $\mathbf{F}$. In *Abductive Information Retrieval*, our system abduces missing knowledge from $\mathbf{H_{ij}}$ and $\mathbf{F_{ij}}$. The system formulates queries to perform IR to retrieve missing knowledge $\mathbf{K_{ij}}$. With the retrieved $\mathbf{K_{ij}}$, $\mathbf{F_{ij}}$, *Information Gain based Re-ranking* and *Passage Selection* our system creates a knowledge passage $\mathbf{P_{ij}}$. In *Question Answering*, our system uses $\mathbf{P_{ij}}$ to answer the questions using a BERT Large-based MCQ model, similar to its use in solving SWAG (Zellers *et al.*, 2018).

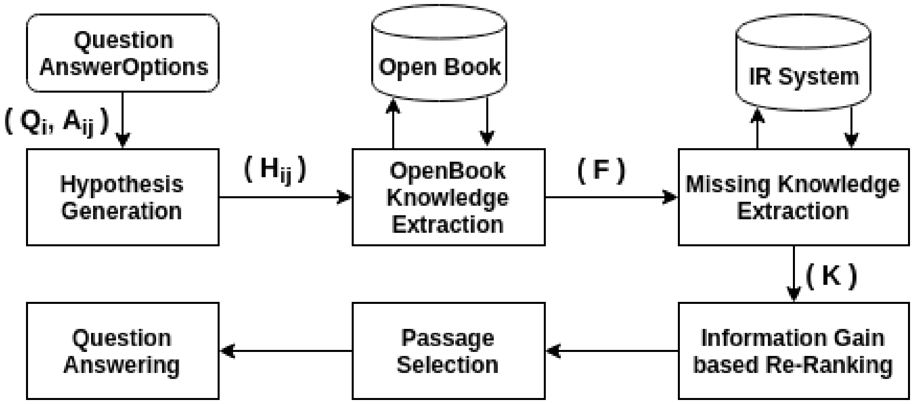


Figure 5.1: Our Approach

5.3.1 Hypothesis Generation

Our system creates a hypothesis for each of the questions and candidate answer options as part of the data preparation phase as shown in the example in Table 5.2. The questions in the OpenBookQA dataset are either with *wh* word or are incomplete statements. To create hypothesis statements for questions with *wh* words, we use the rule-based model of (Demszky *et al.*, 2018). For the rest of the questions, we concatenate the questions with each of the answers to produce the four hypotheses. This has been done for all the training, test, and validation sets.

5.3.2 OpenBook Knowledge Extraction

To retrieve a small set of relevant knowledge facts from the knowledge corpus **F**, a textual similarity model is trained in a supervised fashion on two different datasets and the results are compared. We use the *large-cased* BERT (Devlin *et al.*, 2019) (BERT Large) as the textual similarity model.

BERT Model Trained On STS-B

We train it on the semantic textual similarity (STS-B) data from the GLUE dataset (Wang *et al.*, 2018). The trained model is then used to retrieve the top ten knowledge facts from corpus $\mathbf{F}$ based on the STS-B scores. The STS-B scores range from 0 to 5.0, with 0 being the least similar.

BERT Model Trained On OpenBookQA

We generate the dataset using the gold OpenBookQA facts from $\mathbf{F}$ for the train and validation set provided. To prepare the train set, we first find the similarity of the OpenBook $\mathbf{F}$ facts with respect to each other using the BERT model trained on the STS-B dataset. We assign a score of 5.0 for the gold $\hat{\mathbf{F}}_{\mathbf{i}}$ fact for a hypothesis. We then sample different facts from the OpenBook and assign the STS-B similarity scores between the sampled fact and the gold fact $\hat{\mathbf{F}}_{\mathbf{i}}$ as the target score for that fact $\mathbf{F}_{\mathbf{ij}}$ and $\mathbf{H}_{\mathbf{ij}}$. For example:

Hypothesis : Frilled sharks and angler fish live far beneath the surface of the ocean, which is why they are known as Deep sea animals.

Gold Fact : deep sea animals live deep in the ocean : Score : 5.0

Sampled Facts :

coral lives in the ocean : Score : 3.4

a fish lives in water : Score : 2.8

We do this to ensure a balanced target score is present for each hypothesis and fact. We use this trained model to retrieve the top ten relevant facts for each $\mathbf{H}_{\mathbf{ij}}$ from the knowledge corpus $\mathbf{F}$.

5.3.3 Natural Language Abduction And IR

To search for the missing knowledge, we need to know what we are missing. We use "*abduction*" to figure that out. Abduction is a long-studied task in AI, where normally, both the observation (hypothesis) and the domain knowledge (known fact) are represented in a formal language from which a logical solver abduces possible explanations (missing knowledge). However, in our case, both the observation and the domain knowledge are given as natural language sentences from which we want to find out a possible missing knowledge, which we will then hunt using IR. For example, one of the hypotheses $\mathbf{H_{ij}}$ is "*A red-tailed hawk is searching for prey. It is most likely to swoop down on a gecko.*", and for which the known fact $\mathbf{F_{ij}}$ is "*hawks eats lizards*". From this we expect the output of the natural language abduction system to be $\mathbf{K_{ij}}$ or "*gecko is a lizard*". We will refer to this as "*natural language abduction*".

For natural language abduction, we propose three models, compare them against a baseline model and evaluate each on a downstream question-answering task. All the models ignore stop words except the Seq2Seq model. We describe the three models and a baseline model in the subsequent subsections.

Word Symmetric Difference Model

We design a simple heuristic-based model defined as below:

$$K_{ij} = (H_{ij} \cup F_{ij}) \setminus (H_{ij} \cap F_{ij}) \quad \forall j \in \{1, 2, 3, 4\}$$

where i is the ith question, j is the jth option, H_{ij}, F_{ij}, K_{ij} represents a set of unique words of each instance of hypothesis, facts retrieved from knowledge corpus $\mathbf{F}$ and abduced missing knowledge of validation and test data respectively.

Supervised Bag Of Words Model

In the Supervised Bag of Words model, we select words that satisfy the following condition:

$$P(w_n \in K_{ij}) > \theta$$

where $w_n \in \{H_{ij} \cup F_{ij}\}$. To elaborate, we learn the probability of a given word w_n from the set of words in $H_{ij} \cup F_{ij}$ belonging to the abduced missing knowledge K_{ij}. We select those words which are above the threshold θ.

To learn this probability, we create a training and validation dataset where the words similar (cosine similarity using spaCy) (Honnibal and Montani, 2017) to the words in the gold missing knowledge $\hat{K}_i$ (provided in the dataset) are labeled as positive class and all the other words not present in $\hat{K}_i$ but in $H_{ij} \cup F_{ij}$ are labeled as negative class. Both classes are ensured to be balanced. Finally, we train a binary classifier using BERT Large with one additional feed-forward network for classification. We define the value for the threshold θ using the accuracy of the classifier on the validation set. 0.4 was selected as the threshold.

Copynet Seq2Seq Model

In the final approach, we used the copynet sequence to sequence model (Gu *et al.*, 2016) to generate, instead of predict, the missing knowledge given, the hypothesis **H** and knowledge fact from the corpus **F**. The intuition behind using copynet model is to make use of the copy mechanism to generate essential yet *precise* (minimizing distractors) information which can help in answering the question. We generate the training and validation dataset using the gold $\hat{\mathbf{K}}_{\mathbf{i}}$ as the target sentence, but we replace out-of-vocabulary words from the target with words similar (cosine similarity using spaCy) (Honnibal and Montani, 2017) to the words present in $H_{ij} \cup F_{ij}$. Here, however, we did not remove the stopwords. We choose one, out of multiple generated knowledge based on our model which provided maximum *overlap_score*, given by

$$overlap\_score = \frac{\sum_i count((\hat{H}_i \cup F_i) \cap K_i)}{\sum_i count(\hat{K}_i)}$$

where i is the ith question, $\hat{H}_i$ being the set of unique words of correct hypothesis, F_i being the set of unique words from retrieved facts from knowledge corpus $\mathbf{F}$, K_i being the set of unique words of predicted missing knowledge and $\hat{K}_i$ being the set of unique words of the gold missing knowledge.

Word Union Model

To see if abduction helps, we compare the above models with a Word Union Model. To extract the candidate words for missing knowledge, we used the set of unique words from both the hypothesis and OpenBook knowledge as candidate keywords. The model can be formally represented with the following:

$$K_{ij} = (H_{ij} \cup F_{ij}) \quad \forall j \in \{1, 2, 3, 4\}$$

5.3.4 Information Gain Based Re-ranking

In our experiments we observe that BERT QA model gives a higher score if similar sentences are repeated, leading to the wrong classification. Thus, we introduce Information Gain based Re-ranking to remove redundant information.

We use the same BERT Knowledge Extraction model Trained on OpenBookQA data (section 5.3.2), which is used for extraction of knowledge facts from corpus $\mathbf{F}$ to do an initial ranking of the retrieved missing knowledge $\mathbf{K}$. The scores of this knowledge extraction model are used as relevancy score, rel. To extract the top ten missing knowledge $\mathbf{K}$, we define a redundancy score, red_{ij}, as the maximum cosine similarity, sim, between the previously selected missing knowledge, in the previous iterations till i, and the candidate missing knowledge K_j. If the last selected missing knowledge is K_i, then

$$red_{ij}(K_j) = max(red_{i-1,j}(K_j), sim(K_i, K_j))$$

$$rank\_score = (1 - red_{i,j}(K_j)) * rel(K_j)$$

For missing knowledge selection, we first take the missing knowledge with the highest *rel* score. From the subsequent iteration, we compute the redundancy score with the last selected missing knowledge for each of the candidates and then rank them using the updated *rank_score*. We select the top ten missing knowledge for each $\mathbf{H_{ij}}$.

5.3.5 Question Answering

Once the OpenBook knowledge facts **F** and missing knowledge **K** have been extracted, we move on to the task of answering the questions.

Question-Answering Model

We use BERT Large model for the question-answering task. For each question, we create a passage using the extracted facts and missing knowledge and fine-tune the BERT Large model for the QA task with one additional feed-forward layer for classification. The passages for the training dataset were prepared using the knowledge corpus facts, **F**. We create a passage using the top N facts, similar to the actual gold fact $\mathbf{\hat{F}_i}$, for the train set. The similarities were scored using the STS-B trained model (section 5.3.2). The passages for the training dataset do not use the gold missing knowledge $\mathbf{\hat{K}_i}$ provided in the dataset. For each of our experiments, we use the same trained model, with passages from different IR models.

F	Any Passage			Correct Passage			Accuracy(%)		
N	TF-IDF	Trained	STS-B	TF-IDF	Trained	STS-B	TF-IDF	Trained	STS-B
1	228	258	288	196	229	234	52.6	63.6	59.2
2	294	324	347	264	293	304	57.4	**66.2**	60.6
3	324	358	368	290	328	337	59.2	65.0	60.2
5	350	391	398	319	**370**	366	61.6	65.4	62.8
7	356	411	411	328	**390**	384	59.4	65.2	61.8
10	373	423	420	354	**405**	396	60.4	65.2	59.4

Table 5.3: Compares (a) the Number of Correct Facts That Appears Across Any Four Passages (b) the Number of Correct Facts That Appears in the Passage of the Correct Hypothesis (C) the Accuracy for TF-IDF, Bert Model Trained on STS-B Dataset and Bert Model Trained on Openbook Dataset. N Is the Number of Facts Considered.

The BERT Large model limits passage length to be lesser than or equal to 512. This restricts the size of the passage. To be within the restrictions we create a passage for each of the answer options and score for all answer options against each passage. We refer to this scoring as *sum score*, defined as follows:

For each answer option, A_j, we create a passage P_j and score against each of the answer options A_i. To compute the final score for the answer, we sum up each individual score. If Q is the question, the score for the answer is defined as

$$Pr(Q, A_i) = \sum_{j=1}^{4} score(P_j, Q, A_i)$$

where *score* is the classification score given by the BERT Large model. The final answer is chosen based on,

$$A = \arg\max_{A} Pr(Q, A_i)$$

Passage Selection And Weighted Scoring

In the first round, we score each of the answer options using a passage created from the selected knowledge facts from corpus **F**. For each question, we ignore the passages of the answer options which are in the bottom two. We refer to this as *Passage Selection.* In the second round, we score for only those passages which are selected after adding the missing knowledge $\mathbf{K}$.

We assume that the correct answer has the highest score in each round. Therefore we multiply the scores obtained after both rounds. We refer to this as *Weighted Scoring.* We define the combined passage selected scores and weighted scores as follows :

$$Pr(\mathbf{F}, Q, A_i) = \sum_{j=1}^{4} score(P_j, Q, A_i)$$

where P_j is the passage created from extracted OpenBook knowledge, $\mathbf{F}$. The top two passages were selected based on the scores of $Pr(\mathbf{F}, Q, A_i)$.

$$Pr(\mathbf{F} \cup \mathbf{K}, Q, A_i) = \sum_{k=1}^{4} \delta * score(P_k, Q, A_i)$$

where $\delta = 1$ for the top two scores and $\delta = 0$ for the rest. P_k is the passage created using both the facts and missing knowledge. The final weighted score is :

$$wPr(Q, A_i) = Pr(\mathbf{F}, Q, A_i) * Pr(\mathbf{F} \cup \mathbf{K}, Q, A_i)$$

The answer is chosen based on the top-weighted scores as below:

$$A = \arg\max_{A} wPr(Q, A_i)$$

5.4 Experiments

5.4.1 Dataset And Experimental Setup

The dataset of OpenBookQA contains 4957 questions in the train set and 500 multiple choice questions in validation and test respectively. We train a BERT Large-based QA model using the top ten knowledge facts from the corpus $\mathbf{F}$, as a passage for both the training and validation set. We select the model which gives the best score for the validation set. The same model is used to score the validation and test set with different passages derived from different methods of Abductive IR. In the best Abductive IR model, the number of facts from **F** and **K** are selected from the best validation scores for the QA task.

5.4.2 OpenBook Knowledge Extraction

Question: .. *they decide the best way to save money is ?* (A) to quit eating lunch out (B) to make more phone calls (C) to buy less with monopoly money (D) to have lunch with friends

Knowledge extraction trained with STS-B:

using less resources usually causes money to be saved
a disperser disperses
each season occurs once per year

Knowledge extraction trained with OpenBookQA:

using less resources usually causes money to be saved
decreasing something negative has a positive impact on a thing
conserving resources has a positive impact on the environment

Table 5.3 shows a comparative study of our three approaches for OpenBook knowledge extraction. We show the number of correct OpenBook knowledge extracted for all of the four answer options using the three approaches TF-IDF, BERT model trained on STS-B data and BERT model Trained on OpenBook data. Apart from that, we also show the count of the number of facts presents precisely across the correct answer options. It can be seen that the Precision@N for the BERT model trained on OpenBook data is better than the other models as N increases.

The above example presents the facts retrieved from BERT model trained on OpenBook which are more relevant than the facts retrieved from BERT model trained on STS-B. Both the models were able to find the most relevant fact, but the other facts for the STS-B model introduce more distractors and have lesser relevance. The impact of this is visible from the accuracy scores for the QA task in Table 5.3. The best performance of the BERT QA model can be seen to be 66.2% using only OpenBook facts.

5.4.3 Abductive Information Retrieval

We evaluate the abductive IR techniques at different values for a number of facts from **F** and the number of missing knowledge **K** extracted using IR. Figure 5.2 shows the accuracy against different combinations of **F** and **K**, for all four techniques of IR prior to Information gain-based Re-ranking. In general, we noticed that the trained models performed poorly compared to the baselines. The Word Symmetric Difference model performs better, indicating abductive IR helps. The poor performance of the trained models can be attributed to the challenge of learning abductive inference.

For the above example it can be seen, the pre-reranking facts are relevant to the question but contribute very less considering the knowledge facts retrieved from the corpus **F** and the correct answer. Figure 5.3 shows the impact of Information gain-based Re-ranking. Removal of redundant data allows the scope of more relevant information to be present in the Top N retrieved missing knowledge **K**.

Question: *A red-tailed hawk is searching for prey. It is most likely to swoop down on what?* (A) an eagle (B) a cow (C) **a gecko** (D) a deer

Fact from F : hawks eats lizards

Pre-Reranking K :

red-tail hawk in their search for prey

Red-tailed hawks soar over the prairie and woodlands in search of prey.

Post-Reranking K:

Geckos - only vocal lizards.

Every gecko is a lizard.

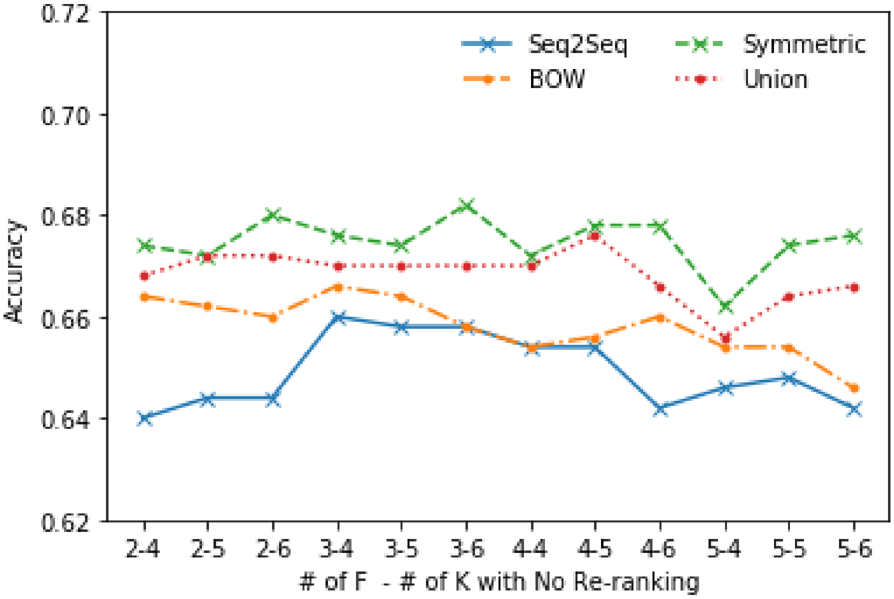


Figure 5.2: Accuracy v/s Number of Facts from F - Number of Facts from K, Without Information Gain Based Re-ranking for 3 Abductive IR Models and Word Union Model (Without Passage Selection and Weighted Scoring).

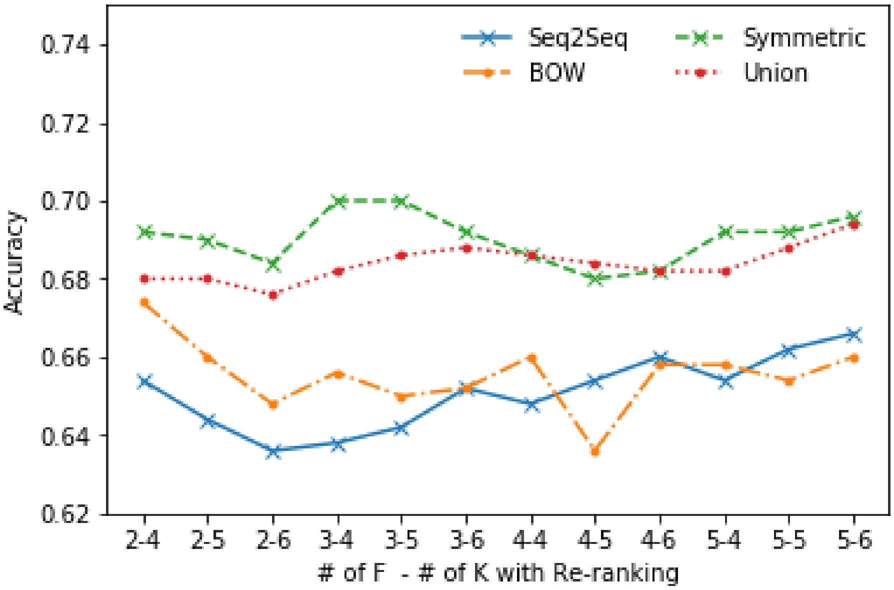


Figure 5.3: Accuracy v/s Number of Facts from F - Number of Facts from K, with Information Gain Based Re-ranking for 3 Abductive IR Models and Word Union Model (Without Passage Selection and Weighted Scoring).

5.4.4 Question Answering

Table 5.4 shows the incremental improvement on the baselines after the inclusion of carefully selected knowledge.

Passage Selection and Weighted Scoring are used to overcome the challenge of boosted prediction scores due to cascading effect of errors in each stage.

Solver	**Accuracy** (%)
Leaderboard	
Guess All ("random")	25.0
Plausible Answer Detector	49.6
Odd-one-out Solver	50.2
Question Match	50.2
Reading Strategies	55.8
Model - BERT-Large (SOTA)	
Only Question (No KB)	60.4
Model - BERT-Large (Our)	
F - TF-IDF	61.6
F - Trained KE	66.2
$\mathbf{F} \cup \mathbf{K}$	70.0
$\mathbf{F} \cup \mathbf{K}$ with Weighted Scoring	70.4
$\mathbf{F} \cup \mathbf{K}$ with Passage Selection	70.8
$\mathbf{F} \cup \mathbf{K}$ with Both	**72.0**
Oracle - BERT-Large	
F gold	74.4
$\mathbf{F} \cup \mathbf{K}$ gold	92.0

Table 5.4: Test Set Comparison of Different Components. The Current State-of-the-art (SOTA) Is the Only Question Model. K Is Retrieved from the Symmetric Difference Model. KE Refers to Knowledge Extraction.

Question: *What eat plants?* (A) leopards (B) eagles (C) owls (D) **robin**

Appropriate extracted Fact from F :

some birds eat plants

Wrong Extracted Fact from F :

a salamander eats insects

Wrong Retrieved Missing Knowledge:

Leopard geckos eat mostly insects

For the example shown above, the wrong answer *leopards* had a very low score with only the facts extracted from knowledge corpus **F**. But the introduction of missing knowledge from the wrong fact from **F** boosts its scores, leading to the wrong prediction. Passage selection helps in the removal of such options and Weighted

Scoring gives preference to those answer options whose scores are relatively high before and after the inclusion of missing knowledge.

5.5 Analysis & Discussion

5.5.1 Model Analysis

BERT Question Answering model: BERT performs well on this task but is prone to distractions. Repetition of information leads to boosted prediction scores. BERT performs well for lookup-based QA, as in RCQA tasks like SQuAD. But this poses a challenge for Open Domain QA, as the extracted knowledge enables lookup for all answer options, leading to an adversarial setting for lookup-based QA. This model is able to find the correct answer, even under the adversarial setting, which is shown by the performance of the *sum score* to select the answer after passage selection.

Symmetric Difference Model This model improves on the baseline Word Union model by 1-2%. The improvement is dwarfed because of inappropriate domain knowledge from **F** being used for abduction. The intersection between the inappropriate domain knowledge and the answer hypothesis is $\varnothing$, which leads to queries that are exactly the same as the Word Union model.

Supervised learned models The supervised learned models for abduction underperform. The Bag of Words and the Seq2Seq models fail to extract keywords for many $\mathbf{F}-\mathbf{H}$ pairs, sometimes missing the keywords from the answers. The Seq2Seq model sometimes extracts the exact missing knowledge, for example, it generates "*some birds is robin*" or "*lizard is gecko*". This shows there is promise in this approach and the poor performance can be attributed to insufficient train data size, which was 4957 only. A fact verification model might improve the accuracy of the supervised learned models. But, for many questions, it fails to extract proper keywords, copying just a part of the question or the knowledge fact.

5.5.2 Error Analysis

Other than errors due to distractions and failed IR, which were around 85% of the total errors, the errors seen are of four broad categories.

Temporal Reasoning: In the example [1] shown below, even though both the options can be considered as night, the fact that 2:00 AM is more suitable for the bats than 6:00 PM makes it difficult to reason. Such issues accounted for 5% of the errors.

Question: *Owls are likely to hunt at?*

(A) 3:00 PM (B) **2:00 AM** (C) 6:00 PM (D) *7:00 AM*

Negation: In the example shown below, a model is needed which handles negations specifically to reject incorrect options. Such issues accounted for 1% of the errors.

Question: *Which of the following is not an input in photosynthesis?*

(A) *sunlight* (B) **oxygen** (C) water (D) carbon dioxide

Conjunctive Reasoning: In the example as shown below, each answer options are partially correct as the word " *bear*" is present. Thus a model has to learn whether all parts of the answer are true or not, i.e. Conjunctive Reasoning. Logically, all answers are correct, as we can see an "or", but option (A) makes more sense. Such issues accounted for 1% of the errors.

Question: *Some berries may be eaten by*

(A) **a bear or person** (B) *a bear or shark* (C) a bear or lion (D) a bear or wolf

Qualitative Reasoning: In the example shown below, each answer option would stop a car but option (D) is more suitable since it will stop the car quicker. Deeper qualitative reasoning is needed to reject incorrect options. Such issues accounted for 8% of the errors.

[1] Predictions are in italics, Correct answers are in Bold.

Question: *Which of these would stop a car quicker?*
(A) a wheel with wet brake pads (B) *a wheel without brake pads* (C) a wheel with worn brake pads (D) **a wheel with dry brake pads**

5.6 Conclusion

In this work, we have pushed the current state of the art for the OpenBookQA task using simple techniques and careful selection of knowledge. We have provided two new ways of performing knowledge extraction over a knowledge base for QA and evaluated three ways to perform abductive inference over natural language. All techniques are shown to improve on the performance of the final task of QA, but there is still a long way to reach human performance.

We analyzed the performance of various components of our QA system. For natural language abduction, the heuristic technique performs better than the supervised techniques. Our analysis also shows the limitations of BERT-based MCQ models, the challenge of learning natural language abductive inference, and the multiple types of reasoning required for an OpenBookQA task. Nevertheless, our overall system improves on the state of the art by 11.6%.

REFERENCES

Charniak, E. and R. Goldman, "A logic for semantic interpretation", in "Proceedings of the 26th annual meeting on Association for Computational Linguistics", pp. 87–94 (Association for Computational Linguistics, 1988).

Charniak, E. and R. P. Goldman, "A semantics for probabilistic quantifier-free first-order languages, with particular application to story understanding.", in "IJCAI", vol. 89, pp. 1074–1079 (Citeseer, 1989).

Choi, E., H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang and L. Zettlemoyer, "Quac: Question answering in context", arXiv preprint arXiv:1808.07036 (2018).

Clark, P., O. Etzioni, T. Khot, A. Sabharwal, O. Tafjord, P. Turney and D. Khashabi, "Combining retrieval, statistics, and inference to answer elementary science questions", in "Thirtieth AAAI Conference on Artificial Intelligence", (2016).

Demszky, D., K. Guu and P. Liang, "Transforming question answering datasets into natural language inference datasets", arXiv preprint arXiv:1809.02922 (2018).

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "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)", pp. 4171–4186 (Association for Computational Linguistics, Minneapolis, Minnesota, 2019), URL `https://www.aclweb.org/anthology/N19-1423`.

Gu, J., Z. Lu, H. Li and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning", in "Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)", pp. 1631–1640 (Association for Computational Linguistics, 2016), URL `http://aclweb.org/anthology/P16-1154`.

Hobbs, J. R., "Abduction in natural language understanding", Handbook of pragmatics pp. 724–741 (2004).

Hobbs, J. R., M. E. Stickel, D. E. Appelt and P. Martin, "Interpretation as abduction", Artificial intelligence **63**, 1-2, 69–142 (1993).

Honnibal, M. and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing", To appear (2017).

Jenkins, T., *Open Book Assessment in Computing Degree Programmes* (Citeseer, 1995).

Joshi, M., E. Choi, D. S. Weld and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension", arXiv preprint arXiv:1705.03551 (2017).

Khashabi, D., S. Chaturvedi, M. Roth, S. Upadhyay and D. Roth, “Looking beyond the surface: A challenge set for reading comprehension over multiple sentences”, in “Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)”, vol. 1, pp. 252–262 (2018).

Khashabi, D., T. Khot, A. Sabharwal and D. Roth, “Learning what is essential in questions”, in “Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)”, pp. 80–89 (2017).

Lai, G., Q. Xie, H. Liu, Y. Yang and E. Hovy, “Race: Large-scale reading comprehension dataset from examinations”, in “Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing”, pp. 785–794 (Association for Computational Linguistics, 2017), URL `http://aclweb.org/anthology/D17-1082`.

Landsberger, J., “Study guides and strategies.”, URL `Http://www.studygs.net/tsttak7.htm`. (1996).

Mihaylov, T., P. Clark, T. Khot and A. Sabharwal, “Can a suit of armor conduct electricity? a new dataset for open book question answering”, in “EMNLP”, (2018).

Mitra, A., P. Clark, O. Tafjord and C. Baral, “Declarative question answering over knowledge bases containing natural language text with answer set programming”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 33, pp. 3003–3010 (2019).

Musa, R., X. Wang, A. Fokoue, N. Mattei, M. Chang, P. Kapanipathi, B. Makni, K. Talamadupula and M. Witbrock, “Answering science exam questions using query rewriting with background knowledge”, arXiv preprint arXiv:1809.05726 (2018).

Ni, J., C. Zhu, W. Chen and J. McAuley, “Learning to attend on essential terms: An enhanced retriever-reader model for scientific question answering”, arXiv preprint arXiv:1808.09492 (2018).

Norvig, P., “Frame activated inferences in a story understanding program.”, in “IJCAI”, pp. 624–626 (1983).

Norvig, P., “Inference in text understanding.”, in “AAAI”, pp. 561–565 (1987).

Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, “Squad: 100,000+ questions for machine comprehension of text”, in “Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing”, pp. 2383–2392 (2016).

Reddy, S., D. Chen and C. D. Manning, “Coqa: A conversational question answering challenge”, arXiv preprint arXiv:1808.07042 (2018).

Richardson, M., C. J. Burges and E. Renshaw, “Mctest: A challenge dataset for the open-domain machine comprehension of text”, in “Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing”, pp. 193–203 (2013).

Seo, M., A. Kembhavi, A. Farhadi and H. Hajishirzi, “Bidirectional attention flow for machine comprehension”, arXiv preprint arXiv:1611.01603 (2016).

Sun, K., D. Yu, D. Yu and C. Cardie, “Improving machine reading comprehension with general reading strategies”, CoRR **abs/1810.13441** (2018).

Tafjord, O., P. Clark, M. Gardner, W.-t. Yih and A. Sabharwal, “Quarel: A dataset and models for answering questions about qualitative relationships”, arXiv preprint arXiv:1811.08048 (2018).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, “Attention is all you need”, in “Advances in neural information processing systems”, pp. 5998–6008 (2017).

Wang, A., A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding”, arXiv preprint arXiv:1804.07461 (2018).

Wilensky, R., “Planning and understanding: A computational approach to human reasoning”, (1983).

Wilensky, R., D. N. Chin, M. Luria, J. Martin, J. Mayfield and D. Wu, “The berkeley unix consultant project”, in “Intelligent Help Systems for UNIX”, pp. 49–94 (Springer, 2000).

Zellers, R., Y. Bisk, R. Schwartz and Y. Choi, “Swag: A large-scale adversarial dataset for grounded commonsense inference”, arXiv preprint arXiv:1808.05326 (2018).

Zhang, Y., H. Dai, K. Toraman and L. Song, “Kgˆ 2: Learning to reason science exam questions with contextual knowledge graph embeddings”, arXiv preprint arXiv:1805.12393 (2018).

Chapter 6

COMMONSENSE REASONING WITH IMPLICIT KNOWLEDGE IN NATURAL LANGUAGE

ABSTRACT

Commonsense Reasoning is a research challenge studied from the early days of AI. In recent years, several natural language QA tasks have been proposed where commonsense reasoning is important. Two common approaches to this are (i) the Use of well-structured commonsense present in knowledge graphs, and (ii) the Use of progressively larger transformer language models. While acquiring and representing commonsense in a formal representation is challenging in approach (i), approach (ii) gets more and more resource-intensive. In this work, we take a middle ground where we use smaller language models together with a relatively smaller but targeted natural language text corpora. The advantages of such an approach are that it is less resource intensive and yet at the same time it can use unstructured text corpora. We define different unstructured commonsense knowledge sources, explore three strategies for knowledge incorporation, and propose four methods competitive to state-of-the-art methods to reason with implicit commonsense.

6.1 Introduction

For an AI agent to reason about the everyday routine human activities, the agent needs to possess commonsense. Consequently, commonsense acquisition and reasoning are considered critical research challenges from the early days of AI (McCarthy, 1959). The need for commonsense reasoning is reemphasized recently (Sap *et al.*, 2019a; Marcus and Davis, 2019), particularly in NL understanding and QA. Several commonsense reasoning tasks have been proposed that study the different aspects of commonsense reasoning, such as abductive commonsense (Bhagavatula *et al.*, 2019), physical commonsense (Bisk *et al.*, 2019), and social commonsense (Sap *et al.*, 2019b). QA systems approach solving tasks using large-pretrained transformers, such as BERT (Devlin *et al.*, 2019), or use complex knowledge fusion methods to perform QA (Lin *et al.*, 2019; Lv *et al.*, 2020).

In this paper, focusing on low resource use, we evaluate the use of smaller transformer language models and a small number of knowledge-rich natural language sentences, where relevant knowledge may be implicitly expressed. To understand what we mean by implicit knowledge, consider an example from (Winograd, 1972): Given the context "*The city councilmen refused the demonstrators a permit because they feared violence.*", and the question "*Who is fearing violence?*", the correct answer is "*The city councilmen*". An unstructured retrieved (through a web search engine) knowledge (Prakash *et al.*, 2019) for this context-question pair that can help answer this question correctly is: "*He also refused to give his full name because he feared for his safety.*". We can use this knowledge to reason that the person who is refusing, is the one who is fearing. From this example, we can observe that the necessary commonsense knowledge to reason may be present in text in many cases but in an implicit way. Moreover, this knowledge is unstructured, and hence current state-of-the-art knowledge fusion methods are unable to utilize this knowledge without a method to represent it in a knowledge graph triple, as present in *ConceptNet*.

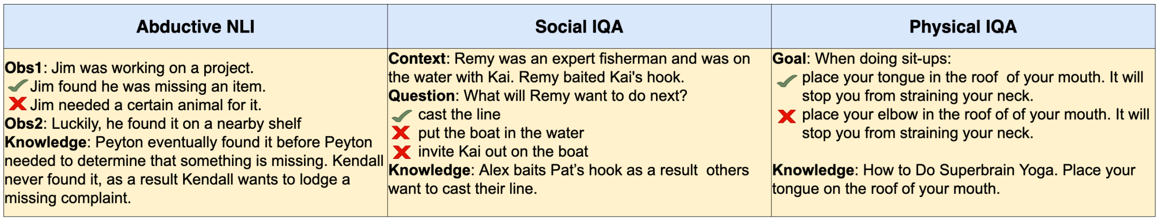
Using natural language sentences (as a source of knowledge) at first glance appears similar to the application of evidence retrieval for open-domain question answering (Yang *et al.*, 2018; Clark *et al.*, 2018; Kwiatkowski *et al.*, 2019), where systems retrieve supporting evidence to be able to answer an open-ended question. However, there is a big difference as, unlike in evidence retrieval, the needed commonsense knowledge may not be *explicitly* available in unstructured knowledge corpora. Our approach is to reason-with-example, in contrast to reading comprehension with retrieved supporting paragraphs containing answers or explicit knowledge that leads to answers. Moreover, a high lexical overlap with retrieved knowledge and context-question-answer does not mean it can be used to answer correctly. For example, another retrieved knowledge for the above question is: "*Demonstrators fear the retaliatory police violence.*". An additional layer of complexity to commonsense reasoning with natural language is added because of such high lexical overlap but distracting sentences.

We limit our study to two pre-trained transformers, namely BERT and RoBERTa. BERT and RoBERTa have been trained using 13GB and 160GB data, respectively. RoBERTa has the same architecture and parameter count but is trained with extensive hyper-parameter tuning and has a larger vocabulary (25K v/s 50K). These allow us to study the implicit commonsense reasoning ability with varying pre-training and vocabulary size. Larger pre-trained transformers have been effectively shown to improve performance on downstream tasks, but training such models is resource-intensive. Hence we ask the following auxiliary question: To what extent can we improve a smaller transformer encoder's performance? Smaller in the sense of pre-training data, vocabulary size, and parameter tuning space.

For addressing the above questions, we propose the following experimental framework. We categorize different unstructured knowledge sources and define a knowledge source preparation and retrieval component. We then propose three strategies for unstructured knowledge infusion. In the *Revision strategy*, we fine-tune the transformer on an unstructured knowledge source. In *Openbook strategy*, we choose a certain number of knowledge statements from the unstructured knowledge source that are textually similar to each of the dataset samples. Then we fine-tune the pre-trained transformer for the question-answering task. In the final strategy, we combine both the strategies mentioned above. We propose three strong baseline methods that utilize knowledge, *concat*, *max*, *simple-sum*, and an explainable reasoning model *weighted-sum* to combine and reason with multiple commonsense knowledge sentences. We evaluate our proposed framework on three public commonsense question-answering datasets: AbductiveNLI (aNLI) (Bhagavatula *et al.*, 2019), PIQA (Bisk *et al.*, 2019) and Social Interaction QA (SIQA) (Sap *et al.*, 2019b).

Our key findings are as follows: (a) Transformers can reason with implicit commonsense knowledge to some extent. We observe that transformers fail to answer questions through detailed error analysis even when sufficient knowledge is present with minimal distractors 30-50% of the time. This observation shows the scope of future improvements. (b) Revision and OpenBook Strategy improve commonsense reasoning performance, but the Revision strategy's impact depends on how well-formed the unstructured knowledge corpus is. (c) Our knowledge retrieval and knowledge infusion methods improve accuracy over pre-trained transformers by 2-9%. They are significantly effective over the smaller transformer encoders and approach larger pre-trained transformers, surpassing T5-11B (Raffel *et al.*, 2019) by 4.14% in aNLI and reducing the gap to 1.75% in SIQA using RoBERTa. These methods should act as future baselines.

In summary, our contributions are: (a) a thorough analysis of transformers'



Abductive NLI	Social IQA	Physical IQA
Obs1: Jim was working on a project. ✓ Jim found he was missing an item. ✗ Jim needed a certain animal for it. **Obs2**: Luckily, he found it on a nearby shelf **Knowledge**: Peyton eventually found it before Peyton needed to determine that something is missing. Kendall never found it, as a result Kendall wants to lodge a missing complaint.	**Context**: Remy was an expert fisherman and was on the water with Kai. Remy baited Kai's hook. **Question**: What will Remy want to do next? ✓ cast the line ✗ put the boat in the water ✗ invite Kai out on the boat **Knowledge**: Alex baits Pat's hook as a result others want to cast their line.	**Goal**: When doing sit-ups: ✓ place your tongue in the roof of your mouth. It will stop you from straining your neck. ✗ place your elbow in the roof of of your mouth. It will stop you from straining your neck. **Knowledge**: How to Do Superbrain Yoga. Place your tongue on the roof of your mouth.

Figure 6.1: Example of All Three Datasets along with Retrieved Knowledge.

ability to perform commonsense reasoning with implicit knowledge on three different commonsense QA tasks using two transformer models. (b) four models representing four ways knowledge can be infused in transformer encoders. These methods apply to multiple commonsense reasoning tasks and improve performance over pre-trained transformers by 2-9% in accuracy. (c) a detailed study to bridge the gap between smaller and larger pre-trained transformers. (d) an extensive investigation to study the impact of different knowledge sources and pre-training on such knowledge sources on commonsense QA tasks.

6.2 MCQ Datasets

To study the extent of transformers' commonsense reasoning ability, we choose the following three datasets to evaluate our models, each with a different kind of commonsense knowledge. Figure 6.1 shows examples from each of the datasets with our retrieved commonsense knowledge sentences.

Abductive NLI (aNLI): This dataset (Bhagavatula *et al.*, 2019) is intended to judge the potential of an AI system to do abductive reasoning to form possible explanations for a given set of observations. The task is to find which of the hypothesis options H_1, and H_2 explains O_2 where O_1 should precede and O_2 should succeed the hypothesis, given a pair of observations O_1 and O_2. This task needs a commonsense understanding of which order sequence of events occurs. There are 169,654 training and 1,532 validation samples. The test set is blind. It has a generation task, but we restrict ourselves to the multiple-choice task.

PIQA (Physical Interaction QA): This dataset is created to evaluate an AI system's physics reasoning capability. The dataset requires reasoning about physical objects and how we use them in our daily lives. The task is to predict the most appropriate choice for the goal G, given a goal G and a pair of choices C_1 and C_2. There are 16,113 training and 1,838 validation samples. The test set is blind.

SIQA: This dataset is a collection of instances of social interaction reasoning and the social implications of their statements. The task is to choose the correct answer option AO_i out of three choices when given a context C of a social situation and a question Q about the situation. There are several question types in this dataset derived from *Atomic* inference dimensions (Sap *et al.*, 2019a). A few of the Atomic inference dimensions are actor *intention*, actor *motivation*, *effect* on the actor,*effect* on others, etc. In total, there are 33,410 training and 1,954 validation samples. The test set is blind.

6.3 Commonsense Knowledge Sources

6.3.1 Knowledge Categorization for Evaluation

Directly Derived: Here the commonsense QA task is directly derived from the knowledge source, and hence using the same knowledge may make the task trivial. We test this scenario on the aNLI task with the following knowledge sources, *ROCStories Corpus* (Mostafazadeh *et al.*, 2016) and *Story Cloze Test*, that were used in creating aNLI. Our motivation is to see how well the model can answer questions when given the "same" or similar implicit/explicit commonsense knowledge.

Partially Derived: Here the commonsense QA task is not directly derived from an external knowledge source, and considerable human knowledge was used to generate the question-answers. In this case, we use SIQA as the task, which uses the *Atomic* (Sap *et al.*, 2019a) knowledge base as the source for social events, but has undergone sufficient human intervention to make the task non-trivial. During dataset creation, the human annotators were asked to turn *Atomic* events into sentences and were asked to create question-answers.

Relevant: Here, the commonsense task is entirely created with human annotators' help without using a specific knowledge source. However, we guess knowledge sources that seem relevant through our QA pairs analysis. We evaluate this using PIQA as the commonsense task and *WikiHow* dataset (Koupaee and Wang, 2018) as the "relevant" external knowledge source.

6.3.2 Knowledge Source Preparation

aNLI: To test our first category of external knowledge, we use the entire *Story Cloze Test* and *ROCStories Corpus*. We also prepare another source that contains knowledge sentences retrieved for the train set of aNLI from the first source. This knowledge source is created to ensure the task is not trivialized by knowledge leakage. We also create a knowledge source from multiple datasets such as *MCTest* (Richardson *et al.*, 2013), *COPA* (Roemmele *et al.*, 2011) and *Atomic*, but not *Story Cloze Test* and *ROCStories Corpus*. These sources contain commonsense knowledge, which might be useful for the aNLI task.

SIQA: We synthetically generate a knowledge source from the events and inference dimensions provided by the *Atomic* dataset (Sap *et al.*, 2019a). The *Atomic* dataset contains events and eight types of if-then inferences. The total number of events is 732,723. Some events are masked, which we fill by using a BERT and masked language modeling (Devlin *et al.*, 2019). We extend the knowledge source and replace *PersonX*

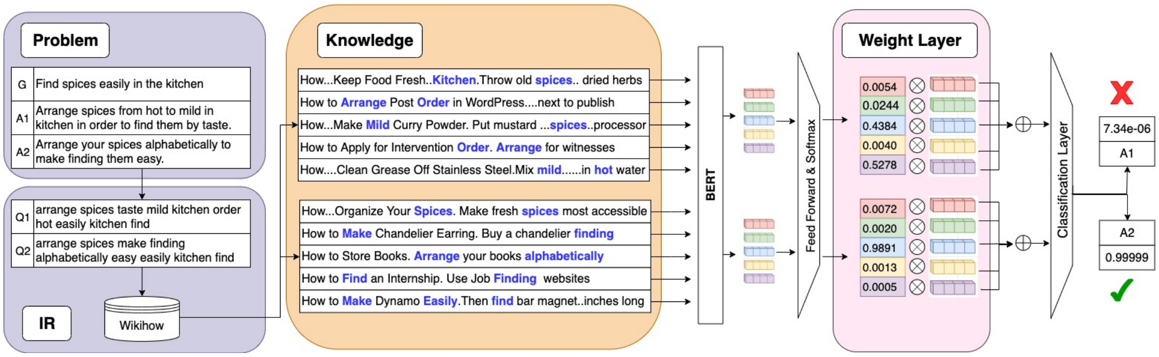

Problem
G
Find spices easily in the kitchen
A1
Arrange spices from hot to mild in kitchen in order to find them by taste.
A2
Arrange your spices alphabetically to make finding them easy.
Q1
arrange spices taste mild kitchen order hot easily kitchen find
Q2
arrange spices make finding alphabetically easy easily kitchen find
IR
Wikihow
Knowledge
How...Keep Food Fresh..Kitchen.Throw old spices.. dried herbs
How to Arrange Post Order in WordPress....next to publish
How...Make Mild Curry Powder. Put mustard ...spices..processor
How to Apply for Intervention Order. Arrange for witnesses
How....Clean Grease Off Stainless Steel.Mix mild......in hot water
How...Organize Your Spices. Make fresh spices most accessible
How to Make Chandelier Earring. Buy a chandelier finding
How to Store Books. Arrange your books alphabetically
How to Find an Internship. Use Job Finding websites
How to Make Dynamo Easily.Then find bar magnet..inches long
BERT
Feed Forward & Softmax
Weight Layer
0.0054
0.0244
0.4384
0.0040
0.5278
0.0072
0.0020
0.9891
0.0013
0.0005
Classification Layer
7.34e-06
A1
A2
0.99999

Figure 6.2: An End-to-end View of Our Approach. From Query Generation, Knowledge Retrieval, the Different Types of Knowledge Retrieved along with Keywords Highlighted in Blue, the Corresponding Learned Weights in the Weighted-sum Model, and Finally to Predicted Logits.

and *PersonY*, as present in the original *Atomic* dataset, using gender-neutral names. These steps may approximate the steps taken by humans to generate QA pairs.

PIQA: We use the *Wikihow* dataset for PIQA. It contains paragraphs (214,544) with detailed steps or actions to complete a task. We extract the title of each paragraph and split the paragraphs into sentences. The title is concatenated to each of the sentences. This preprocessing ensures that the task's goal is present in each of the sentences.

A Combined Commonsense Corpus is created which combines the partially related and relevant corpora, for example, combining *Wikihow*, *Atomic*, *MCTest*.

6.3.3 Knowledge Retrieval

Query Generation: We concatenate the question, answer option, and the context if present, and remove standard English stopwords for query generation. We use common nouns, verbs, adjectives, and adverbs from the QA pairs. Explicit bias towards specific names (John, Jane) is avoided.

Information Retrieval System: We use Elasticsearch to index all knowledge base sentences. We retrieve the top 50 sentences for each QA pair with the default BM-25 ranking model (Robertson and Walker, 1994). The retrieved sentences may contain the key search words in any order.

Re-Ranking: We re-rank the retrieved knowledge sentences to remove redundant sentences containing the same information. We use sentence similarity and knowledge redundancy to perform the iterative re-ranking. We use Spacy, to compute the cosine similarity between sentence Glove vector (Pennington *et al.*, 2014) representations; for knowledge redundancy, we find similarity with the already selected sentences and discard a new sentence if it is > 0.9 similar to higher-ranked sentences. After re-ranking, we select the **top ten** sentences.

We keep our Information Retrieval system generic as the tasks require varying kinds of commonsense knowledge; for example, If-then rules in SIQA, Scripts or Stories in aNLI, and an understanding of Processes and Tools in PIQA.

6.4 Method

After extracting relevant knowledge from the respective KBs, we move onto the task of Question-Answering. We perform our experiments on BERT encoders, with 340M and 355M parameters respectively, BERT-Large (Low vocab-size 25K and pretraining data 13GB) BERT (Devlin *et al.*, 2019) and RoBERTa (high-vocab size 50K and pretraining data 160 GB) RoBERTa (Liu *et al.*, 2019).

QA-Model: As a baseline, we use these pre-trained transformers for the question-answering task with an extra feed-forward layer for classification as a fine-tuning step.

		BERT				RoBERTa			
Dataset	**Strategy**	**Concat**	**Max**	**Sim-Sum**	**Wtd-Sum**	**Concat**	**Max**	**Sim-Sum**	**Wtd-Sum**
aNLI	OPENBOOK	73.9± 0.8	73.7± 0.1	73.5± 0.7	73.3± 1.0	83.9± 0.5	80.8± 0.9	81.7± 0.6	84.4± 0.4
	REVISION	72.7± 0.3	N/A	N/A	N/A	82.4	N/A	N/A	N/A
	REVISION & OPENBOOK	74.4± 0.2	74.3± 0.1	74.0± 0.9	75.1±0.4	84.2± 0.7	81.4± 0.8	82.6± 0.6	**86.7**± 0.6
PIQA	OPENBOOK	67.8± 0.4	72.4± 0.6	72.6± 1.2	72.5± 0.1	74.8± 0.5	75.2± 0.9	75.6± 0.7	77.1± 0.2
	REVISION	74.5± 0.3	N/A	N/A	N/A	75.2± 0.8	N/A	N/A	N/A
	REVISION & OPENBOOK	67.7± 0.1	73.8± 0.8	76.8± 0.5	76.8± 0.3	75.4± 0.7	76.2± 0.8	76.8± 0.4	**80.2**± 0.6
SIQA	OPENBOOK	70.1± 0.8	67.8± 0.1	70.0± 0.7	70.2± 0.4	76.5± 0.7	77.2± 0.6	77.4± 0.2	78.3± 0.5
	REVISION	69.5± 0.9	N/A	N/A	N/A	76.8± 0.3	N/A	N/A	N/A
	REVISION & OPENBOOK	68.8± 0.4	66.6± 0.4	68.9± 0.1	69.3± 0.6	78.2± 0.3	77.4± 0.9	76.7± 0.5	**79.5**± 0.9

Table 6.1: Validation Set Accuracy (%) of Each of the Four Models (Concat, Max, Simple Sum, Weighted Sum). Revision Only Method Has No Retrieved Passage, so Only Q-A Is Concatenated.

6.4.1 Modes of Knowledge Infusion

We experiment with four different models of using knowledge with the transformer architecture for the open-book strategy. The first three, *concat*, *max*, and *simple-sum* act as stronger baselines that use the same implicit knowledge as our proposed *weighted-sum* model. Each of these modules takes as input a problem instance which contains a question Q, n answer choices $a_1, ..., a_n$, and a list called *premises* of length n, one for each answer. Each element in *premises* contains m number of knowledge passages, which might be useful while answering the question Q. Let K_{ij} denote the j th knowledge passage for the i th answer option. Each model computes a score of $score(i)$ for each of the n answer choices. The final answer is the answer choice that receives the maximum score. We now describe how the different models compute the scores differently.

Concat: In this model, all the m knowledge passages for the i-th choice are joined together to make a single knowledge passage K_i. The sequence of tokens {[CLS] K_i [S] Qa_i [S]} is then passed to BERT to pool the [CLS] embedding ($z^{[CLS]}$) from the last layer. This way we get n $z^{[CLS]}$ for n answer choices, each of which is projected to a real number (`score`(i)) using a linear layer.

Parallel-Max: For each answer choice a_i, Parallel-Max uses each of the knowledge passage K_{ij} to create the sequence {[CLS] K_{ij} [S] Qa_i [S]} which is then passed to the BERT model to obtain the $z^{[CLS]}$ from the last layer that is then projected to a real number using a linear layer. `score`(i) is the max of the m scores obtained using each of the m knowledge passage.

Simple Sum: In *simple sum* and the next model assumes that the information is scattered over multiple knowledge passages and tries to aggregate that scattered information. To do this, the *simple sum* model, for each answer choice a_i and each of the knowledge passage K_{ij} creates the sequence {[CLS] K_{ij} [S] Qa_i [S]} which it then passes to the BERT model to obtain the $z^{[CLS]}$ from the last layer. All of these m vectors are then summed to find the summary vector, projected to a scalar using a linear layer to obtain the `score`(i).

Weighted Sum: The *weighted sum* model computes a weighted sum of the m $z^{[CLS]}$ as some of the knowledge passages might be more useful than others. It computes the $z^{[CLS]}$ in a similar way to that of the *simple sum* model. It computes a scalar weight w_{ij} for each of the m $z^{[CLS]}$ using a linear projection layer which we will call as the *weight layer*. The weights are then normalized through a softmax layer and used to compute the weighted sum of the $z^{[CLS]}$. It then uses (1) a linear layer or (2) reuses the weight layer (*tied version*) to compute the final score `score`(i) for the option a_i. We experiment with both options.

Formally, given m $z^{[CLS]}$, we learn two projections w_1 and w_2, such that:

$$score(i) = w_2\Big(\sum_{j=1}^{n} w_1(z^{[CLS]}) * z^{[CLS]}\Big) \tag{6.1}$$

This weighted sum of vectors is similar to the attention weights learned to create contextual word vectors (Vaswani *et al.*, 2017) but we extend it to multiple sentences.

Models/ Accuracy	**aNLI**		**PIQA**		**SIQA**	
	Val	**Test**	**Val**	**Test**	**Val**	**Test**
BERT	67.36	66.75	68.08	69.23	64.88	64.50
GPT-2 XL	N/A	N/A	70.20	69.50	47.50	45.30
RoBERTa	85.05	83.91	76.28	76.80	77.85	76.74
RoBERTa 5 Ensemble	N/A	83.22	N/A	79.66	N/A	78.68
$L2R^2$ (Zhu *et al.*, 2020)	N/A	**86.81**	N/A	N/A	N/A	N/A
KagNet (Lin *et al.*, 2019)	N/A	N/A	N/A	N/A	65.05	64.59
GBR (Lv *et al.*, 2020)	N/A	N/A	N/A	N/A	75.64	76.25
UnifiedQA T5 11B (Khashabi *et al.*, 2020)	N/A	80.04	N/A	**89.50**	N/A	**79.75**
Ours: BERT + WS	74.60	74.96	76.82	72.28	70.21	67.22
Ours: RoBERTa + WS	85.90	84.18	80.20	78.24	79.53	78.00

Table 6.2: Performance of the Weighted-sum Model with *Revision* & *Openbook* Strategy, Compared to Current Best Methods. Underlined Are Methods That We Beat Statistically Significantly. Partially Derived and Related Sources Are Used. Unavailable→N/A. Best→bold.

We minimize the cross-entropy loss between the score and the ground-truth answer. We observe a single-layer network achieves the best accuracy compared to multi-layer feed-forward networks and highway networks for projection.

6.5 Experiments

Let D be an MCQ dataset, T be a pre-trained transformer, K_D be a knowledge source (a set of paragraphs or sentences) that is useful for D and let K be a general knowledge source where T was pre-trained, and K might or might not contain K_D. We consider three approaches to infusing knowledge.

Revision: In this strategy, T is fine-tuned on K_D using Masked LM (both BERT and RoBERTa) and the next sentence prediction task (BERT) and then fine-tuned on the dataset D for the QA task.

Openbook: Here a subset of K_D is assigned to each of the training samples in the dataset D as a knowledge passage context, and the model T is fine-tuned on the

Model	Knowledge Source	aNLI	PIQA	SIQA
BERT	Directly/Partially Derived	75.1± 0.4	N/A	70.2± 0.4
	TrainOnly Directly/Partially	74.6± 0.8	N/A	69.8± 0.7
	Related Knowledge	73.2± 0.5	76.8± 0.3	68.6± 0.5
RoBERTa	Directly/Partially Derived	86.7± 0.6	N/A	79.5± 0.9
	TrainOnly Directly/Partially	85.9± 0.8	N/A	78.9± 1.2
	Related Knowledge	85.0± 1.1	80.2± 0.6	77.4± 0.8

Table 6.3: Effect of Different Knowledge Source Types on the Weighted-sum Knowledge Infused Model. Related Knowledge Source Is the Combination of All Relevant Knowledge Sources, Referred to as the Combined Commonsense Corpus. Metric Is Accuracy.

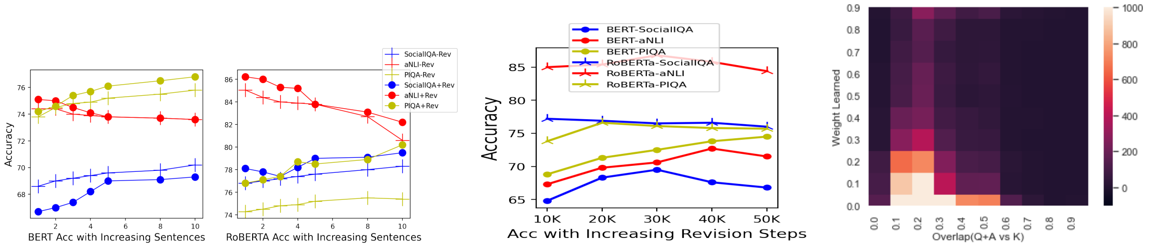
modified dataset D.

Revision with an Openbook: In this strategy, T is fine-tuned on K_D using Masked LM (both BERT and RoBERTa), and the next sentence prediction task (BERT) and also a subset of K_D is assigned to each of the training samples on D. The model is then fine-tuned for the modified dataset D.

We train the models on 4 Nvidia V100 16GB GPUs with learning rates in the range [1e-6,5e-5] and batch sizes of [16,32,48,64]. We report the mean accuracy for three random seed runs. We perform five hyper-parameter trials and param-selection on the validation set.

6.6 Results and Discussion

Tables 6.1, 6.2, and 6.3 summarize our results on three datasets. BERT and RoBERTa baseline validation and hidden test scores are present in Table 6.2. Adding knowledge in natural language form improves QA accuracy statistically significantly across all datasets over the baseline BERT with $p \leq 0.05$ based on Wilson score intervals (Wilson, 1927). This includes retrieving knowledge from related knowledge



(a) ↑ Knowledge Sentences.

(b) ↑ Revision Steps.

(c) Wts. v/s Lex. Overlap

Figure 6.3: For **a**, **b**, and **c** the Knowledge Infusion Model Is Weighted-Sum with Knowledge Retrieved from a Relevant Knowledge Source. In Fig. **a**, We Observe the Effect of an Increasing Number of Implicit Knowledge Sentences. In Fig. **b** We Observe the Effect of Increasing the Number of *Revision* Pre-training Steps. Fig. **c** Shows the Weights Learned Vs. Normalized Lexical Overlap Between Knowledge and Concatenated QA Pair for All Samples of the PIQA Dev Set.

sources, seen in Tables 6.2 and 6.3. The *concat* mode of knowledge infusion improves over the baseline BERT by 1-6%, and the Weighted-Sum model further improves it by 2-4%. In Table 6.2 we can observe the Weighted-Sum model is 4.1% better than T5 in aNLI and reduces the gap to 1.75% in SIQA with 30 times less number of parameters (11B v/s 355M). It also surpasses complex graph-based approaches like GBR and KagNet (Lin *et al.*, 2019; Lv *et al.*, 2020). Other prior work use directly derived knowledge sources and model for specific tasks as in L2R (Zhu *et al.*, 2020). Moreover, UnifiedQA T5 11B (Khashabi *et al.*, 2020) is trained on many datasets, whereas we train only on the provided train dataset, making our approach more sample efficient. This observation validates our hypothesis of using implicit knowledge expressed in natural language to bridge the gap to super-large transformers. Our generic framework improves on all three datasets with models trained only using the provided training dataset.

Strategy	Training Src.	aNLI	SIQA	PIQA
OpenBook	**aNLI**	N/A	63.2 65.5	51.2 57.8
	SIQA	72.4 84.1	N/A	48.5 54.3
	PIQA	62.5 74.2	49.6 54.2	N/A
Revision	**aNLI**	N/A	65.3 66.2	56.2 65.8
	SIQA	70.9 83.8	N/A	52.4 57.8
	PIQA	66.1 78.0	57.4 67.6	N/A
OpenBook + Revision	**aNLI**	N/A	65.8 68.2	55.4 62.8
	SIQA	73.1 85.2	N/A	53.2 59.4
	PIQA	63.8 75.6	52.8 63.1	N/A

Table 6.4: Effect of Cross-dataset Knowledge Source Accuracy on Weighted-Sum (When a Relevant Source for a Different Task Is Used). BERT Left, RoBERTa Right.

Effect of different strategies: Both the *Openbook* and the *Revision* strategies perform well. Together the performance improves even further. The performance of the *Revision* strategy is low for SIQA. The drop in performance may be attributed to the sentences' synthetic nature and the unavailability of next-sentence prediction task data, as the knowledge in the KB for SIQA is single sentences and not paragraphs. PIQA and aNLI results are better due to natural and contiguous sentences. For PIQA, the BERT model improves with knowledge, whereas the RoBERTa model underperforms, indicating RoBERTa gets distracted by the retrieved knowledge, and the pre-training knowledge is more useful. BERT with implicit knowledge approaches RoBERTa without knowledge, with the gap reduced by 4% on average. Similarly, RoBERTa approaches T5 with *Revision* & *Openbook* strategy.

Effect of different knowledge sources: Table 6.3 shows the impact of different knowledge sources on the downstream question-answering task. Even a knowledge source with somewhat related knowledge is impactful for the question-answering task, as seen in the case of Related Knowledge and TrainOnly Partially Derived for aNLI and SIQA. In Directly and Partially derived knowledge categories, such as RoCStories for aNLI and *Atomic* for SIQA, the model accuracy with knowledge is significantly

Knowledge	aNLI	SIQA	PIQA
Explicitly Present	14%	11%	10%
Implicitly Present	55%	59%	51%
Fully Irrelevant	31%	30%	39%

Types of Error	aNLI	SIQA	PIQA
Annotation	41%	38%	10%
Model Prediction	48%	27%	29%
Distracting Knowledge	11%	35%	61%

Table 6.5: Left: Percent of Correct Predictions Where the Implicit Knowledge Is Categorized as above, for the RoBERTa Weighted-sum Model. Right: Different Types of Errors Were Observed in the QA Pairs Where the RoBERTa Weighted-Sum Model Failed to Answer Correctly.

more than the baseline but does not reach near-human accuracy. However, the model can still not answer all questions because the model fails to reason well even with sufficient knowledge, and the annotators have modified the information present in the source knowledge significantly. As a result, the knowledge does not overlap with the gold answer, cause if it did, the model will use lexical overlap as a shortcut and perform better. In Table 6.4, we can observe aNLI and SIQA require similar commonsense knowledge, as training with the relevant knowledge source of aNLI has a non-detrimental effect for SIQA and vice-versa. We also observe PIQA performance decreases if we use a knowledge source of aNLI and PIQA, indicating it introduces a significant amount of distraction such that even the implicit knowledge in pre-trained transformers is ignored.

Comparisons between modes of knowledge fusion: The Weighted-Sum model is observed to be consistent across datasets. The other strong baseline models also improve over the no-knowledge models indicating even simple scoring methods over implicit commonsense knowledge sentences can lead to improvements. The Max, Simple-Sum, and Weighted-Sum models have the additional advantage of being partially explainable by observing the weights associated with the knowledge sentences.

Weighted-Sum outperforms them as it has the flexibility to attend in varying degrees to multiple sentences, in contrast to other models. Figure 2 shows the weight versus overlap between knowledge and QA pair distribution for PIQA. There is an overall low overlap, but the model learns to give high weights regardless of the overlap. It indicates that the model captures the implicit knowledge and not just a simple word overlap. We observe that 61% of such low lexical overlap sentences have sufficient implicit knowledge on manual analysis.

Why the impact of external knowledge is less for RoBERTa? RoBERTa has been pre-trained using a gigantic corpus of 160 GB of text. We assume for these tasks that the model needs additional knowledge to answer, but we hypothesize that the pre-training corpus of RoBERTa might contain the knowledge we are trying to infuse, leading to a reduced impact. This observation calls for further analysis of pre-training corpora to categorize such commonsense knowledge. The significant improvement over BERT (3-14%) shows the ability of these methods to utilize implicit knowledge, which is especially useful for low-resource languages, target domains where we can pre-train using fewer data and use ad-hoc knowledge to solve a target task and have smaller vocab and params. But, there is an assumption that at least sufficient data (~10GB) to train a BERT model is necessary. Future work will explore the size v/s knowledge impact for even smaller language models.

Error Analysis: We analyzed 200 correct predictions and error samples from each of our best models, respectively. In Table 6.5, we can observe for around two-thirds of the correct predictions, we have relevant knowledge present. The model also ignores partial noise by reducing its weight and the entire knowledge passage if needed. In those cases, we hypothesize that the knowledge acquired during the revision phase or the original language model pre-training phase helps answer correctly. We divide the errors into three categories, as seen in Table 6.5. *Annotation Errors* are when more than one answer option is correct, or an incorrect answer option is labeled correctly. The questions for which information is insufficient to select a specific answer option also fall into this category. *Distracting knowledge* is where the retrieved knowledge is noisy and does not have sufficient relevant knowledge. *Model prediction* error is where the relevant knowledge is present, though the knowledge is not wholly exact. However, a human could have reasoned with the provided knowledge.

6.7 Related Work

Commonsense Reasoning: Several attempts were made to inject external knowledge into neural networks to improve commonsense QA in recent years. A knowledge selection algorithm to rank knowledge paths from *ConceptNet* via PMI and frequency-based scoring was proposed by (Bauer *et al.*, 2018). (Wang and Jiang, 2019) improve word representations by integrating common word vectors between document and question-answer options. A commonsense-based pre-training was proposed by (Zhong *et al.*, 2019) to learn direct and indirect *ConceptNet* relations. (Lin *et al.*, 2019) proposed a knowledge-augmented graph-based reasoner and pruning knowledge paths using a function adapted from a graph embedding algorithm. (Lv *et al.*, 2020) is the closest work that utilizes both a structured knowledge base and explicit unstructured plain text as a source to enhance contextual representations. Our Revision strategy is similar to task adaptive pre-training, but we focus on commonsense knowledge infusion, whereas (Gururangan *et al.*, 2020) focuses on textual domain adaptation for text classification.

Transformers Reasoning Abilities: Recently, a few attempts were made to understand the different reasoning abilities of transformer models. (Clark *et al.*, 2020) observe that transformers can reason with explicit conjunctive implication rules and observe a strong performance. (Talmor *et al.*, 2020) study to what extent transformers can reason over explicit symbolic facts while retaining implicit pre-training knowledge. (Richardson and Sabharwal, 2020) study if the transformer QA models know definitions and taxonomic reasonings and propose probing datasets. (Gontier *et al.*, 2020) study the ability to generate proofs given knowledge encoded in natural language. In contrast to the above studies, we study the ability to reason with additional implicit commonsense knowledge.

6.8 Conclusion

In this work, we comprehensively study transformers' ability to reason with implicit knowledge expressed in natural language. We propose an experimental framework with knowledge infusion methods and observe a considerable improvement of 2-9% over strong baselines. We observe our methods, trained with fewer samples and parameters, perform competitively with huge pre-trained language models, and surpass complex graph-based methods (Lin *et al.*, 2019; Lv *et al.*, 2020). Moreover, the approaches we studied are general enough to apply to other knowledge-intensive tasks and languages. Our methods reduce the gap between smaller and large pre-trained transformers. We critically analyze the different components and identify that transformers are still unable to answer 30-50% of the time, even with sufficient knowledge, identifying the need for better methods to perform reasoning with implicit knowledge. We hope our findings will help design models that respond better to instructions (Mishra *et al.*, 2021) containing knowledge expressed in natural language.

REFERENCES

Bauer, L., Y. Wang and M. Bansal, “Commonsense for generative multi-hop question answering tasks”, in “Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing”, pp. 4220–4230 (2018).

Bhagavatula, C., R. L. Bras, C. Malaviya, K. Sakaguchi, A. Holtzman, H. Rashkin, D. Downey, S. W.-t. Yih and Y. Choi, “Abductive commonsense reasoning”, arXiv preprint arXiv:1908.05739 (2019).

Bisk, Y., R. Zellers, R. L. Bras, J. Gao and Y. Choi, “Piqa: Reasoning about physical commonsense in natural language”, arXiv preprint arXiv:1911.11641 (2019).

Clark, P., I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick and O. Tafjord, “Think you have solved question answering? try arc, the ai2 reasoning challenge”, arXiv preprint arXiv:1803.05457 (2018).

Clark, P., O. Tafjord and K. Richardson, “Transformers as soft reasoners over language”, arXiv preprint arXiv:2002.05867 (2020).

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, “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)”, pp. 4171–4186 (2019).

Gontier, N., K. Sinha, S. Reddy and C. Pal, “Measuring systematic generalization in neural proof generation with transformers”, arXiv preprint arXiv:2009.14786 (2020).

Gururangan, S., A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey and N. A. Smith, “Don’t stop pretraining: Adapt language models to domains and tasks”, arXiv preprint arXiv:2004.10964 (2020).

Khashabi, D., T. Khot, A. Sabharwal, O. Tafjord, P. Clark and H. Hajishirzi, “Unifiedqa: Crossing format boundaries with a single qa system”, arXiv preprint arXiv:2005.00700 (2020).

Koupaee, M. and W. Y. Wang, “Wikihow: A large scale text summarization dataset”, arXiv preprint arXiv:1810.09305 (2018).

Kwiatkowski, T., J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, “Natural questions: a benchmark for question answering research”, Transactions of the Association for Computational Linguistics **7**, 453–466 (2019).

Lin, B. Y., X. Chen, J. Chen and X. Ren, “Kagnet: Knowledge-aware graph networks for commonsense reasoning”, 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)”, pp. 2822–2832 (2019).

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach", arXiv preprint arXiv:1907.11692 (2019).

Lv, S., D. Guo, J. Xu, D. Tang, N. Duan, M. Gong, L. Shou, D. Jiang, G. Cao and S. Hu, "Graph-based reasoning over heterogeneous external knowledge for commonsense question answering.", in "AAAI", pp. 8449–8456 (2020).

Marcus, G. and E. Davis, *Rebooting AI: Building artificial intelligence we can trust* (Pantheon, 2019).

McCarthy, J., *Programs with common sense* (RLE and MIT computation center, 1959).

Mishra, S., D. Khashabi, C. Baral and H. Hajishirzi, "Natural instructions: Benchmarking generalization to new tasks from natural language instructions", arXiv preprint arXiv:2104.08773 (2021).

Mostafazadeh, N., N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli and J. Allen, "A corpus and evaluation framework for deeper understanding of commonsense stories", arXiv preprint arXiv:1604.01696 (2016).

Pennington, J., R. Socher and C. D. Manning, "Glove: Global vectors for word representation", in "EMNLP (2014)", pp. 1532–1543 (2014), URL `http://www.aclweb.org/anthology/D14-1162`.

Prakash, A., A. Sharma, A. Mitra and C. Baral, "Combining knowledge hunting and neural language models to solve the winograd schema challenge", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 6110–6119 (2019).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", arXiv preprint arXiv:1910.10683 (2019).

Richardson, K. and A. Sabharwal, "What does my qa model know? devising controlled probes using expert knowledge", Transactions of the Association for Computational Linguistics **8**, 572–588 (2020).

Richardson, M., C. J. Burges and E. Renshaw, "Mctest: A challenge dataset for the open-domain machine comprehension of text", in "Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing", pp. 193–203 (2013).

Robertson, S. E. and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval", in "SIGIR'94", pp. 232–241 (Springer, 1994).

Roemmele, M., C. A. Bejan and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning", in "2011 AAAI Spring Symposium Series", (2011).

Sap, M., R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith and Y. Choi, “Atomic: an atlas of machine commonsense for if-then reasoning”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 33, pp. 3027–3035 (2019a).

Sap, M., H. Rashkin, D. Chen, R. LeBras and Y. Choi, “Social iqa: Commonsense reasoning about social interactions”, in “EMNLP”, (2019b).

Talmor, A., O. Tafjord, P. Clark, Y. Goldberg and J. Berant, “Teaching pre-trained models to systematically reason over implicit knowledge”, arXiv preprint arXiv:2006.06609 (2020).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, “Attention is all you need”, in “Advances in neural information processing systems”, pp. 5998–6008 (2017).

Wang, C. and H. Jiang, “Explicit utilization of general knowledge in machine reading comprehension”, in “Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics”, pp. 2263–2272 (2019).

Wilson, E. B., “Probable inference, the law of succession, and statistical inference”, Journal of the American Statistical Association **22**, 158, 209–212 (1927).

Winograd, T., “Understanding natural language”, Cognitive psychology **3**, 1, 1–191 (1972).

Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov and C. D. Manning, “Hotpotqa: A dataset for diverse, explainable multi-hop question answering”, in “Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing”, pp. 2369–2380 (2018).

Zhong, W., D. Tang, N. Duan, M. Zhou, J. Wang and J. Yin, “Improving question answering by commonsense-based pre-training”, in “CCF International Conference on Natural Language Processing and Chinese Computing”, pp. 16–28 (Springer, 2019).

Zhu, Y., L. Pang, Y. Lan and X. Cheng, “L2r2: Leveraging ranking for abductive reasoning”, arXiv preprint arXiv:2005.11223 (2020).

Chapter 7

AN UNIFIED MODELING APPROACH IN THE CYBERSECURITY DOMAIN

ABSTRACT

With the increase in cybersecurity vulnerabilities of software systems, the ways to exploit them are also increasing. Besides these, malware threats, irregular network interactions, and discussions about exploits in public forums are also on the rise. To identify these threats faster, to detect potentially relevant entities from any texts, and to be aware of software vulnerabilities, automated approaches are necessary. Application of natural language processing (NLP) techniques in the Cybersecurity domain can help in achieving this. However, there are challenges such as the diverse nature of texts involved in the cybersecurity domain, the unavailability of large-scale publicly available datasets, and the significant cost of hiring subject matter experts for annotations. One of the solutions is building multi-task models that can be trained jointly with limited data. In this work, we introduce a generative multi-task model, Unified Text-to-Text Cybersecurity (UTS), trained on malware reports, phishing site URLs, programming code constructs, social media data, blogs, news articles, and public forum posts. We show UTS improves the performance of some cybersecurity datasets. We also show that with a few examples, UTS can be adapted to novel unseen tasks and the nature of data.

7.1 Introduction

In recent times, increasing cybersecurity risks, malware threats, and ransomware attacks are getting more dangerous and common. There are often discussions about them in public forums (Li *et al.*, 2021; Almukaynizi *et al.*, 2018, 2017b) and social media (Shu *et al.*, 2018; Huang and Wu, 2020) both before and after attacks. Private companies also release detailed malware reports such as Symantec (DiMaggio, 2015) and Cylance (Gross and team, 2016). In addition, government agencies (NIST) and other non-profit organizations keep reports of software vulnerabilities through NVD[1] and MITRE[2] respectively to prevent exploitations. Natural language processing (NLP) methods can help in reducing the potential threats by identifying texts mentioning cybersecurity vulnerabilities or malicious exploits (*text-classification*), extracting mentions of relevant entities of threats in public discussions (*named entity recognition*) or identifying the relation of threats with other entities (*relation classification*). It can be used to extract a threat or an event from a text (*event detection*) along with its arguments (*event argument extraction*) to find its source or estimate its damage.

Natural language (NL) domains have seen significant improvements in all the natural language understanding (NLU) tasks with many powerful transformer-based models such as BERT (Devlin *et al.*, 2019a), RoBERTa (Liu *et al.*, 2019b), and XLNet (Yang *et al.*, 2019b) in recent times. With improvements in the NL domain, these models have been adapted and shown to improve performance in other domains such as BioBERT (Lee *et al.*, 2020b), mimicBERT (Singh *et al.*, 2020), Clinical-BERT (Huang *et al.*, 2019), blueBERT (Peng *et al.*, 2019) in the biomedical domain, sciBERT (Beltagy *et al.*, 2019) in the scientific domain (computer science and biomedical), LegalBERT (Chalkidis *et al.*, 2020) in the legal domain and FinBERT (Liu *et al.*,

[1] `https://nvd.nist.gov/`

[2] `https://cve.mitre.org/`

2020b) in the financial service domain. Motivated by these approaches, we introduce a unified model in the cybersecurity domain capable of performing multiple NL tasks.

Unlike other domains, in Cybersecurity domain the nature of texts is quite diverse (natural language text, URLs, malware reports, system calls, source code, binaries, decompiled code, network traffic, software logs (Phandi *et al.*, 2018; Kirillov *et al.*, 2011; Queiroz *et al.*, 2019; Marchal *et al.*, 2014; Satyapanich *et al.*, 2020; Bridges *et al.*, 2013; Chua *et al.*, 2017; Zhang *et al.*, 2021a; Pei *et al.*, 2021b)). This led to the introduction of specific models capable of performing individual tasks like cyber-bullying detection CyberBERT (McDonnell *et al.*, 2021) and cybersecurity claim classification CyBERT (Ameri *et al.*, 2021). Apart from this, there is a scarcity of large-scale publicly available annotated datasets. These challenges demand the need of developing *robust* models capable of performing *multiple tasks* by *learning from many datasets together* and can *perform unseen tasks* or *known tasks on unseen datasets*. Hence, we introduce an **U**nified, **T**ext-to-Text Cyber**S**ecurity (UTS) model.

In this work, a transformer-based generative model, T5 (Brown *et al.*, 2020b), is trained in a multi-task setting on *eight* fine-grained NLP tasks involving *13* datasets in the cybersecurity domain. We used task-based prompt prefixes to help the model to learn the task instead of learning specific datasets. Our goal is to make the model more robust by training on a variety of texts. We show the model's applicability on unseen tasks (task transfer) and on unseen datasets (domain transfer) in three few-shot settings. In the spirit of open science, we will release our research artifacts, including all processed datasets, the source code, and our trained models, upon acceptance.

We summarize our contributions as follows. We

- Propose a unified text-to-text transformer model (UTS) in the cybersecurity domain which is capable of performing four fundamental NLP tasks and their sub-tasks. To the best of our knowledge, this is the first attempt to unify varied text nature in this domain.

- Establish a benchmark of *13* existing cybersecurity datasets processed in text-to-text format involving *eight* NLP tasks for future models to compare with.

- Perform extensive experiments with UTS to assess its ability to adapt to novel tasks and the nature of texts in three few-shot settings.

Dataset	**Nature**	**Cybersecurity Task**	**Mapped NLP Task**	**Dataset Identifier**	**#Samples**	**#Class**
MalwareTextDB-V_2 (Phandi *et al.*, 2018)	APT Reports	Malware Text Detection	Text Classification	MDB-SENTCLS	12,736	2
MalwareTextDB-V_2 (Phandi *et al.*, 2018)	APT Reports	Malware Entity Relation Identification	Relation Classification	MDB-RELCLS	10,802	4
CyberThreatDetection (Queiroz *et al.*, 2019)	Public Forum Posts	Hacker's Threat Detection	Text Classification	CTD	12,575	2
SMS Spam (Almeida *et al.*, 2011)	Text Messages	Spam Message Detection	Text Classification	SMS-SPAM	5,574	2
Phishstorm (Marchal *et al.*, 2014)	URLs	Phishing URL Detection	Text Classification	URL	95,911	2
Soft-Flaw CLS (Saganowski, 2020))	Social Media (Twitter)	Vulnerable Tweet Detection	Text Classification	Soft-Flaw CLS	1,000	2
CASIE (Satyapanich *et al.*, 2020)	CS News Articles	Event Argument Role Identification	Token Classification	CASIE-ARGROLE	11,222	13
Stucco Auto-labelled (Bridges *et al.*, 2013)	NVD-CVE Descriptions	Information Security Entities Extraction	Named Entity Recognition	SAL	15,192	15
Soft-Flaw NER (Saganowski, 2020)	Social Media (Twitter)	Cybersecurity entity Detection	Named Entity Recognition	Soft-Flaw NER	826	1
SOFTNER (Tabassum *et al.*, 2020)	Text with Source Codes	Computer Programming Entity Extraction	Named Entity Recognition	SOFTNER	24,092	20
CASIE (Satyapanich *et al.*, 2020)	CS News Articles	Event nuggets(keywords) Extraction	Event Extraction	CASIE-EVTDET	16,230	5
CASIE (Satyapanich *et al.*, 2020)	CS News Articles	Detect arguments of event from sentence	Event Argument Extraction	CASIE-ARGDET	17,956	21
CVSS (Shahid and Debar, 2021)	CVE Description	Vulnerability Impact Score Estimation	Score Generation	CVE-IMPACT	48,827	-

Table 7.1: Dataset Descriptions With Eight Fine-Grained NLP Tasks. #Samples Represent Full Dataset Samples

7.2 Approach

We develop a generative transformer-based model (UTS) trained on various natures of texts in multi-task settings to perform fundamental NLP tasks like text classification (CLS), named entity recognition (NER), event detection (ED), and Score Generation (GEN) together. We assign task-based control codes (prompts) to teach the model different tasks. Our approach can be seen in Figure 7.1.

Generative approach: We consider T5-base as the underlying model of UTS. This generative text-to-text approach helps us to formulate various NLP tasks into a uniform input-output format and train together with multiple tasks. For CLS tasks, we train the model to generate the exact class names for the given input. For NER and ED tasks, the model needs to extract the entities in a given text along with their types. So, we train the models to generate a concatenation (using '—') of the entity name along with its type (separated by '*'). The model is trained to generate the exact Score Generation scores for GEN task.

Multi-Task Training: All the training datasets of these four fundamental NLP tasks - CLS, NER, ED, and GEN - are grouped together for joint training with the hypothesis that in this way the model can learn from more examples of the same task and similar examples of multiple tasks. Under CLS task, there are four fine-grained classification tasks: Text, Sentence, Relation, and Token Classification. We parse the textual output generated by the model and evaluate UTS on the test data of each of the corresponding datasets. To avoid confusion in the model in identifying similar yet textually different categories, we use unique mapping of the entity types across all extraction tasks.

Prompt-Based Approach: We use task-based control codes as prompt-prefix for training the models in a multi-task setting so that it learns to perform each task instead of learning from any particular dataset. We prepend task acronyms CLS, NER, EVNT, and GEN with the input for classification, named entity recognition, event detection, and Score Generation tasks respectively.

Problem Formulation: The problem formulation is defined here. Given an input text $I = \{i_1, i_2, ..., i_n\}$ and a task T, the model should generate a stream of output tokens O = $\{o_1|o_2|...|o_n\}$ defined by the task. For CLS and GEN tasks, O = $\{o_1\}$, which represents the class name and floating-point value respectively. For NER and ED tasks, each o_i represents an entity and entity type separated by a pre-defined marker i.e. o_i = $\{e_i \star t_i\}$. The task (T) is formulated as an instruction to help the models learn individual tasks in this setting.

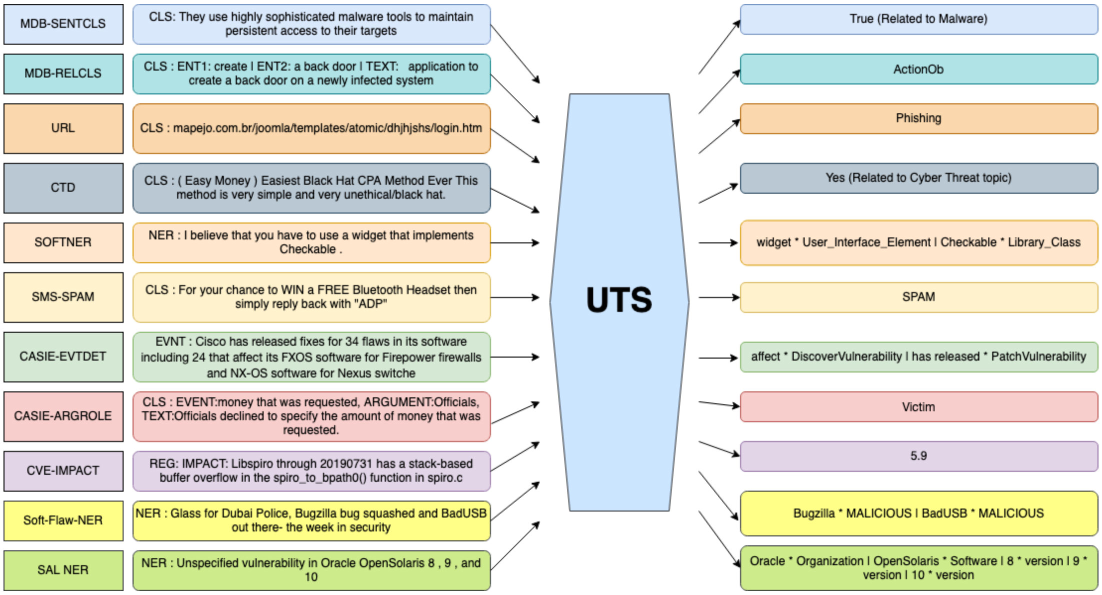

MDB-SENTCLS
CLS: They use highly sophisticated malware tools to maintain persistent access to their targets
MDB-RELCLS
CLS : ENT1: create | ENT2: a back door | TEXT: application to create a back door on a newly infected system
URL
CLS : mapejo.com.br/joomla/templates/atomic/dhjhjshs/login.htm
CTD
CLS : (Easy Money) Easiest Black Hat CPA Method Ever This method is very simple and very unethical/black hat.
SOFTNER
NER : I believe that you have to use a widget that implements Checkable .
SMS-SPAM
CLS : For your chance to WIN a FREE Bluetooth Headset then simply reply back with "ADP"
CASIE-EVTDET
EVNT : Cisco has released fixes for 34 flaws in its software including 24 that affect its FXOS software for Firepower firewalls and NX-OS software for Nexus switche
CASIE-ARGROLE
CLS : EVENT:money that was requested, ARGUMENT:Officials, TEXT:Officials declined to specify the amount of money that was requested.
CVE-IMPACT
REG: IMPACT: Libspiro through 20190731 has a stack-based buffer overflow in the spiro_to_bpath0() function in spiro.c
Soft-Flaw-NER
NER : Glass for Dubai Police, Bugzilla bug squashed and BadUSB out there- the week in security
SAL NER
NER : Unspecified vulnerability in Oracle OpenSolaris 8 , 9 , and 10
UTS
True (Related to Malware)
ActionOb
Phishing
Yes (Related to Cyber Threat topic)
widget * User_Interface_Element | Checkable * Library_Class
SPAM
affect * DiscoverVulnerability | has released * PatchVulnerability
Victim
5.9
Bugzilla * MALICIOUS | BadUSB * MALICIOUS
Oracle * Organization | OpenSolaris * Software | 8 * version | 9 * version | 10 * version

Figure 7.1: Illustration Of UTS (Unified Text-to-Text CyberSecurtiy) Model

7.3 Dataset Preparation

We prepare *13* datasets involved in *eight* NLP tasks. The summary of the datasets is presented in Table 7.1. There are four fundamental tasks in the collected datasets; Classification, Event Detection, Named Entity Recognition, and Score Generation respectively. For each of the datasets, we used the original test splits if mentioned in the paper. Otherwise, we consider 20% of the data chosen randomly as the test split. The details of each dataset are as follows.

7.3.1 Classification

MalwareTextDB-V2: This dataset (Phandi *et al.*, 2018) is constructed from 83 APT reports. Each report contains multiple paragraphs regarding various activities of malware. We consider two tasks from this dataset for *UTC*. They are : (1) *Sentence Classification* - classifying whether individual sentences are relevant to cybersecurity applications, and (2) *Relation Classification* - classifying the relation between two given entities.We take 68 documents as a train and 15 documents as test datasets. Each document has multiple sentences which we pre-process as each input sample.

SMS-SPAM: Another classification subtask is to classify spam messages. This benchmark dataset (Almeida *et al.*, 2011) is for detecting SMS spam messages. The SMS-SPAM dataset is a combination of several publicly available SMS corpora and websites.

CyberThreatDetection: This dataset (Queiroz *et al.*, 2019) was constructed from various hacker forums. Posts were collected and labeled by humans into 3 categories. *Yes*, for posts that appear as malicious posts. *No*, for posts not related to hacker activity. *Undecided*, for posts where the annotator did not have enough information. The original authors counted the *Undecided* labels as *Yes* labels.

PhishStorm: This dataset (Marchal *et al.*, 2014) includes around 96k URLs. These URLs are labeled as normal or phishing and were collected through PhishTank,[3] which is a crowd-sourced project where people submit phishing URLs and were later confirmed by several people.

7.3.2 Event Detection

CASIE: This is the first cybersecurity Event Detection dataset (Satyapanich *et al.*, 2020) with five main types of events. We consider three tasks from this dataset: (1) *Event Extraction* (2) *Event Argument Detection* and (3) *Event Argument Role Detection.* Event Extraction is a task to extract event nuggets which are words or phrases that best express the event occurrence clearly. Event Argument Detection is a task to detect event arguments that are event participants or property values. They can be tangible entities involved in the event such as a person or organization, or attributes that specify important information such as time or amount. Event

[3] `http://www.phishtank.com`

Argument Role Detection is a task to find roles between given event nuggets and event arguments. A role is a semantic relation between an event nugget and an argument. Thus, each event type specifies the roles it can have and constraints on the arguments that can fill them.

7.3.3 Named Entity Recognition

Stucco-Autolabeled: This dataset (Bridges *et al.*, 2013) is constructed from Common Vulnerabilities and Exposure (CVE) databases containing descriptions of information security issues from Jan 2010 to Mar 2013. In the Stucco-Autolabeled dataset, each word in the corpus is auto-annotated with an entity type. This dataset has 15 entity types.

SOFTNER: This dataset (Tabassum *et al.*, 2020) has 20 annotated entity types from 1237 StackOverflow QA pairs. The text is embedded with source code constructs from many programming languages.

Soft-Flaw NER: Cybersecurity NER corpus 2019 corpus (Saganowski, 2020) consists of 1000 annotated tweets. The entities marked are usually the name of the software/system/device/company with a security-related issue or the name of malware. There is a corresponding classification dataset as well (Soft-Flaw CLS).

7.3.4 Score Generation

NVD CVE metrics: The NIST National Vulnerability Dataset uses vulnerabilities found through the CVE (Common Vulnerabilities and Exposure) system. Human security experts assign a corresponding CVSS (Common Vulnerability Scoring System) vector, and from that, the exploitability and impact score for the vulnerability is calculated. We split the data from 2002 onward into train and test in a 1:1 proportion as per the previous work (Shahid and Debar, 2021) and directly generate the scores from the descriptions.

We describe our Unified Model datasets and Transfer Learning datasets in the next subsection.

7.3.5 Unified Model Datasets

Out of 13 datasets in Table 7.1, we jointly train UTS on 10 datasets: MDB-SENTCLS, MDB-RELCLS, URL, CTD, SMS-SPAM, and CASIE-ARGROLE for classification, CASIE-EVTDET for event detection, SOFTNER and SAL for named entity recognition and CVSS-IMPACT for Score Generation task. We consider the full volume of each of these datasets for unified training.

7.3.6 Transfer Learning Datasets

Task Transfer: We prepare Entity Extraction (EE) and Entity Typing (ET) tasks from two NER datasets; SAL, and SOFTNER. We also prepare event argument extraction (EAE) and event argument typing (EAT) tasks from CASIE-ARGDET dataset. From the Soft-Flaw NER dataset, we only prepare a dataset for the EE task since there is only one entity type 'Malicious'. We do not consider Soft-Flaw during unified training since the dataset volume is small. To prepare the few-shot datasets, we randomly pick at least one sample per type from EE and EAE tasks (if the number of types is more than the size of the few-shot dataset, we randomly pick a subset) to make label-balanced data.

Domain Transfer: We only consider the Soft-Flaw dataset for this experiment since the nature of texts is unique as compared to other datasets. This dataset is prepared from social media (Twitter) texts and can be used to see the adaptability of UTS on different natures of texts. To prepare the few-shot datasets, we make sure that the positive and negative samples are balanced.

7.4 Experiments

7.4.1 Unified Experiments

First, we pre-process the training data of each of these 10 datasets into the text-to-text format as mentioned in subsection 7.3.5. Then, we train T5-base in a multi-task setting with the prepared training data. After the training, we evaluate the trained model (UTS) on individual test datasets. In addition, we compare the performance with existing best models and T5-base trained individually with each dataset.

7.4.2 Few-shot Experiments

We consider three few-shot settings (FS-20, FS-50, and FS-100) based on the number of examples (20, 50, 100) on which UTS is trained on.

Task Transfer: We experiment to see whether UTS can adapt to novel tasks from another known task on 3 few-shot settings and compare with T5-base trained on the full dataset. To understand the extent of task transfer by UTS, we consider three few-shot sub-categories: (1) *Domain Known Task Related (DKTR)*: trained model has knowledge of the data and NER task but has not learned entity extraction (EE) and entity typing (ET) tasks (2) *Domain Known Task Unrelated (DKTU)*: trained model knows the data and how to perform event detection (ED) task and event argument role classification task but does not know event argument extraction (EAE) and event argument typing (EAT) tasks. Here, Argument Role and Argument Types classes are different and (3) *Domain Unknown Task Related (DUTR)* - trained model knows NER task but neither knows EE task nor has seen the data.

Domain Transfer: We experiment with whether UTS can adapt to a different textual nature input for a known task. We consider the social media (Twitter) dataset, Soft-Flaw (both CLS and NER tasks), for this experiment in three few-shot settings where we train UTS with 20, 50, and 100 training samples and evaluate on full test data. In addition, we compare UTS with T5 trained on full training data. For the Soft-Flaw CLS dataset, the zero-shot experiment is done to see if UTS can adapt to a different text nature without training on any examples.

7.4.3 Metrics

The generated output string is parsed and evaluated based on the task. For all variations of classification and Score Generation, we consider an *exact match* between the generated and original gold output. For extraction tasks, we parse generated outputs based on the predefined markers to get the entity sets (entity name and entity type). We report weighted F1 scores for all the tasks except the Score Generation tasks where we consider exact-match accuracy as the metric for evaluation.

7.4.4 Experimental Setup

We use T5-base (220M parameters) for UTS. We set a predefined training budget of 30 epochs and hyperparameter tuning for our experiments. We train with a 5e-5 learning rate and 0.01 warm-up ratio. We perform the experiments with four 81GB Nvidia A100 GPUs with a training batch size of 12. We consider a beam size of 4. The average training time is ~24hrs.

7.4.5 Implementation

For building and training UTS, we use publicly available packages : PyTorch (Paszke *et al.*, 2019) 1.9.1, HuggingFace Transformers (Wolf *et al.*, 2020b) 4.15.0, HuggingFace Datasets (Lhoest *et al.*, 2021) 1.16.1, seqeval (Nakayama, 2018) 1.2.2, sklearn (Pedregosa *et al.*, 2011) 1.0 and pandas 1.3.4.

Dataset	Previous Best	T5	UTS
MDB-SENTCLS	57.00◊	84.04	84.44
MDB-RELCLS	85.70◊	99.79	99.69
CTD‡	93.00⧫	92.17	92.00
SMS-SPAM†	91.90★	99.45	98.54
URL†	94.70♣	98.99	99.01
SAL	93.40♡	98.46	97.60
SOFTNER	79.10△	72.90	77.02
CASIE-EVTDET	79.90♠	81.43	83.53
CASIE-ARGROLE†	82.90♠	91.67	92.50
CVE-IMPACT	NA	76.58	76.95

Table 7.2: Performance (Wtd F1 Score) Of UTS Compared To T5-base Trained On Individual Datasets And Previous Best. ‡ Represents The Performance Is Compared With Positive Recall. † Represents The Performance Is Compared With Macro-F1 Score While Weighted-F1 Score For The Rest. The Previous Best Scores Are From The Following Works Respectively; ◊ (Phandi *et al.*, 2018),⧫ (Queiroz *et al.*, 2019), ★ (Mohasseb *et al.*, 2020), ♣ (Marchal *et al.*, 2014), ♠ (Satyapanich *et al.*, 2020), ♡ (Simran *et al.*, 2019), And △ (Tabassum *et al.*, 2020).

7.5 Results and Discussion

We first try to understand the impact of the multi-task training approach on each dataset by comparing it with a fully supervised T5 model on individual datasets. Then we test the adaptability of the model on an unknown domain and unknown tasks in few-shot and zero-shot settings.

R1: How much does the multi-task training help UTS?

Multi-task training helps UTS to gain an understanding of multiple text-nature in the Cybersecurity domain. Most importantly, it helps the model to be tuned to fundamental NLP tasks. Table 7.2 shows the performance of T5 trained on individual training data compared to UTS trained on all 10 datasets in a multi-task setting. We find that SOFTNER and CASIE-EVTDET show 4% and 2% improvements respectively. For the rest of the tasks, the performance change is marginal and most importantly it does not drop significantly. Thus, the trained UTS model has the understanding of four fundamental NLP tasks controlled by task-specific control codes.

In addition, we show how UTS performs as compared to the previous best approaches in Table 7.2. We can see there is an improvement of ~3% up to ~27% in eight datasets. The Previous Best scores and their methods from the following works respectively; ◊ (Phandi *et al.*, 2018): BiLSTM for MDB-SENTCLS and Rule-Based method for MDB-RELCLS,⧫ (Queiroz *et al.*, 2019): CNN + Word Embedding method, ★ (Mohasseb *et al.*, 2020): Random Forest classifier with SMOTE algorithm, ♣ (Marchal *et al.*, 2014): Random Forest, ♠ (Satyapanich *et al.*, 2020): pure-built BERT method for EVTDET and Noe Event Specific system for ARGROLE, ♡ (Simran *et al.*, 2019): Bidirectional GRU+CNN-CRF model, and △ (Tabassum *et al.*, 2020): SOFTNER (BERTOverflow). The performance, however, drops by 2% for SOFTNER because of the use of domain-specific embeddings and hence lack of generalization of text nature.

R2: To what extent Task Transfer is possible with UTS in few-shot settings? The motivation of UTS is to develop a unified model tuned to multiple tasks Table 7.3 shows the performance of various settings of Entity Extraction (EE) task transfer. We can see for both DKTR and DKTU settings, even with only 20 samples, UTS can achieve performance very close (within ~2%) to T5-base trained on full data. This shows even though the model is not explicitly trained for a task it has learned to

Dataset	FS-20	FS-50	FS-100	T5-FL
CASIE-EVTARG (DKTU)	65.90	66.23	67.64	69.89
SAL (DKTR)	89.31	89.63	89.73	90.42
SOFT-NER (DKTR)	78.60	78.22	80.45	80.85
Soft-Flaw NER (DUTR)	50.10	53.16	54.95	76.71

Table 7.3: Entity Extraction (EE) Task Transfer - FS: Few-shot UTS On 20, 50, 100 Samples, T5-FL: T5 On Full

Dataset	FS-20	FS-50	FS-100	T5-FL
CASIE-EVTARG (DKTU)	86.26	94.61	96.09	97.94
SAL (DKTR)	1.06	3.46	85.86	99.44
SOFT-NER (DKTR)	28.65	33.97	42.67	76.69

Table 7.4: Entity Typing (ET) Task Transfer - FS: Few-shot UTS On 20, 50, 100 Samples, T5-FL: T5 On Full

generalize well on similar tasks with very few examples. However, for DUTR, the model achieves (~50 F1 points) for FS-20 but falls quite short (~12 F1 points) of the T5-FL setting. This shows that task transfer becomes challenging when the data nature significantly changes.

Table 7.4 shows the performance of various settings of Entity Typing (ET) task transfer. This task is harder for the models since the model has to generate the types not present in the text provided. Hence it can be seen that the FS-20 performance is poor as compared to T5-FL for DKTR categories. For the DKTU category, the model has the knowledge from two unrelated tasks of assigning roles to event arguments and detecting events. Even though the role categories do not overlap with arguments, the model still has some understanding of the argument type from these two unrelated tasks.

For both the tasks, EE and ET, UTS's performance increased with more samples in all settings. We also notice that T5-base performed poorly for each few-shot setting (Tables 7.5 and 7.6) and often was not able to correct a single sample.

Dataset	FS-20	FS-50	FS-100	T5-FL
CASIE-EVTARG (DKTU)	0.00	0.00	11.44	69.89
SAL (DKTR)	0.00	0.04	71.34	90.42
SOFT-NER (DKTR)	46.2	20.85	38.35	80.85
Soft-Flaw-NER (DUTR)	0.00	0.00	53.73	76.71

Table 7.5: Entity Extraction (EE) Task Transfer - FS: Few-shot T5 Base Model On 20, 50, 100 Samples, T5-FL: T5 On Full

Dataset	FS-20	FS-50	FS-100	T5-FL
CASIE-EVTARG (DKTU)	23.65	33.50	71.52	97.94
SAL (DKTR)	0.20	0.07	72.89	99.44
SOFT-NER (DKTR)	21.64	16.96	22.53	76.69

Table 7.6: Entity Typing (ET) Task Transfer - FS: Few-shot T5 Base Model On 20, 50, 100 Samples, T5-FL: T5 On Full

R3: To what extent Domain Transfer is possible with UTS in few-shot

	Soft-Flaw (CLS)	Soft-Flaw (NER)
FS-20 (UTS)	82.14	50.00
FS-50 (UTS)	82.17	61.54
FS-100 (UTS)	82.21	65.67
Supervised (T5-FL)	83.63	76.71

Table 7.7: Domain Transfer On Twitter Dataset, Supervised: Trained With T5-Base on Full Dataset, Fewshot(FS) With 20, 50, 100 Samples.

settings ?

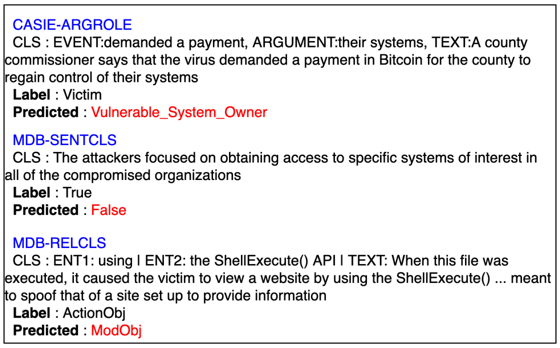
Table 7.7 shows how much domain transfer can UTS perform with Twitter texts. For the Soft-Flaw CLS dataset, FS-20 performance is within ~1.5% F1 of T5-FL while Soft-Flaw NER dataset the model falls quite short of the T5 full dataset trained model. This, we believe, is because classification is an easier task than NER for generative models and can be learned with only a few examples.

R4: Is it possible to perform Zero-shot Domain Transfer with UTS ? We explore if UTS can be adapted to another domain for the same task or the same domain for some other tasks in a zero-shot setting. We find that for the classification task (CLS), UTS can predict whether a text is 'malicious' or not with 82.92% F1 which is less than 1% short of T5-FL. This is marginally greater than FS-100 performance. A possible explanation can be that these few-shot datasets are label balanced and the model learns well for the positive labels and not so well for the negative labels.

7.6 Case Studies

We analyze the prediction output of UTS for each dataset. Here we present a few of them.

Classification: Figure 7.2 shows one example from each classification dataset where our model fails. Here, the first example (CASIE Event Role) is interesting since the classification decision is quite close. The model understood that the 'their system' argument is a vulnerable system but failed to understand that it is not the owner. In the MDB-RELCLS example the relation between the two entities 'using' and 'the ShellExecute() API' should be Action Object rather than Modifier Object.

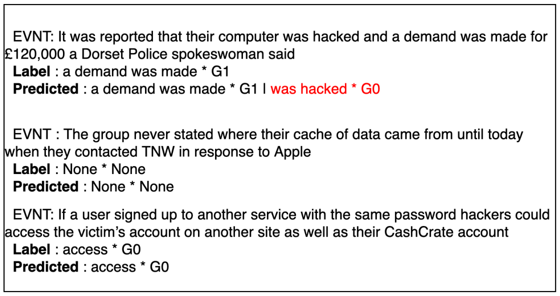


CASIE-ARGROLE
CLS : EVENT:demanded a payment, ARGUMENT:their systems, TEXT:A county commissioner says that the virus demanded a payment in Bitcoin for the county to regain control of their systems
Label : Victim
Predicted : Vulnerable_System_Owner

MDB-SENTCLS
CLS : The attackers focused on obtaining access to specific systems of interest in all of the compromised organizations
Label : True
Predicted : False

MDB-RELCLS
CLS : ENT1: using I ENT2: the ShellExecute() API I TEXT: When this file was executed, it caused the victim to view a website by using the ShellExecute() ... meant to spoof that of a site set up to provide information
Label : ActionObj
Predicted : ModObj

Figure 7.2: CLS Example Predictions: MDB And CASIE



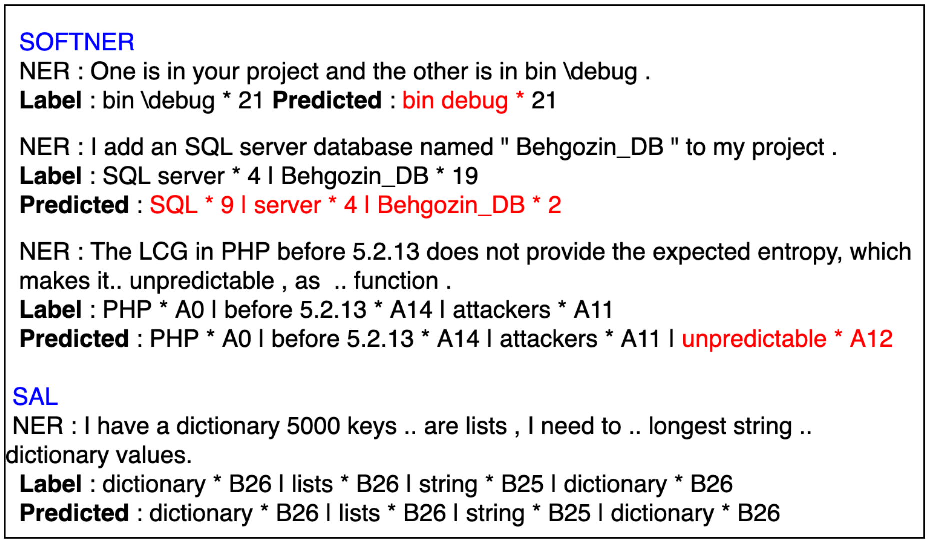
EVNT: It was reported that their computer was hacked and a demand was made for £120,000 a Dorset Police spokeswoman said
Label : a demand was made * G1
Predicted : a demand was made * G1 I was hacked * G0

EVNT : The group never stated where their cache of data came from until today when they contacted TNW in response to Apple
Label : None * None
Predicted : None * None

EVNT: If a user signed up to another service with the same password hackers could access the victim's account on another site as well as their CashCrate account
Label : access * G0
Predicted : access * G0

Figure 7.3: Event Detection: CASIE

Event Detection: Figure 7.3 shows some examples of successful and incorrect prediction cases of the CASIE Event Detection dataset. In the first example, two events are detected out of which one is correct but the other is difficult to understand by the model. Here "was hacked * Databreach" is not in the gold label since ransomware attacks do not always link to databreach. In addition, CASIE only has five event types (Databreach, Phishing, Ransom, Vulnerability (discover), and Vulnerability (patch). These five types do not cover the whole cybersecurity event such as Malware, Viruses, Trojans, and Spyware. Thus, we suspect that our model detected a potential event phrase “was hacked” and assigned one of the five event types even if there is no suitable type in the candidates.



SOFTNER

NER : One is in your project and the other is in bin \debug .
Label : bin \debug * 21 **Predicted** : bin debug * 21

NER : I add an SQL server database named " Behgozin_DB " to my project .
Label : SQL server * 4 | Behgozin_DB * 19
Predicted : SQL * 9 | server * 4 | Behgozin_DB * 2

NER : The LCG in PHP before 5.2.13 does not provide the expected entropy, which makes it.. unpredictable , as .. function .
Label : PHP * A0 | before 5.2.13 * A14 | attackers * A11
Predicted : PHP * A0 | before 5.2.13 * A14 | attackers * A11 | unpredictable * A12

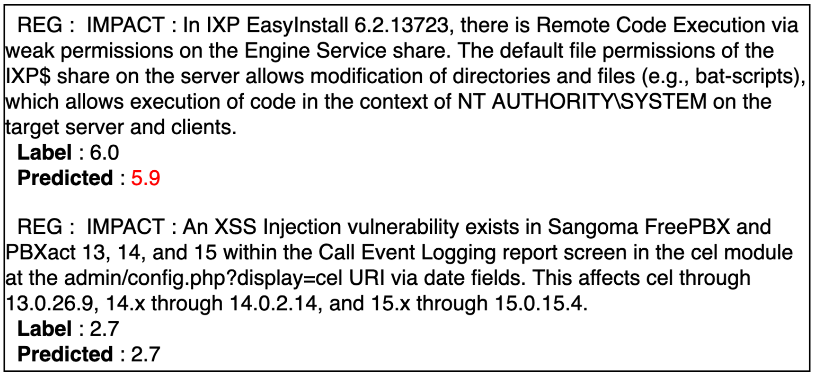
SAL

NER : I have a dictionary 5000 keys .. are lists , I need to .. longest string .. dictionary values.
Label : dictionary * B26 | lists * B26 | string * B25 | dictionary * B26
Predicted : dictionary * B26 | lists * B26 | string * B25 | dictionary * B26

Figure 7.4: Named Entity Recognition

Named Entity Recognition: Figure 7.4 shows four examples of NER tasks. The first three examples are from the SOFTNER dataset and the last one is from the SAL dataset. The first example shows that our model predicted words and entity types correctly. However, the special character “\” is missing from the prediction. Since we use exact-match metrics for evaluation this category of incorrect prediction penalizes the models. We also find similar examples where our models could not generate full entities with other characters like ‘{’ and ‘}’. The second example has two issues; the first one is split ‘SQL server’ into ‘SQL’ and ‘server’ and assigned different entities to the ‘SQL’ part, the second one is that the word/phrase is predicted correctly, however, the entity type is incorrect. We also find cases in the last example, where UTS has correctly predicted more than three entities and their types.

Score Generation: Figure 7.5 shows successful and unsuccessful predictions of the Score Generation task. While the original Impact Scores are calculated based on several features in the vulnerability, our model predicts these scores based on only the textual description of the vulnerability (CVE descriptions). The second example shows that UTS missed predicting the actual impact score by a close margin.



REG : IMPACT : In IXP EasyInstall 6.2.13723, there is Remote Code Execution via weak permissions on the Engine Service share. The default file permissions of the IXP$ share on the server allows modification of directories and files (e.g., bat-scripts), which allows execution of code in the context of NT AUTHORITY\SYSTEM on the target server and clients.
Label : 6.0
Predicted : 5.9

REG : IMPACT : An XSS Injection vulnerability exists in Sangoma FreePBX and PBXact 13, 14, and 15 within the Call Event Logging report screen in the cel module at the admin/config.php?display=cel URI via date fields. This affects cel through 13.0.26.9, 14.x through 14.0.2.14, and 15.x through 15.0.15.4.
Label : 2.7
Predicted : 2.7

Figure 7.5: Score Generation: Impact Score

7.7 Other Case Studies

NER Task Error Analysis:

SAL Dataset:

Text:

The embedded HTTP server in multiple Lexmark laser and inkjet printers and MarkNet devices, including X94x, W840, T656, N4000, E462, C935dn, 25xxN, and other models, allows remote attackers to cause a denial of service (operating system halt) via a malformed HTTP Authorization header.

Gold:

Lexmark * N — X94x * F — W840 * F — T656 * F — N4000 * F — E462 * F — C935dn * F — 25xxN * F — allows * L — remote attackers * L — denial of service * L — Authorization * L

Predicted:

Lexmark * N — inkjet * O — MarkNet * A — X94x * O — W840 * O — T656 * O — N4000 * O — E462 * O — C935dn * O — 25xxN * O — allows * L — remote attackers * L — denial of service * L — Authorization * L

Text:

Certain patch-installation scripts in Oracle Solaris allow local users to append data to arbitrary files via a symlink attack on the /tmp/CLEANUP temporary file , related to use of Update Manager.

Gold:

Solaris * I — local users * L — arbitrary files * L — symlink attack * L

Predicted:

Oracle * N — Solaris * A — local users * L — arbitrary files * L — symlink attack * L

ED Task Error Analysis:

CASIE Event Detection dataset:

Text:

It was reported that their computer was hacked and a demand was made for £120,000 a Dorset Police spokeswoman said

Gold:

a demand was made * Ransom

Predicted:

a demand was made * Ransom — was hacked * Databreach

Text:

EVNT : The group never stated where their cache of data came from until today when they contacted TNW in response to Apple **Gold:**

None * None

Predicted:

None * None

Text:

Launched in 2016 the No More Ransom scheme brings law enforcement and private industry together in the fight against cybercrime and has helped thousands of ransomware victims retrieve their encrypted files without lining the pockets of crooks

Gold:

Ransom * Ransom

Predicted:

The No More Ransom scheme * Ransom

CASIE Event Arguments dataset:

Text:

The attack disabled servers early Tuesday morning, and city officials say it was contained by 5:30 PM Wednesday.

Gold:

servers * System — early Tuesday morning * Time — 5:30 PM Wednesday * Time

Predicted:

disabled servers early Tuesday morning * Capabilities

Text:

In some cases, a generic password is required, although security researchers have discovered that in many cases, FTP servers can be accessed without a password.

Gold:

FTP servers * System — can be accessed without a password * Capabilities — security researchers * Person

Predicted:

security researchers * Person — FTP servers can be accessed without a password * Capabilities

Classification Task Error Analysis

MalwareTextDB v2: *Sentence Classification*

Text:

The Skelky (from skeleton key) tool is deployed when an attacker gains access to a victims network ; the attackers may also utilize other tools and elements in their attack.

Gold:

False

Predicted:

True

Text:

The attackers focused on obtaining access to specific systems of interest in all of the compromised organizations.

Gold: True

Predicted: False

MalwareTextDB v2: *Relation Classification*

Text:

a tool — allows — <doc>.

Gold: SubjAction

Predicted: CoRefer

MalwareTextDB v2: *Attribute Classification*

Text:

executing — After the C&C reply, Moose continues with infection, executing commands on the victim device.

Gold:

Capability TacticalObjectives StrategicObjectives

Predicted:

Capability ActionName StrategicObjectives TacticalObjectives

Text:

obtaining — The attackers focused on obtaining access to specific systems of interest in all of the compromised.

Gold:

Capability StrategicObjectives

Predicted:

Capability

Text:

allows — On January 12, 2015, Dell Secureworks blogged about a tool (Trojan.Skelky) that allows attackers to...

Gold:

Capability StrategicObjectives

Predicted:

ActionName

Score Generation Task Error Analysis

NVD CVE metrics: *Impact score*

Text:

An issue was discovered in B&R Industrial Automation APROL before R4.2 V7.08. Some web scripts in the web interface allowed injection and execution of arbitrary unintended commands on the web server, a different vulnerability than CVE-2019-16364.

Gold: 5.9

Predicted: 3.6

NVD CVE metrics: *Exploitability score*

Text:

Libspiro through 20190731 has a stack-based buffer overflow in the spiro_to_bpath0() function in spiro.c.

Gold: 2.2

Predicted: 2.8

7.8 Related Work

Multitask Learning in Diverse domains: In the natural language domain, DecaNLP (McCann *et al.*, 2018) introduced the approach of converting multiple tasks into a single QA format to train and evaluate ten tasks. With the gradual introduction of stronger generative NLP models like GPT, T5, and BART, the text-to-text unified models gained prominence. The multi-task approach has been shown to perform well in various domains like SciFive (Phan *et al.*, 2021) in the biomedical domain, CodeT5 (Wang *et al.*, 2021) in the source code domain, LEGAL-BERT (Chalkidis *et al.*, 2020) in the legal domain and FinBERT (Liu *et al.*, 2020b) in the financial service domain. Using "teacher forcing" for all tasks for training with a maximum likelihood objective, SciFive enables multitask learning. CodeT5 is a unified pre-trained encoder-decoder Transformer model and it can handle various tasks across various directions between program languages and natural languages.

Task-Based Unified Models: Apart from these, there are individual task-based unified models like InstructionNER which expands the existing methods for sentence-level tasks to an instruction-based generative framework for low-resource named entity recognition (Wang *et al.*, 2022a). In the biomedical domain, KGNER (Banerjee *et al.*, 2021) formulated the NER task as a multi-answer knowledge-guided question-answer task and experimented with 18 datasets. UnifiedNER (Yan *et al.*, 2021) works on unifying span-based, nested, and discontinuous NER tasks. UnifiedQA (Khashabi *et al.*, 2020) showed that a unified training of QA tasks helps in the improvement of other QA tasks. Similar results are shown in common-sense reasoning tasks by Unicorn (Lourie *et al.*, 2021).

NLP Approaches on Cybersecurity: NLP approaches have been applied in the cybersecurity domain on various natures of texts involving function-calls, software binaries, and network traffics (Chua *et al.*, 2017; Zhang *et al.*, 2021a; Pei *et al.*, 2021b). There are approaches that apply to specific cybersecurity tasks like lexical analysis of domain name (Kidmose *et al.*, 2018), a syntactic analysis (parsing) (Perera *et al.*, 2018), keyword extraction for phishing classification (L'Huillier *et al.*, 2010), NER based automated system to diagnose cybersecurity situations in Internet of Thing (IoT) networks (Georgescu *et al.*, 2019). There are many systems using social media, blog posts, and discussion forums for analyzing and extracting CTI(Cyber Threat Intelligence) information as well as measuring the risk of vulnerability exploitation (Zhao *et al.*, 2020; Zenebe *et al.*, 2019; Deliu *et al.*, 2018, 2017; Sabottke *et al.*, 2015; Portnoff *et al.*, 2017; Almukaynizi *et al.*, 2017a, 2019, 2020; Suciu *et al.*, 2021). In addition, there have been works in extracting flow structure from unstructured software vulnerability analysis discussions in public forums (Pal *et al.*, 2021b). However, our focus here is to unify the varied nature of texts and introduce a unified approach in this domain.

7.9 Conclusion and Future Work

In this work, we introduce a multi-nature, multi-task approach, UTS, in the cybersecurity domain. We experiment with T5-base, a transformer-based generative model, and show that the unified approach shows significant improvements on two datasets when compared with individual training. Also, it improves over most of the previous best performances. We show that task transfer is possible when the UTS model is trained with fewer samples of training data. This indicates that UTS can be adapted to new tasks and only a few training samples are necessary. We believe this will reduce the annotation costs for new tasks. We also show that UTS is robust to the new nature of texts and also requires a few samples to adapt. In the future, apart from the four fundamental NLP tasks, we would like to add more tasks such as multi-label classification or relation extraction. We believe the approach and the benchmarks we establish can be used as a baseline for future studies in the cybersecurity domain. The experiments we perform with this UTS approach, are limited to using either natural language texts or text with embedded source code constructs as input. There is a potential to include system calls or binary codes in these unified cybersecurity models. More details are available in the pre-print Pal *et al.* (2023).

7.10 Limitations

We presented a multitask model trained jointly with limited data in the cybersecurity domain. There are some limitations to our work. First, we work with multiple natures of texts aggregating which is a challenge. In this research, our focus is on unifying mostly variations of textual nature along with some embedded software code constructs. We do not include other natures of cybersecurity texts like source code, binaries, decompiled code, or network traffic. Second, we have not included datasets from other languages (NER datasets in Russian texts) which also pose challenges to training in a multi-task setting and might require multi-lingual approaches. Third, for the few-shot experiments we randomly chose some examples from each category to make the dataset label-balanced. Selecting the few-shot examples might lead to minor variations in performance. Fourth, a few of the older datasets have no explicit test set mentioned in their paper. For adapting them to our approach, we randomly chose 20% as a test set leading to a difference in comparison. Hence, we include individual T5-trained baselines as a comparison.

7.11 Ethics Statement

All our experiments are performed with the well-known publicly released model transformer-based model T5. We work on 13 different datasets published in notable peer-reviewed works. We do not create, collect or process these datasets in any way such that they can be considered unethical. Our trained model checkpoints will be able to perform common natural language processing tasks similar to a general natural language processing domain.

Chapter 8

INCORPORATING LOGICAL REASONING SKILLS INTO TRANSFORMERS-BASED LANGUAGE MODELS

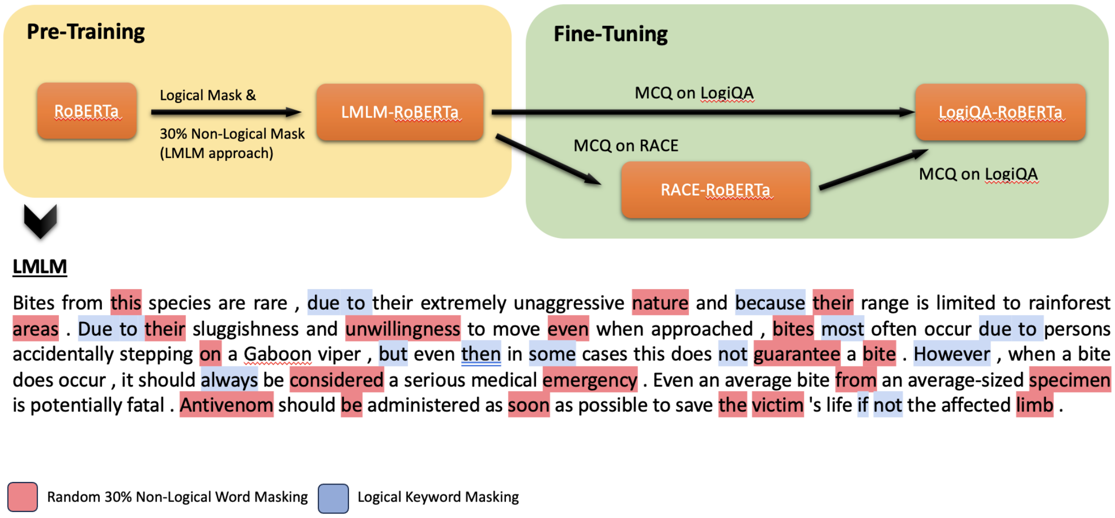
8.1 Introduction

The transformer-based models have made tremendous progress in NLP in the last few years. Even though they have achieved remarkable performance in general NLP tasks, they fall short when it comes to reasoning tasks. Reasoning has been a long-time goal for the Artificial Intelligence community.

Since these models are trained to perform general-purpose NLP tasks, they are pre-trained on a massive collection of texts. The encoder-based transformer models are all training by mask language modeling pre-training (MLM) objective. No special training task or specific attention is provided to the logical words which control the meaning of a sentence. For example, the word “no” reverses the meaning of the contextual situation. It is expected that such models can learn the representation of such keywords automatically learning from a huge pre-training corpus.

So in this project, we make sure that the logical words are identified and masked first. Then the model tries to predict these words and in the process learns a rich representation of such words and their co-relation with other words in the context. Apart from that we also mask the non-logical words as well to learn better co-relations with self-attention.

We find that such a continued logical masked pre-training approach improves the performance of the general RoBERTa model by a significant margin.



Bites from this species are rare , due to their extremely unaggressive nature and because their range is limited to rainforest areas . Due to their sluggishness and unwillingness to move even when approached , bites most often occur due to persons accidentally stepping on a Gaboon viper , but even then in some cases this does not guarantee a bite . However , when a bite does occur , it should always be considered a serious medical emergency . Even an average bite from an average-sized specimen is potentially fatal . Antivenom should be administered as soon as possible to save the victim 's life if not the affected limb .

Figure 8.1: LMLM Approach

8.2 Approach

We demonstrate our overall approach through Figure 8.1. The whole approach can be broken down into three phases: Logical Word Selection, Logical Pre-training and Fine-Tuning.

8.2.1 Logical Word Selection

We locate words that can indicate logical actions. These words can be single words (SLW) or a group of words (MLW). We first create a dictionary of single logical words which has the potential to control the understanding of a statement. These SLW are *’because’, ’since’, ’if’, ’for’, ’why’, ’assuming’, ’therefore’, ’thus’, ’then’, ’so’, ’consequently’, ’argue’, ’conclude’, ’imply’, ’infer’, ’suggest’, ’accordingly’, ’hence’, ’thence’, ’inconsequence’, ’whence’, ’wherefore’, ’not’, ’either’, ’neither’, ’nor’, ’despite’, ’notwithstanding’, ’challenge’, ’object’, ’counter’, ’critique’, ’criticize’, ’conflict’, ’doubt’, ’problem’, ’weakness’, ’some’, ’all’,’few’,’most’,’many’,’each’,’every’,’none’, ’much’, ’someone’,’nobody’,’everybody’.*

8.2.2 Continued Logical Pre-Training

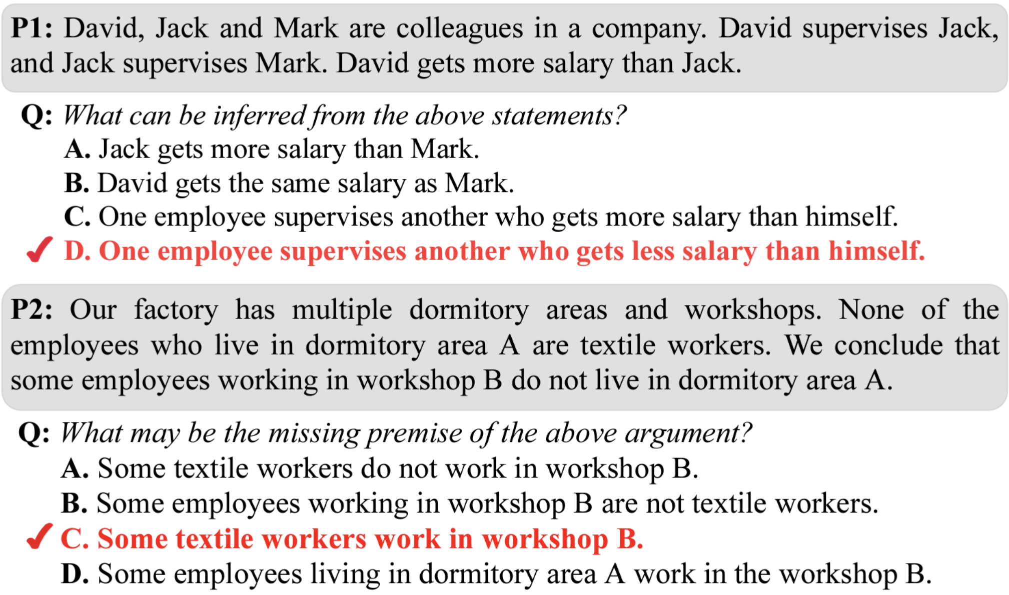
We choose a Transformer-Encoder based language model and continued its pre-training with a different pre-training objective. We consider the large version of RoBERTa (Liu *et al.*, 2019b) called RoBERTa-large. We start our pre-training on Huggingface version of Wikipedia (Merity *et al.*, 2016). We first select statements where such logical words appear. Then we select only 10K samples to start the continued pre-training.

Unlike the random mask language modeling pre-training objective, we make sure that the logical keywords (SLW) are masked and predicted by the model during the pre-training. We believe doing this will force the model to understand the correlation of these logical words with other tokens in a paragraph.

However, only predicting the SLWs would deviate from the model's objective of learning rich tokens for each token. So along with SLWs we also select randomly some percentage of tokens and ask the model to predict. We consider doubling the number of masking that is done in BERT (Devlin *et al.*, 2019b). We believe increasing the number of tokens to be predicted helps the model to correlate the representation of SLWs with many non-logical words (NOL). We consider this pre-training approach as Logical *Mask Language Modeling (LMLM).*

8.2.3 Fine-Tuning

We took the best performing LMLM model and train on the LogiQA training dataset with an additional layer on top of it for the task of multiple choice Question Answering. During training and inference, we append the dataset name in the prefix and concatenate it with the context and each of the answer options. We then use the fine-tuned model for inference.



P1: David, Jack and Mark are colleagues in a company. David supervises Jack, and Jack supervises Mark. David gets more salary than Jack.

Q: *What can be inferred from the above statements?*
A. Jack gets more salary than Mark.
B. David gets the same salary as Mark.
C. One employee supervises another who gets more salary than himself.
✓ **D. One employee supervises another who gets less salary than himself.**

P2: Our factory has multiple dormitory areas and workshops. None of the employees who live in dormitory area A are textile workers. We conclude that some employees working in workshop B do not live in dormitory area A.

Q: *What may be the missing premise of the above argument?*
A. Some textile workers do not work in workshop B.
B. Some employees working in workshop B are not textile workers.
✓ **C. Some textile workers work in workshop B.**
D. Some employees living in dormitory area A work in the workshop B.

Figure 8.2: LogiQA Datasets - Two Examples (Liu *et al.*, 2020a)

8.3 Datasets

For this project, we worked on a logical reasoning dataset LogiQA (Liu *et al.*, 2020a). LogiQA is a multiple choice question answering (MCQ) dataset where a context of a situation is provided and a question is asked. The task is to choose the correct answer for four answer options. Two examples can be seen in Figure 8.2. This dataset has around 7K training data, and 651 testing examples and is built on Civil Servants Examination Questions. This dataset is specially introduced to test the language model's ability to perform logical reasoning.

8.4 Experimental Results

8.4.1 Settings

We consider RoBERTa-Large as the base model for our project. We pre-train the model for 50 epochs with a per GPU batch size of 16 and block size of 512 on 4 GPUs.

During fine-tuning, we train the LMLM RoBERTa model for 50 epochs as well, with a batch size of 8. All our experiments ran on 4 x NVIDIA A-100 GPUs in Pytorch version 1.9 and CUDA 11.3. We use Huggingface transformers repository for the base version of our codes.

8.4.2 Metrics

We use the exact match accuracy metric in percentage (%) for the evaluation.

8.4.3 Baselines

The **Random Baseline** is created by selecting randomly one choice among the four for each of the questions in the testing set. We consider the **RoBERTa**-large model as our baseline. We also keep another baseline **DAGN** (Huang *et al.*, 2021). In this approach, the authors use discourse-aware graph networks for logical reasoning. They improved this model using augmented graph features in **DAGN (Aug)**.

8.4.4 Results

The experimental results are presented in Table 8.1. It can be seen that our approach along with RoBERTa-large significantly improves over the vanilla RoBERTa models by around 4.8%. It outperformed all other baseline approaches using continued pre-training on additional 10K paragraphs from Wikipedia.

8.4.5 Ablation Studies & Discussions

Table 8.2 shows an ablation study on varying the number of non-logical tokens that we masked during the pre-training. We pre-trained the RoBERTa model using just 10K Wikipedia samples. We see from the table that using only logical words (SLW) reduces (by ~3%) the performance of RoBERTa model. We hypothesize that

Model	Accuracy (%)
Random Baseline	25.0
RoBERTa	35.3
DAGN	38.7
DAGN (Aug)	39.3
DeBERTa-v3	40.1
RoBERTa + LMLM (Ours)	40.1 Δ+4.8
RoBERTa + LMLM + RACE-Middle (FT)	40.9 Δ+5.6
RoBERTa + LMLM + RACE-All (FT)	42.4 Δ+7.1

Table 8.1: LogiQA Performance (Accuracy). Performance Improvement Shown is over the RoBERTa Model

Model	Accuracy (%)
RoBERTa	35.3
RoBERTa + SLW	32.4 Δ -2.9
RoBERTa + SLW + 15% NOL	37.2 Δ+1.9
RoBERTa + SLW + 30% NOL	40.1 Δ+4.8
RoBERTa + MLM (30% RANDOM)	37.2 Δ+1.9

Table 8.2: LogiQA Performance (Accuracy) with our LMLM approach

when we do not mask other tokens for the LMLM, the model learns only to learn the prediction of these tokens and not how much this token affects other tokens. We also find that when we doubled the number of non-logical tokens masking then the performance improvement is significant (~5%). This shows that the representations are built for all those tokens considering the co-relation with the SLW tokens.

8.5 Related Works

The recent large language models like GPT-3 (Brown *et al.*, 2020a), PaLM (Chowdhery *et al.*, 2022), and UnifiedQA (Khashabi *et al.*, 2020) has shown remarkable performance on a diverse set of general natural language understanding tasks. There have been efforts on studying various kinds of reasoning skills like linguistic reasoning (Kumar *et al.*, 2019), multi-hop reasoning (Yang *et al.*, 2018b; Yu *et al.*, 2020b), numerical reasoning (Amini *et al.*, 2019; Dua *et al.*, 2019a; Ran *et al.*, 2019; Pal and Baral, 2021) and commonsense reasoning (Bhagavatula *et al.*, 2019; Lin *et al.*, 2019b; Sakaguchi *et al.*, 2021). While all these types of reasoning have been extensively studied, logical reasoning with transformer-based models has been understudied at present times.

Logical reasoning is an important reasoning skill because it plays a crucial role in many domains such as math, science, and law. The earliest effort of logical reasoning mainly focused on designing formal logic languages to represent rules and knowledge and develop automated theorem provers (Lifschitz, 2019). However, such approaches require expert knowledge (e.g. syntax and semantics of the formal logic) to compose the rules manually. As a result, recent efforts gradually shifted to tackling logical reasoning tasks (Tian *et al.*, 2021; Han *et al.*, 2022; Clark *et al.*, 2020) with pre-trained language models by creating natural logical reasoning benchmarks. In addition, some recent work in this area has explored evaluating robust deductive reasoners on simple perturbations like logical equivalences (Sanyal *et al.*, 2022a), creating logically consistent adversarial attacks (Gaskell *et al.*, 2022), and exploring some pre-training objectives to improve model's ability to reason logically (Sanyal *et al.*, 2022b).

There have been prior approaches that performed logical reasoning using specially designed models like LRReasoner (Wang *et al.*, 2022b) which parses symbolic logical structures for logical reasoning question-answering datasets. ReClor (Yu *et al.*, 2020c) uses data augmentation using logical context extensions. Ouyang et al. (Ouyang *et al.*, 2021) constructed logical graphs using the chain of facts present in a task instance and used GNNs to reason on the graph. Jiao et al. (Jiao *et al.*, 2022) proposed MERIt, which used Wikipedia to generate sentence pairs for contrastive learning that are logically related, and trained the PLM using contrastive loss. However, these methods make pre-training specific to downstream tasks rather than focusing on generalization.

Recently, to overcome this limitation, Sanyal et al. (Sanyal *et al.*, 2022b) has proposed a logical keyword-based pre-training approach for enabling logical reasoning skills in large language models.

8.6 Future Works

The approach can be improved considering the problem from multiple approaches.

Increasing the Pre-Training Data: We have pre-trained with specially selected 10K paragraphs from Wikipedia which provided us with some significant improvements. But there are 25GB of Wikipedia texts available to us. Training the model on such text is likely to improve its reasoning performance further.

Increasing the SLW: We created a dictionary of logical keywords. We can use some automated approaches to extract more such specific keywords.

Including Multi-Worded Logical Keywords: We have used only the single logical words and masked them for prediction. However, there are multi-worded logical keywords like *'because of', 'the reason is', 'as a result of', 'due to', 'as evident in','justified by', 'after all', 'on account of', 'on the grounds', 'there from', 'there upon', 'to that end', "don't", 'leading to', 'resulting in', 'and so', 'for this reason', 'called into question by','undermined by', 'not all'.* We believe masking and predicting them would help the model understand logical reasoning even better.

8.7 Conclusion

In this work, we present a continued pre-training approach of a transformer encoder-based language model where we masked logical words along with some percentage of non-logical words for prediction. This logical mask language modeling (LMLM) approach showed significant improvement over the vanilla RoBERTa model in fine-tuning with LogiQA, a logical reasoning dataset. We also propose several ways this work can be improved and logical reasoning skills can be embedded in the language models.

Chapter 9

INVESTIGATING NUMERACY LEARNING ABILITY OF A TEXT-TO-TEXT TRANSFER MODEL

ABSTRACT

The transformer-based pre-trained language models have been tremendously successful in most of the conventional NLP tasks. But they often struggle in those tasks where numerical understanding is required. Some possible reasons can be the tokenizers and pre-training objectives which are not specifically designed to learn and preserve numeracy. Here we investigate the ability of the text-to-text transfer learning model (T5), which has outperformed its predecessors in the conventional NLP tasks, to learn numeracy. We consider four numeracy tasks: numeration, magnitude order prediction, finding minimum and maximum in a series, and sorting. We find that, although T5 models perform reasonably well in the interpolation setting, they struggle considerably in the extrapolation setting across all four tasks.

9.1 Introduction

Recent advances in transfer learning in NLP have led to the emergence of pre-trained models which show a much stronger contextual representation of words than earlier static word embeddings. They have all performed extremely well in conventional NLP tasks. Yet, they fail to capture a better understanding of numbers. Numbers are an integral part of natural language texts which can change the meaning of a sentence. So there is a need for NLP models which can identify numbers represented in any surface forms like words, floats, or strings (Numeration), understand their values in various contexts (Magnitude Order Prediction), compare their values with others (List-MinMax) or able to rearrange a series of numbers based on its values (Sorting).

The transfer-learned models are pre-trained on a huge amount of natural language texts with specially designed tasks and tokenizers to create stronger word embeddings. This causes the numbers embedded in the texts to lose their meaning and inherent rules of numeracy guiding them (Thawani *et al.*, 2021; Nogueira *et al.*, 2021). This is possibly the reason they perform worse in numerical reasoning tasks on numbers absent in training data (Nogueira *et al.*, 2021; Wallace *et al.*, 2019).

In this paper, we test the numeracy learning ability of a text-to-text transfer learning generative model, T5 (Raffel *et al.*, 2020) which has outperformed its predecessors in conventional NLP tasks. The text-to-text format of input and output helps the model to generalize all the NLP tasks as a unified model. We use four numeracy tests both in interpolation (training and testing on the same range of data) and extrapolation settings (training on lower and testing on the higher range of data) and study how much numeracy skill it can acquire. Figure 9.1 shows some examples of each of the numeracy tests.

Our contributions in this paper are (1) an Extensive study on three versions of T5

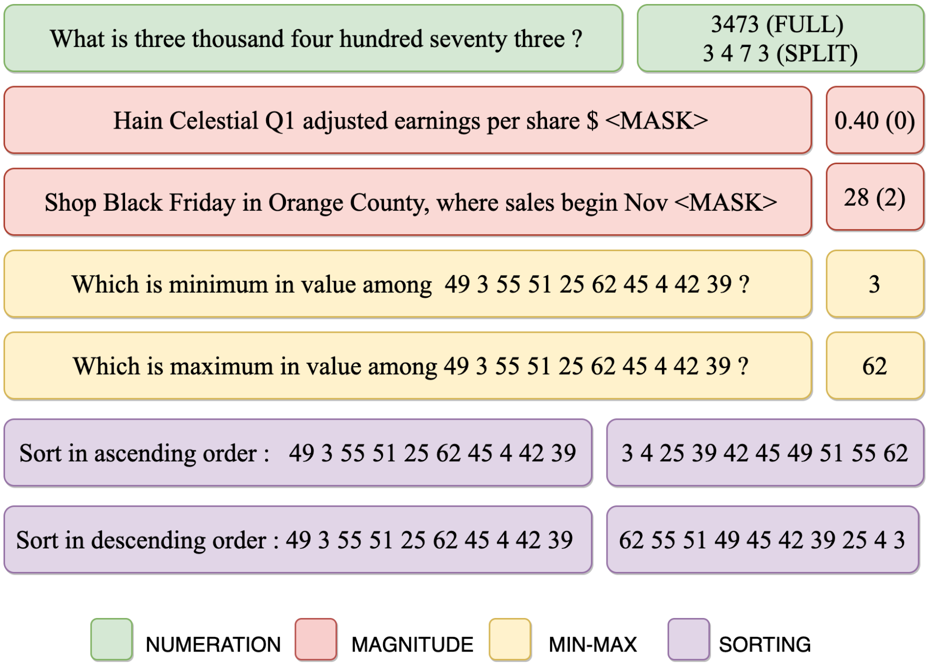

What is three thousand four hundred seventy three ?
3473 (FULL)
3 4 7 3 (SPLIT)
Hain Celestial Q1 adjusted earnings per share $ <MASK>
0.40 (0)
Shop Black Friday in Orange County, where sales begin Nov <MASK>
28 (2)
Which is minimum in value among 49 3 55 51 25 62 45 4 42 39 ?
3
Which is maximum in value among 49 3 55 51 25 62 45 4 42 39 ?
62
Sort in ascending order : 49 3 55 51 25 62 45 4 42 39
3 4 25 39 42 45 49 51 55 62
Sort in descending order : 49 3 55 51 25 62 45 4 42 39
62 55 51 49 45 42 39 25 4 3
NUMERATION
MAGNITUDE
MIN-MAX
SORTING

Figure 9.1: Examples of Numeracy Tests

models (small, base, large) on four numeracy tests in interpolation and extrapolation settings. (2) Reporting interesting observations in the behavior of each model version across multiple experimental settings through detailed manual error analysis. The synthetically generated data and codes are publicly available[1] for future numeracy analysis in similar settings.

9.2 Numeracy Tests

We perform four essential numeracy tests to explore the model's ability to understand numerical values.

Motivation: These four elementary tasks are simple and easy for the models since they do not need to generate a completely new number in a different numerical range (like in mathematical tests: multiplication, division, exponentiation). Here we evaluate

[1] https://github.com/kuntalkumarpal/T5Numeracy

whether the models learn the numeracy tasks or they simply learn bias from the number range seen in training data.

9.2.1 Numeration

The probability of a number represented in multiple surface forms (word, scientific, float, integer) increases with the increase in the volume of the pre-training corpus of the language models. It is impractical for an end-to-end NLP model to semantically parse these numbers accurately and convert them into a single representation to retain their value or reason with. This task tests the model's ability to understand the word representation of a number and to decode it into integer form.

9.2.2 Magnitude Order Prediction

The task is to identify the order of magnitude of a missing (masked) number that fits the context of a natural language text. This task is important in numerical commonsense reasoning (Lin *et al.*, 2020) and prompt-based methods (Liu *et al.*, 2021). Here, we do not expect the model to predict the exact number that fits the context as this may vary in different domains. Instead, this task tests the model's ability to understand a missing number's context and predict its appropriate range.

9.2.3 List-MinMax

We test the model's ability to understand numerical values and compare them. Given a series of n positive numbers, the task is to find the minimum and the maximum number. This is the basis of many question-answering and commonsense numerical reasoning datasets like SQuAD (Rajpurkar *et al.*, 2016), DROP (Dua *et al.*, 2019) and NUMBERGAME (Mishra *et al.*, 2020). We simplify the task by generating templates so that the models can concentrate on understanding the task rather than getting confused by the language complexities.

9.2.4 Sorting

In addition to understanding the values of each number in a series, the model will have to rearrange them in the correct order through this task, making it even harder than List-MinMax. Even if a model is successful in the previous test, it is necessary to identify whether it has actually been compared among all the numbers in the series. Hence, sorting a list of n numbers in ascending and descending orders ensures that the model compares all the numbers and rearranges them into two different sequences.

9.3 Experiments

9.3.1 Experimental Setup

We use T5-SM (small, 60M parameters), T5-BS (base, 220M), T5-LG (large, 770M), and positive integers for the experiments. The results are average of three random seeds. We perform experiments in two settings: *interpolation* (training and testing on the same numerical range) and *extrapolation* (training on lower and testing on the higher numerical range). The latter helps us to analyze whether a model has learned the task, or it has exploited bias in the numerical range of the training data.

9.3.2 Data Preparation

Numeration: We create a dataset keeping in mind that at least a few examples of all unique words needed to represent each number are present in the training data (Trask *et al.*, 2018). In Table 9.1, interpolation samples are from [0,10K) and 99K extrapolation samples are from [10K,1000K). We use *num2words*[2] for generating

[2] https://github.com/savoirfairelinux/num2words

# TRAIN →		4.9K		1.3K		0.9K	
TP	**Model**	**IN**	**EX**	**IN**	**EX**	**IN**	**EX**
FL	T5-SM	45.31	0.08	1.90	0.01	0.33	0.00
	T5-BS	92.16	1.03	66.47	0.45	37.20	0.42
	T5-LG	98.06	1.91	89.49	1.96	79.48	1.58
SP	T5-SM	69.67	39.35	26.89	1.10	0.23	0.01
	T5-BS	99.50	11.31	81.21	22.44	73.61	31.06
	T5-LG	100.00	10.05	99.97	7.35	91.59	12.92

Table 9.1: Numeration EM Scores w/ Split (SP) And w/o Split (FL) Representation on 4.9K, 1.3K, 0.9K Train-data In Interpolation (IN) And Extrapolation (EX) Settings.

		LIST MINIMUM						LIST MAXIMUM					
# ELEMENTS		**3**		**5**		**10**		**3**		**5**		**10**	
Range	**Model**	**IN**	**EX**	**IN**	**EX**	**IN**	**EX**	**IN**	**EX**	**IN**	**EX**	**IN**	**EX**
< 99	T5-SM	90.5	0.6	86.5	0.1	65.9	0.0	80.4	0.5	71.6	0.3	74.7	0.1
	T5-BS	96.2	33.9	99.1	13.0	98.2	2.8	92.3	22.7	96.8	6.0	90.4	1.1
	T5-LG	100.0	22.2	99.4	2.8	100.0	0.5	100.0	29.6	100.0	13.6	100.0	2.0
< 999	T5-SM	72.6	41.8	55.5	22.2	49.9	9.7	65.3	38.4	54.8	17.5	40.0	5.2
	T5-BS	91.5	67.2	92.1	42.6	80.4	27.1	89.1	65.3	90.8	47.2	88.3	25.0
	T5-LG	98.3	70.1	96.1	49.3	87.4	34.7	96.1	61.2	97.8	58.7	95.2	35.3
< 9999	T5-SM	59.1	44.7	43.5	30.4	30.7	17.1	51.2	47.0	36.0	27.0	20.9	11.1
	T5-BS	89.6	68.8	86.9	53.8	85.4	38.1	87.1	58.6	83.1	43.4	81.6	29.9
	T5-LG	97.1	81.3	93.7	71.8	94.0	58.2	96.2	84.9	94.9	76.4	94.9	59.1

Table 9.2: List-MinMax (Series Length: 3, 5, 10) In Three Different Number Ranges Evaluated as Interpolation (IN) and Extrapolation (EX) Exact-match Scores on 1k Test Data.

Datasets →	**AT**		**MC**	
Models ↓	μ**F1**	m**F1**	μ**F1**	m**F1**
LR	62.49	30.81	71.25	60.80
CNN	69.27	35.96	77.17	58.49
GRU	70.92	38.43	78.25	58.08
BiGRU	71.49	39.94	<u>80.16</u>	62.74
CRNN	69.50	36.15	78.00	<u>64.62</u>
CNN-capsule	63.11	29.41	75.89	59.22
GRU-capsule	70.73	33.57	77.36	**64.71**
BiGRU-capsule	71.49	34.18	77.97	64.34
BiLSTM-DICE	75.56	**46.80**	-	-
T5-SM	69.87	31.36	66.11	34.68
T5-BS	<u>78.06</u>	40.04	72.22	47.44
T5-LG	**81.40**	<u>44.64</u>	**80.29**	59.16

Table 9.3: Magnitude Order Prediction for Market Comments (MC) and Article Titles (AT) Datasets of Numeracy600k in Micro-f1 (μF1) and Macro-f1 (mF1). The Best Score Is in Bold and the Second Best Is Underlined.

the word form of each integer. To simulate fewer shot settings, we carefully craft two smaller training sets taking only 20% and 10% data. We show two number representation schemes with split-digits (SP) and without split (FL) hypothesizing that for a generative model, it would be easier to correctly generate individual digits instead of full integers at once.

We have 4906, 2097, and 2997 in train, dev, and test respectively. We make sure that all numbers within 10K are present in any of the train, dev or test. For extrapolation, we select 1K integers randomly from every 10K range from [10K,1000K) making it a total of 99K.

Magnitude Order Prediction: For this task we work on Numeracy600K (Chen

Train on →	**AT**		**MC**	
Models ↓	μ**F1**	m**F1**	μ**F1**	m**F1**
BiGRU	25.59	10.58	31.38	11.08
T5-SM	28.88	12.04	37.35	10.81
T5-BS	35.53	14.48	31.51	12.25
T5-LG	**50.18**	**21.24**	**38.43**	**12.32**

Table 9.4: Cross Domain (Extrapolation) Tests of Order Prediction. Train on MC, Test on AT, and Vice-versa.

et al., 2019) dataset. We consider this as a mask prediction task. We train models to find the exact number that fits the mask. Then, we map the predicted numbers into their magnitude order, save the model based on the best magnitude order and calculate the evaluation metrics on test data. Since this is a generation task we reject those answers which are not valid floating point numbers. The baseline results in Table 9.3 are from (Chen *et al.*, 2019; Sundararaman *et al.*, 2020). We also consider the extrapolation setting by showing the cross-domain performance (train on market comments and test on the article title and vice-versa) in Table 9.4.

For this data, we consider 450K, 50K, and 100K samples for train, dev, and test data respectively from each of the market comments and article titles data.

List Min-Max & Sort: We experiment on three different number ranges: [0,100), [0,1K), [0,10K). For interpolation tests, the numbers in the test data are from the same ranges. The extrapolation numbers are from the maximum of respective ranges to 100K. To prevent the model's bias on number lengths, we bring them closer following prior work (Wallace *et al.*, 2019). We extend the experiment on a series of 3, 5, and 10 numbers (for each range) to study how each of the models behaves with increasing series length. We consider the same data for sorting experiments as well. The results

		LIST-SORT ASCENDING						LIST-SORT DESCENDING					
# ELEMENTS		3		5		10		3		5		10	
Range	Model	IN	EX	IN	EX	IN	EX	IN	EX	IN	EX	IN	EX
< 99	T5-SM	54.0	12.4	7.6	0.0	0.0	0.0	56.0	12.6	5.9	0.4	0.0	0.0
	T5-BS	80.6	12.2	87.2	0.0	0.4	0.0	84.3	12.9	75.5	0.0	6.2	0.0
	T5-LG	100.0	5.8	99.9	0.0	69.7	0.1	100.0	13.1	96.6	0.1	57.6	0.1
< 999	T5-SM	32.6	15.1	1.4	0.6	0.0	0.0	38.0	22.3	3.4	1.3	0.0	0.0
	T5-BS	74.7	45.7	64.0	8.0	12.5	0.0	73.1	42.0	62.6	9.6	16.8	0.1
	T5-LG	95.1	64.2	91.8	16.8	61.9	1.7	94.7	63.5	92.5	25.7	61.2	1.6
< 9999	T5-SM	23.4	17.1	1.0	0.1	0.0	0.0	30.4	21.2	0.7	0.4	0.0	0.0
	T5-BS	63.1	45.5	51.1	12.7	15.0	0.2	59.8	43.9	51.4	12.4	14.3	0.3
	T5-LG	94.5	76.0	87.4	43.2	74.6	12.6	94.2	76.1	86.1	44.4	75.6	11.9

Table 9.5: List-sort (Ascending & Descending) on Series Lengths: 3, 5, 10 in Three Different Integer Ranges Evaluated as Interpolation (IN) and Extrapolation (EX) Exact-match Scores on 1k Test Data.

are in Table 9.2 for List-MinMax and Table 9.5 for List-Sort.

We consider both the task of arranging in ascending and descending orders since if a series is already sorted in ascending order the model can directly predict by copying it from the given input.

9.3.3 Hyperparameters

For all the experiments we use a maximum sequence length of 128 and 256 for question context. The maximum sequence length of the answers is kept as [5, 10, 20, 25] for different tasks. We ran for 20 epochs and save a model based on validation EM performance. Our training and validation batch size varies between [2, 4, 8, 16, 32] based on the experiment. We work on 4 Tesla V100 GPUs. We use AdamW optimizer and StepLR scheduler with a step size of 2, a learning rate of 5e-5, and a gamma of

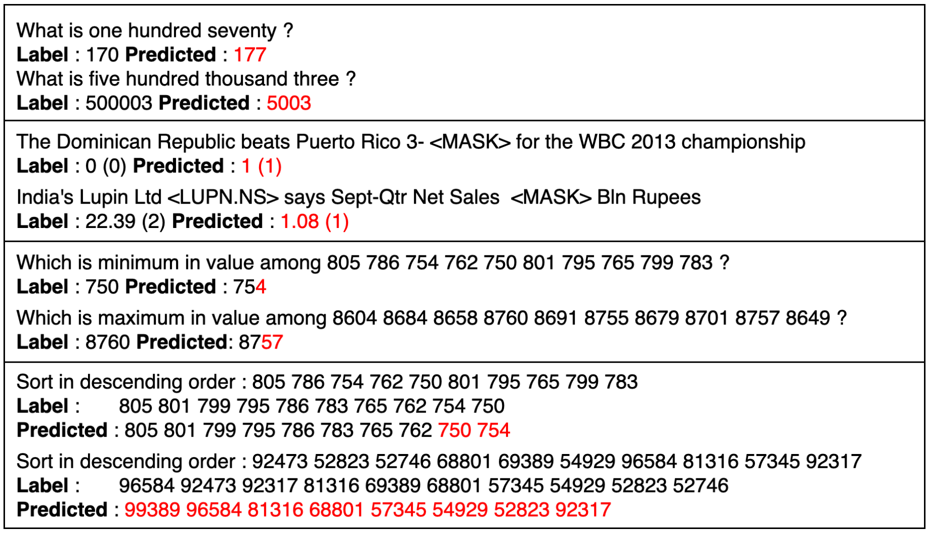
0.1.

9.4 Results and Error Analysis

Table 9.1 shows all versions of T5 benefit when they are trained with split representation. When trained with 4.9K data, T5-SM gains 24% points in interpolation evaluation whereas T5-LG gains only 2%. None of the models perform well on unseen number data ranges. In fewer shot interpolation settings, however, only the T5-LG model maintains its performance beyond 90% which is not surprising because of its large parameter space. We noticed that the best model could only partially decode numbers having multiple zeros (Figure 9.2). In the first example, the model predicts an extra seven and in the second (extrapolation), it ignored the keyword 'hundred' as it attempts to fit this unseen data into a similar seen number range (4 digits).

In magnitude order prediction (Table 9.3), T5-LG's performance improves by 5 μF1 in the article title. For extrapolation (Table 9.4), all T5 versions beats previous estimates (BiGRU) by at most 25%. This shows that T5 can learn robust numeric representations based on contexts. Both the samples in Fig 9.2 are hard as they need prior explicit knowledge. Yet they are able to predict numbers in similar feasible ranges. This shows that the model is not randomly assigning magnitude but has learned based on the domain and context. We found that the best T5 model predicted an order of 1 instead of 2 for market and article data making a maximum error of 39.07% and 33.59% respectively.

Table 9.2 shows List-MinMax results. Both T5-BS and T5-LG perform over 80% across all ranges and series lengths. T5-SM however, degrades in performance as the range increases along with the list size. As the model learns more variations in numbers, the extrapolation performance increases to a max of 81% (List-Min) and 84.9% (List-Max). But the performance drops as the series length increases. The best



What is one hundred seventy ?
Label : 170 **Predicted** : 177
What is five hundred thousand three ?
Label : 500003 **Predicted** : 5003

The Dominican Republic beats Puerto Rico 3- <MASK> for the WBC 2013 championship
Label : 0 (0) **Predicted** : 1 (1)
India's Lupin Ltd <LUPN.NS> says Sept-Qtr Net Sales <MASK> Bln Rupees
Label : 22.39 (2) **Predicted** : 1.08 (1)

Which is minimum in value among 805 786 754 762 750 801 795 765 799 783 ?
Label : 750 **Predicted** : 754
Which is maximum in value among 8604 8684 8658 8760 8691 8755 8679 8701 8757 8649 ?
Label : 8760 **Predicted**: 8757

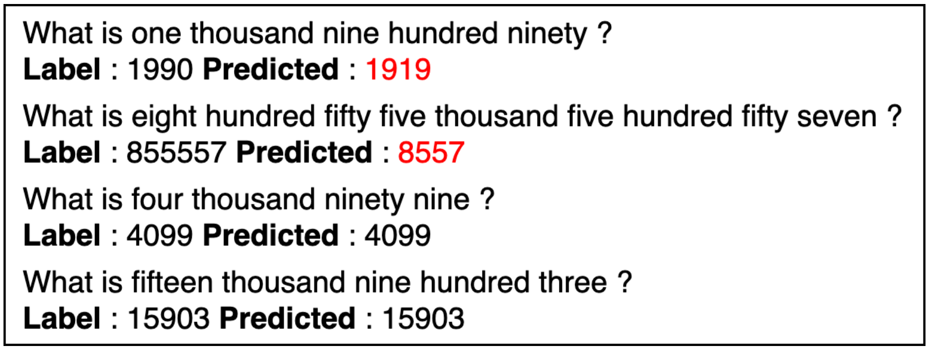
Sort in descending order : 805 786 754 762 750 801 795 765 799 783
Label : 805 801 799 795 786 783 765 762 754 750
Predicted : 805 801 799 795 786 783 765 762 750 754
Sort in descending order : 92473 52823 52746 68801 69389 54929 96584 81316 57345 92317
Label : 96584 92473 92317 81316 69389 68801 57345 54929 52823 52746
Predicted : 99389 96584 81316 68801 57345 54929 52823 92317

Figure 9.2: Two Incorrect Predictions For Each Task.

model predicted the second minimum and maximum element in the examples of Fig 9.2.

From the sorting results (Table 9.5), we see T5-SM performance drops (18-22% from 2-3 digits, 8-9% from 3-4) as number ranges increase across the series length of 3. T5-SM fails to generate a single correct order for a series of 10 elements and achieves less than 10% success in 5-element series across all ranges. This degrading performance can be attributed to its mere 60M parameter space. As the number of parameters keeps increasing the models perform consistently across each of the 3, 5, and 10 elements in series, both for interpolation and extrapolation settings. With the increasing range of training data, the models become more robust to extrapolated numbers across all series lengths with 8-30% change in ascending order and 7-20% change in descending order. Finally, for sorting, we find a variety of incorrect predictions: missing order of one element, omission of one and two elements, or repeating a particular element.

Overall, none of the models were able to perform well on extrapolation samples showing the inherent rules of numeracy are difficult for these models to learn. But,



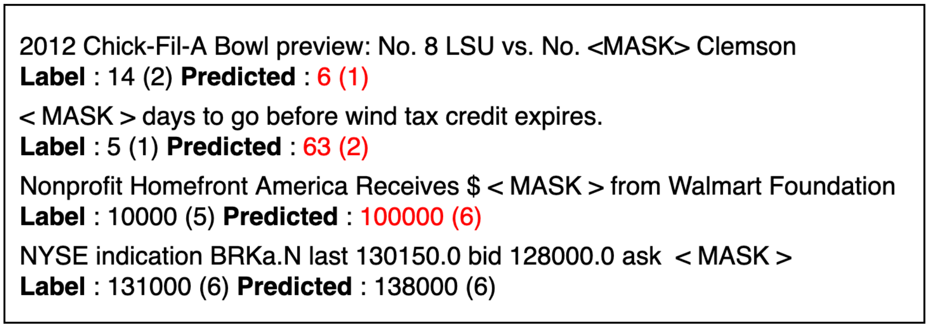
What is one thousand nine hundred ninety ?
Label : 1990 **Predicted** : 1919

What is eight hundred fifty five thousand five hundred fifty seven ?
Label : 855557 **Predicted** : 8557

What is four thousand ninety nine ?
Label : 4099 **Predicted** : 4099

What is fifteen thousand nine hundred three ?
Label : 15903 **Predicted** : 15903

Figure 9.3: Some Predictions For Numeration Task.



2012 Chick-Fil-A Bowl preview: No. 8 LSU vs. No. <MASK> Clemson
Label : 14 (2) **Predicted** : 6 (1)

< MASK > days to go before wind tax credit expires.
Label : 5 (1) **Predicted** : 63 (2)

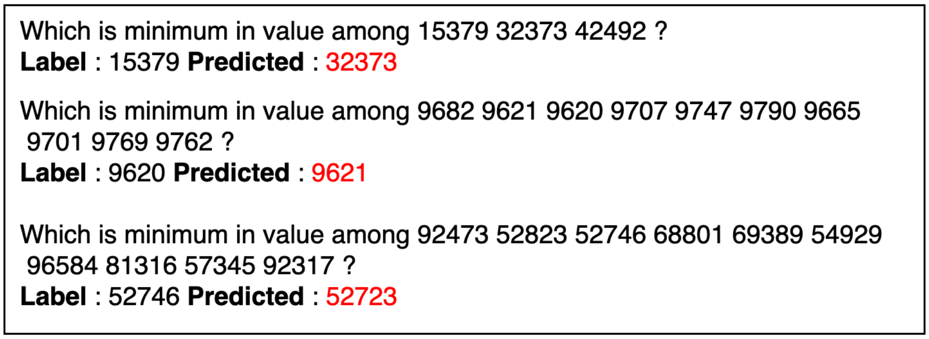
Nonprofit Homefront America Receives $ < MASK > from Walmart Foundation
Label : 10000 (5) **Predicted** : 100000 (6)

NYSE indication BRKa.N last 130150.0 bid 128000.0 ask < MASK >
Label : 131000 (6) **Predicted** : 138000 (6)

Figure 9.4: Magnitude Order Prediction Examples.

it also shows, more variations in numbers (increasing the range) help them perform better in extrapolation settings. The smaller model's limited parameter space affects its performance in all four tasks whereas larger models are able to pick up some numeracy skills through training. We show more predictions in Figure 9.3, 9.4, 9.5, 9.6.

Analysis of NT5: We test with the NT5 (Yang *et al.*, 2021) model on all our experiments and compared the results with T5-small. For the Numeration task with the split number representation, NT5 performed 73.07 (accuracy), a 4% improvement over T5. The performance however did not improve for the MinMax and Sorting tasks. For 3-element sorting, it dropped by 10-20%. In the Magnitude Order Prediction, we find the cross-domain (extrapolation) μF1 score increases by 5-7% while the in-domain

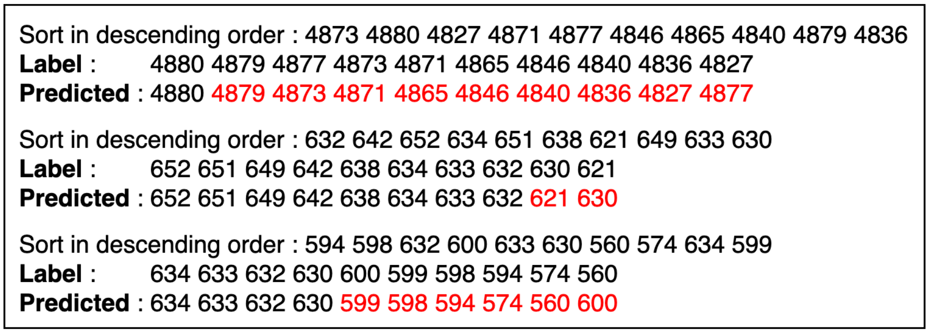


Which is minimum in value among 15379 32373 42492 ?
Label : 15379 **Predicted** : 32373

Which is minimum in value among 9682 9621 9620 9707 9747 9790 9665 9701 9769 9762 ?
Label : 9620 **Predicted** : 9621

Which is minimum in value among 92473 52823 52746 68801 69389 54929 96584 81316 57345 92317 ?
Label : 52746 **Predicted** : 52723

Figure 9.5: Some Predictions For List-MinMax Task.



Sort in descending order : 4873 4880 4827 4871 4877 4846 4865 4840 4879 4836
Label : 4880 4879 4877 4873 4871 4865 4846 4840 4836 4827
Predicted : 4880 4879 4873 4871 4865 4846 4840 4836 4827 4877

Sort in descending order : 632 642 652 634 651 638 621 649 633 630
Label : 652 651 649 642 638 634 633 632 630 621
Predicted : 652 651 649 642 638 634 633 632 621 630

Sort in descending order : 594 598 632 600 633 630 560 574 634 599
Label : 634 633 632 630 600 599 598 594 574 560
Predicted : 634 633 632 630 599 598 594 574 560 600

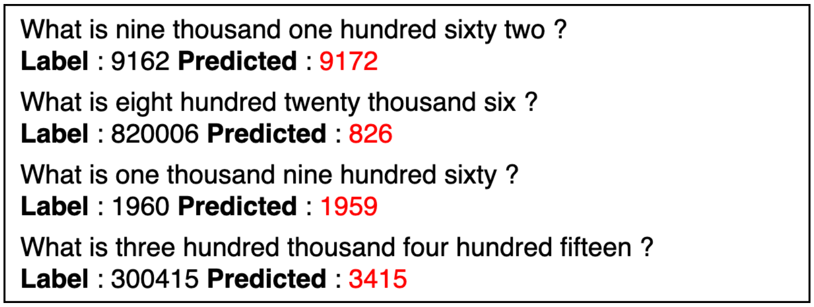
Figure 9.6: Some Predictions For List-Sort task.

decreases by 3-6%. This might be because NT5 has seen more variety of contexts of numbers and can generalize well on this task.

Zero-Shot Magnitude Order Prediction: We also experimented with zero-shot magnitude order predictions. We found 553 and 8783 exact matches out of 100K test data using T5-large which shows that the performance is very poor without proper fine-tuning. We show some more predictions of the best performing T5 model in Figure 9.7, 9.8, 9.9, 9.10.

9.5 Related Works

Numeracy Tests: Multiple numeracy tests have been proposed to evaluate the static word embeddings (Naik *et al.*, 2019) like GloVe, Word2Vec, FastText, and



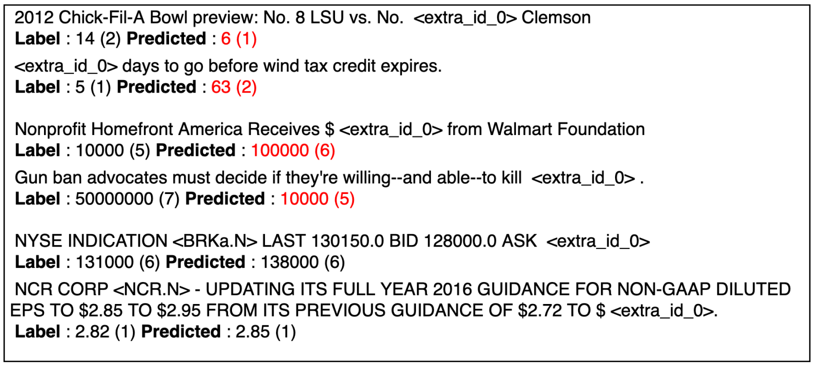
What is nine thousand one hundred sixty two ?
Label : 9162 **Predicted** : 9172

What is eight hundred twenty thousand six ?
Label : 820006 **Predicted** : 826

What is one thousand nine hundred sixty ?
Label : 1960 **Predicted** : 1959

What is three hundred thousand four hundred fifteen ?
Label : 300415 **Predicted** : 3415

Figure 9.7: More Predictions For Numeration Task.



2012 Chick-Fil-A Bowl preview: No. 8 LSU vs. No. <extra_id_0> Clemson
Label : 14 (2) **Predicted** : 6 (1)

<extra_id_0> days to go before wind tax credit expires.
Label : 5 (1) **Predicted** : 63 (2)

Nonprofit Homefront America Receives $ <extra_id_0> from Walmart Foundation
Label : 10000 (5) **Predicted** : 100000 (6)

Gun ban advocates must decide if they're willing--and able--to kill <extra_id_0> .
Label : 50000000 (7) **Predicted** : 10000 (5)

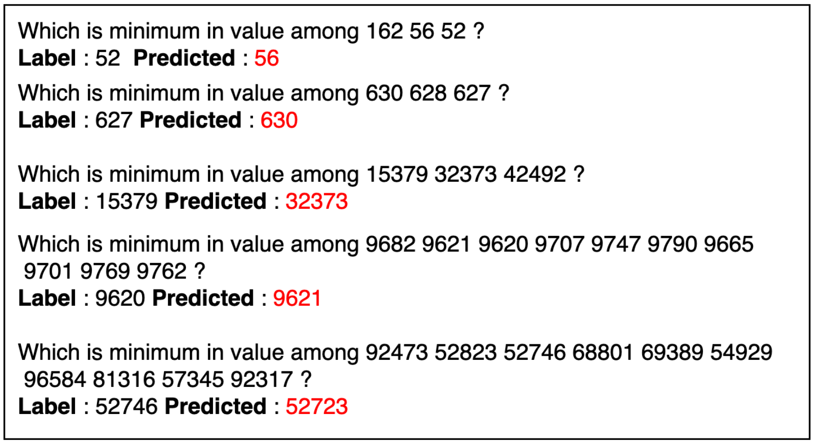
NYSE INDICATION <BRKa.N> LAST 130150.0 BID 128000.0 ASK <extra_id_0>
Label : 131000 (6) **Predicted** : 138000 (6)

NCR CORP <NCR.N> - UPDATING ITS FULL YEAR 2016 GUIDANCE FOR NON-GAAP DILUTED EPS TO $2.85 TO $2.95 FROM ITS PREVIOUS GUIDANCE OF $2.72 TO $ <extra_id_0>.
Label : 2.82 (1) **Predicted** : 2.85 (1)

Figure 9.8: More Magnitude Order Prediction Examples.

contextual embeddings (Wallace *et al.*, 2019) like BERT through probing tasks like numeration, magnitude comparison, addition, list-maximum. Multilingual numeration (Johnson *et al.*, 2020) tests have been performed by probing models like DistilBERT, XLM, and BERT. CNN, BiGRU models have been shown to perform well in magnitude order prediction (Chen *et al.*, 2019) and T5 on addition and subtraction tasks (Nogueira *et al.*, 2021) through training on similar texts. We, however, focus on studying how much text-to-text transfer models (T5) can learn across four fundamental numeracy tasks in samples containing both in-domain and out-of-domain numerical ranges.

Specially Designed Models: NALU (Trask *et al.*, 2018), NAU and NMU (Madsen and Johansen, 2020), numBERT (Zhang *et al.*, 2020), GenBERT (Geva *et al.*, 2020), NT5 (Yang *et al.*, 2021) have emerged in the last few years to incorporate arithmetic



Which is minimum in value among 162 56 52 ?
Label : 52 **Predicted** : 56

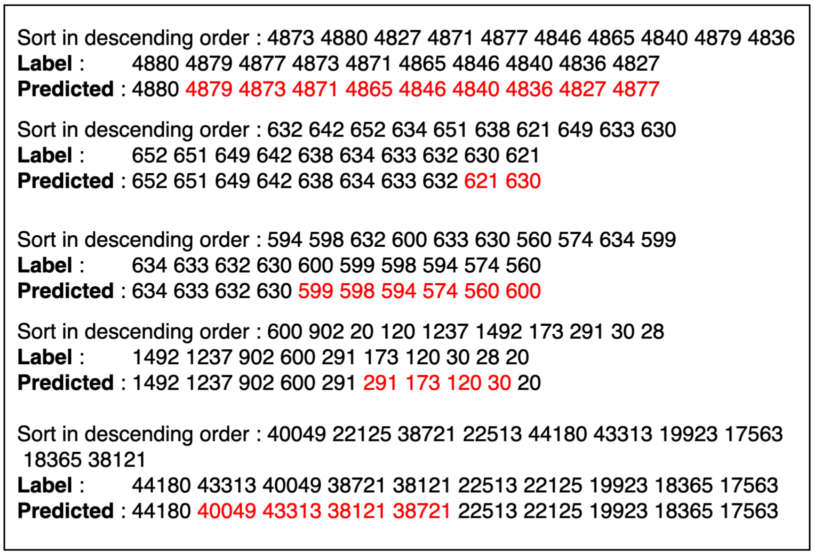
Which is minimum in value among 630 628 627 ?
Label : 627 **Predicted** : 630

Which is minimum in value among 15379 32373 42492 ?
Label : 15379 **Predicted** : 32373

Which is minimum in value among 9682 9621 9620 9707 9747 9790 9665 9701 9769 9762 ?
Label : 9620 **Predicted** : 9621

Which is minimum in value among 92473 52823 52746 68801 69389 54929 96584 81316 57345 92317 ?
Label : 52746 **Predicted** : 52723

Figure 9.9: More predictions For List-MinMax Task.



Sort in descending order : 4873 4880 4827 4871 4877 4846 4865 4840 4879 4836
Label : 4880 4879 4877 4873 4871 4865 4846 4840 4836 4827
Predicted : 4880 4879 4873 4871 4865 4846 4840 4836 4827 4877

Sort in descending order : 632 642 652 634 651 638 621 649 633 630
Label : 652 651 649 642 638 634 633 632 630 621
Predicted : 652 651 649 642 638 634 633 632 621 630

Sort in descending order : 594 598 632 600 633 630 560 574 634 599
Label : 634 633 632 630 600 599 598 594 574 560
Predicted : 634 633 632 630 599 598 594 574 560 600

Sort in descending order : 600 902 20 120 1237 1492 173 291 30 28
Label : 1492 1237 902 600 291 173 120 30 28 20
Predicted : 1492 1237 902 600 291 291 173 120 30 20

Sort in descending order : 40049 22125 38721 22513 44180 43313 19923 17563 18365 38121
Label : 44180 43313 40049 38721 38121 22513 22125 19923 18365 17563
Predicted : 44180 40049 43313 38121 38721 22513 22125 19923 18365 17563

Figure 9.10: More predictions for List-Sort task.

skills into models through specially designed architecture or fine-tuning tasks which improves the performance in synthetic arithmetic or crowd-sourced numerical reasoning tasks like DROP.

Numerical Embeddings: There are limited prior works in numeracy-aware embeddings that show good performance in extrapolation settings. One approach (Jiang *et al.*, 2019) represents numerals as a weighted average of prototype numeral embeddings obtained using either a self-organizing map or Gaussian Mixture models. DICE (Sundararaman *et al.*, 2020) is a deterministic numeral embedding approach, independent of the corpus, which preserves the relative magnitude between two numerals and their embeddings.

9.6 Conclusion & Future Works

We show that text-to-text models are able to learn numeracy quite well in an interpolation setting. Our extensive experiments show that T5 models struggle to learn with numbers outside training data ranges. We believe that to make further progress in transfer learning, models need to achieve such elementary numeracy skills and this gap between interpolation and extrapolation performance needs to be reduced. We are of the opinion that, adding more data would not bridge this gap since the domain of numbers is open. However, special pre-training objectives for digits rather than whole numbers can be designed to teach inherent numeracy to models. In the future, we intend to explore these objectives centered around preserving numeracy rules in transfer-learned models to generalize between in-domain and out-of-domain numbers.

REFERENCES

Chen, C.-C., H.-H. Huang, H. Takamura and H.-H. Chen, “Numeracy-600k: learning numeracy for detecting exaggerated information in market comments”, in “Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics”, pp. 6307–6313 (2019).

Dua, D., Y. Wang, P. Dasigi, G. Stanovsky, S. Singh and M. Gardner, “Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs”, arXiv preprint arXiv:1903.00161 (2019).

Geva, M., A. Gupta and J. Berant, “Injecting numerical reasoning skills into language models”, in “Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics”, pp. 946–958 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.acl-main.89`.

Jiang, C., Z. Nian, K. Guo, S. Chu, Y. Zhao, L. Shen and K. Tu, “Learning numeral embeddings”, arXiv preprint arXiv:2001.00003 (2019).

Johnson, D., D. Mak, A. Barker and L. Loessberg-Zahl, “Probing for multilingual numerical understanding in transformer-based language models”, in “Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP”, pp. 184–192 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.blackboxnlp-1.18`.

Lin, B. Y., S. Lee, R. Khanna and X. Ren, “Birds have four legs?! numersense: Probing numerical commonsense knowledge of pre-trained language models”, in “Proceedings of EMNLP”, (2020), to appear.

Liu, P., W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing”, (2021).

Madsen, A. and A. R. Johansen, “Neural arithmetic units”, arXiv preprint arXiv:2001.05016 (2020).

Mishra, S., A. Mitra, N. Varshney, B. Sachdeva and C. Baral, “Towards question format independent numerical reasoning: A set of prerequisite tasks”, arXiv preprint arXiv:2005.08516 (2020).

Naik, A., A. Ravichander, C. Rose and E. Hovy, “Exploring numeracy in word embeddings”, in “Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics”, pp. 3374–3380 (2019).

Nogueira, R., Z. Jiang and J. Li, “Investigating the limitations of the transformers with simple arithmetic tasks”, arXiv preprint arXiv:2102.13019 (2021).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", Journal of Machine Learning Research **21**, 140, 1–67, URL http://jmlr.org/papers/v21/20-074.html (2020).

Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text", in "Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing", pp. 2383–2392 (Association for Computational Linguistics, Austin, Texas, 2016), URL https://aclanthology.org/D16-1264.

Sundararaman, D., S. Si, V. Subramanian, G. Wang, D. Hazarika and L. Carin, "Methods for numeracy-preserving word embeddings", in "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)", pp. 4742–4753 (2020).

Thawani, A., J. Pujara, P. A. Szekely and F. Ilievski, "Representing numbers in nlp: a survey and a vision", arXiv preprint arXiv:2103.13136 (2021).

Trask, A., F. Hill, S. E. Reed, J. Rae, C. Dyer and P. Blunsom, "Neural arithmetic logic units", 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., 2018), URL https://proceedings.neurips.cc/paper/2018/file/0e64a7b00c83e3d22ce6b3acf2c582b6-Paper.pdf.

Wallace, E., Y. Wang, S. Li, S. Singh and M. Gardner, "Do NLP models know numbers? probing numeracy in embeddings", 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)", pp. 5307–5315 (Association for Computational Linguistics, Hong Kong, China, 2019), URL https://www.aclweb.org/anthology/D19-1534.

Yang, P.-J., Y. T. Chen, Y. Chen and D. Cer, "Nt5?! training t5 to perform numerical reasoning", arXiv preprint arXiv:2104.07307 (2021).

Zhang, X., D. Ramachandran, I. Tenney, Y. Elazar and D. Roth, "Do language embeddings capture scales?", in "Findings of the Association for Computational Linguistics: EMNLP 2020", pp. 4889–4896 (Association for Computational Linguistics, Online, 2020), URL https://www.aclweb.org/anthology/2020.findings-emnlp.439.

Chapter 10

CONCLUSION

In this dissertation, I presented various projects, not only in the natural language domain but also in resource-limited Cybersecurity and Biomedical domains, to understand the role of knowledge in improving the transformer-based language models. First, I demonstrated through Knowledge Extraction (KX), how LMs can be used effectively to extract specific explicit, implicit, and structured knowledge from data which improves our understanding of the data by helping us to gain insights and take informed decisions. Then through Knowledge Integration (KI), I presented methods to show how LMs can incorporate targeted external knowledge to improve their reasoning skills (multi-step, commonsense, and logical). Finally, through Knowledge evaluation (KE), I demonstrated how LM's knowledge can be assessed and their limitations can be learned. Thus knowledge takes various forms from an output in KX to an input in KI to finally a evaluator to interact with transformer-based language models.

10.1 Knowledge Extraction (KX)

Chapters 2, 3, and 4 demonstrated how LMs can extract knowledge effectively. Chapter 2 showed how LMs can be applied in resource-limited biomedical domains to extract entities and assign their types in a multi-task setting. In this project, I develop a novel knowledge-guided named entity recognition approach in a QA setting and improved the state-of-the-art performance of 12 out of 18 publicly available biomedical datasets. In Chapter 3, I introduced a dataset for the data-limited Cybersecurity domain to analyze structured instruction flow in public forum documents. Then I establish a benchmark on an information flow-graph prediction task. Finally, I demonstrate that the approaches also work on two datasets of natural language domains. In Chapter 4, I showed how missing textual information like variable names can be recovered from decompiled binaries using both pre-existing datasets and our improved dataset. The chapter showed that the approach significantly outperforms existing approaches in two decompilers.

In this direction, I would also like to mention two of my collaborative efforts with other researchers. In the first project (Kashihara *et al.*, 2023), I collaborated in *predicting thread structure from unstructured public forum conversations* in the cybersecurity domain which is relevant to Chapter 3. In this project, we perform better social network prediction based on forum interactions using Instruction Prompting-based Next Paragraph Prediction (NPP-IP) method, outperforming the existing thread-structuring methods on two datasets - Reddit and Hacker Forums. In the second collaborative effort (Agrawal *et al.*, 2023), I contributed to *developing an Ontology for the Cybersecurity education domain for students or novice researchers*. Such an ontology can be used to develop AI-based automated Cybersecurity education systems that can improve cognitive engagement and active learning. We also introduced AISecKG, a triple dataset with cybersecurity-related entities and relations as defined by the ontology. We show a downstream application of our high-quality NER dataset by extracting malicious named entities using BERT and RoBERTa models.

10.2 Knowledge Integration (KI)

Chapter 5 showed how multihop reasoning skills can be incorporated into language models by using external knowledge extracted from an open book and missing knowledge from relevant knowledge sources. The approach showed significant performance improvement over the existing approaches on Open-Book QA dataset. Chapter 6 demonstrated four approaches by which external instance-specific knowledge can be incorporated into transformer-based language models. The integration of knowledge can help language models to improve their commonsense reasoning abilities about the physical interaction of humans with objects. In Chapter 7, I showed that training a generative LM on the diverse nature of cybersecurity texts and tasks in a multi-task setting can help it to adapt to unseen domains and tasks. In Chapter 8, I also presented that by continued pre-training, logical reasoning skills can be incorporated into language models. I showed that such an approach can achieve good performance over vanilla language models on logical MCQ datasets.

In this direction, I would also like to mention a project where I collaborated. In this project, *Super-naturalinstructions* (Wang *et al.*, 2022c), we show that language models trained with declarative natural instructions of NLP tasks could generalize over unseen tasks. In this work, we crowd-sourced a dataset of 1616 diverse NLP tasks and their expert-written instructions. Our collection covers 76 distinct task types, including (but not limited to) classification, extraction, infilling, sequence tagging, text rewriting, and text composition. Furthermore, we build Tk-Instruct, a transformer model trained to follow a variety of in-context instructions (plain language task definitions or k-shot examples). We show that even being orders of magnitude smaller than InstructGPT we were able to achieve a 9% improvement over it.

10.3 Knowledge Evaluation (KE)

While extraction and integration of knowledge are two aspects of knowledge for transformer-based language model development, their evaluation is of utmost necessity to find avenues for further improvements. In Chapter 9, I show that LMs often lack numerical reasoning skills. I developed and curated challenging numerical datasets for each of the tasks of numeration (number decoding), magnitude order prediction, finding minimum and maximum, and sorting numbers in a list of numbers. I show that T5 models are good at reasoning with numbers on which it is trained but fail to generalize on out-of-domain number range.

I would also like to highlight my contributions to two other projects in this direction. For the first project, we *evaluated large language models like GPT-2, GPT-3, and T5 about their feasibility reasoning capabilities* (Gupta *et al.*, 2022) by carefully creating challenge adversarial datasets in multiple-choice QA format and showing that they are marginally better than random performance. For the second project, we evaluated *how much language models can reason about the effects of actions* (Banerjee *et al.*, 2020) considering three classical planning domains like Blocksworld, Logistics, and Dock-Worker-Robots domains. In this project, we develop three synthetic multiple-choice QA datasets (leveraging Answer Set Programming) and show that if enough information about the current world knowledge and actions taken are provided then a transformer-based model can achieve near-perfect performance.

Overall, in this dissertation, through multiple projects, I proposed new approaches and models which pushed the state-of-the-art performance on existing datasets, developed novel datasets and performed an extensive analysis of proposed models and their prediction results. As can be seen from various projects demonstrated in this dissertation, knowledge plays multiple roles in improving transformer-based language models by infusing them into the models. This is possible by extracting such knowledge effectively. Also, they can be improved further by learning about their limitations through their proper knowledge evaluation. All the artifacts of the projects including datasets, codes, and presentations have been made public for future researchers in natural language, biomedical and cybersecurity domains.

REFERENCES

Abacha, A. B., A. G. S. de Herrera, K. Wang, L. R. Long, S. Antani and D. Demner-Fushman, "Named entity recognition in functional neuroimaging literature", in "2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)", pp. 2218–2220 (IEEE, 2017).

Agrawal, G., K. Pal, Y. Deng, H. Liu and C. Baral, "Aiseckg: Knowledge graph dataset for cybersecurity education", AAAI-MAKE 2023: Challenges Requiring the Combination of Machine Learning 2023 (2023).

Almeida, T. A., J. M. G. Hidalgo and A. Yamakami, "Contributions to the study of SMS spam filtering: new collection and results", in "Proceedings of the 2011 ACM Symposium on Document Engineering, Mountain View, CA, USA, September 19-22, 2011", edited by M. R. B. Hardy and F. W. Tompa, pp. 259–262 (ACM, 2011), URL https://doi.org/10.1145/2034691.2034742.

Almukaynizi, M., A. Grimm, E. Nunes, J. Shakarian and P. Shakarian, "Predicting cyber threats through hacker social networks in darkweb and deepweb forums", in "Proceedings of the 2017 International Conference of The Computational Social Science Society of the Americas", p. 12 (ACM, 2017a).

Almukaynizi, M., E. Marin, E. Nunes, P. Shakarian, G. I. Simari, D. Kapoor and T. Siedlecki, "DARKMENTION: A deployed system to predict enterprise-targeted external cyberattacks", in "2018 IEEE International Conference on Intelligence and Security Informatics, ISI 2018, Miami, FL, USA, November 9-11, 2018", pp. 31–36 (IEEE, 2018), URL https://doi.org/10.1109/ISI.2018.8587334.

Almukaynizi, M., E. Marin, M. Shah, E. Nunes, G. I. Simari and P. Shakarian, "A logic programming approach to predict enterprise-targeted cyberattacks", in "Data Science in Cybersecurity and Cyberthreat Intelligence", pp. 13–32 (Springer, 2020).

Almukaynizi, M., E. Nunes, K. Dharaiya, M. Senguttuvan, J. Shakarian and P. Shakarian, "Proactive identification of exploits in the wild through vulnerability mentions online", in "2017 International Conference on Cyber Conflict, CyCon U.S. 2017, Washington, DC, USA, November 7-8, 2017", edited by E. Sobiesk, D. Bennett and P. Maxwell, pp. 82–88 (IEEE Computer Society, 2017b), URL https://doi.org/10.1109/CYCONUS.2017.8167501.

Almukaynizi, M., E. Nunes, K. Dharaiya, M. Senguttuvan, J. Shakarian and P. Shakarian, "Patch before exploited: An approach to identify targeted software vulnerabilities", in "AI in Cybersecurity", pp. 81–113 (Springer, 2019).

Ameri, K., M. Hempel, H. R. Sharif, J. Lopez and K. S. Perumalla, "Cybert: Cybersecurity claim classification by fine-tuning the bert language model", Journal of Cybersecurity and Privacy (2021).

Amini, A., S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi and H. Hajishirzi, "Mathqa: Towards interpretable math word problem solving with operation-based formalisms", arXiv preprint arXiv:1905.13319 (2019).

Amith, M., L. Cui, K. Roberts and C. Tao, “Towards an ontology-based medication conversational agent for PrEP and PEP”, in “Proceedings of the First Workshop on Natural Language Processing for Medical Conversations”, pp. 31–40 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.nlpmc-1.5`.

Babić, D., S. Bucur, Y. Chen, F. Ivančić, T. King, M. Kusano, C. Lemieux, L. Szekeres and W. Wang, “Fudge: fuzz driver generation at scale”, in “Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering”, pp. 975–985 (2019).

Bacciu, D., A. Micheli and M. Podda, “Graph generation by sequential edge prediction.”, in “ESANN”, (2019).

Bada, M., M. Eckert, D. Evans, K. Garcia, K. Shipley, D. Sitnikov, W. A. Baumgartner, K. B. Cohen, K. Verspoor, J. A. Blake *et al.*, “Concept annotation in the craft corpus”, BMC bioinformatics **13**, 1, 161 (2012).

Banerjee, P., C. Baral, M. Luo, A. Mitra, K. Pal, T. C. Son and N. Varshney, “Can transformers reason about effects of actions?”, arXiv preprint arXiv:2012.09938 (2020).

Banerjee, P., K. K. Pal, M. V. Devarakonda and C. Baral, “Biomedical named entity recognition via knowledge guidance and question answering”, ACM Trans. Comput. Heal. **2**, 4, 33:1–33:24, URL `https://doi.org/10.1145/3465221` (2021).

Bauer, L., Y. Wang and M. Bansal, “Commonsense for generative multi-hop question answering tasks”, in “Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing”, pp. 4220–4230 (2018).

Beltagy, I., K. Lo and A. Cohan, “SciBERT: A pretrained language model for scientific text”, 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)”, pp. 3615–3620 (Association for Computational Linguistics, Hong Kong, China, 2019), URL `https://www.aclweb.org/anthology/D19-1371`.

Bethard, S., G. Savova, M. Palmer and J. Pustejovsky, “SemEval-2017 task 12: Clinical TempEval”, in “Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)”, pp. 565–572 (Association for Computational Linguistics, Vancouver, Canada, 2017), URL `https://www.aclweb.org/anthology/S17-2093`.

Bhagavatula, C., R. L. Bras, C. Malaviya, K. Sakaguchi, A. Holtzman, H. Rashkin, D. Downey, S. W.-t. Yih and Y. Choi, “Abductive commonsense reasoning”, arXiv preprint arXiv:1908.05739 (2019).

Bisk, Y., R. Zellers, R. L. Bras, J. Gao and Y. Choi, “Piqa: Reasoning about physical commonsense in natural language”, arXiv preprint arXiv:1911.11641 (2019).

Bodenreider, O., “The unified medical language system (umls): integrating biomedical terminology”, Nucleic acids research **32**, suppl_1, D267–D270 (2004).

Borthwick, A., J. Sterling, E. Agichtein and R. Grishman, “Exploiting diverse knowledge sources via maximum entropy in named entity recognition”, in “Sixth Workshop on Very Large Corpora”, (1998).

Bourquin, M., A. King and E. Robbins, “Binslayer: accurate comparison of binary executables”, in “Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop”, pp. 1–10 (2013).

Bridges, R. A., C. L. Jones, M. D. Iannacone and J. R. Goodall, “Automatic labeling for entity extraction in cyber security”, CoRR **abs/1308.4941**, URL `http://arxiv.org/abs/1308.4941` (2013).

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners”, Advances in neural information processing systems **33**, 1877–1901 (2020a).

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, “Language models are few-shot learners”, CoRR **abs/2005.14165**, URL `https://arxiv.org/abs/2005.14165` (2020b).

Chalkidis, I., M. Fergadiotis, P. Malakasiotis, N. Aletras and I. Androutsopoulos, “Legal-bert: The muppets straight out of law school”, arXiv preprint arXiv:2010.02559 (2020).

Charniak, E. and R. Goldman, “A logic for semantic interpretation”, in “Proceedings of the 26th annual meeting on Association for Computational Linguistics”, pp. 87–94 (Association for Computational Linguistics, 1988).

Charniak, E. and R. P. Goldman, “A semantics for probabilistic quantifier-free first-order languages, with particular application to story understanding.”, in “IJCAI”, vol. 89, pp. 1074–1079 (Citeseer, 1989).

Chen, C.-C., H.-H. Huang, H. Takamura and H.-H. Chen, “Numeracy-600k: learning numeracy for detecting exaggerated information in market comments”, in “Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics”, pp. 6307–6313 (2019).

Chen, Q., J. Lacomis, E. J. Schwartz, C. Le Goues, G. Neubig and B. Vasilescu, “Augmenting Decompiler Output with Learned Variable Names and Types”, in “Proceedings of the USENIX Security Symposium”, (2022).

Choi, E., H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang and L. Zettlemoyer, “Quac: Question answering in context”, arXiv preprint arXiv:1808.07036 (2018).

Chowdhery, A., S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, “Palm: Scaling language modeling with pathways”, arXiv preprint arXiv:2204.02311 (2022).

Chowdhuri, S., S. McCrea, D. D. Fushman and C. O. Taylor, “Extracting biomedical terms from postpartum depression online health communities”, AMIA Summits on Translational Science Proceedings **2019**, 592 (2019).

Chua, Z. L., S. Shen, P. Saxena and Z. Liang, “Neural nets can learn function type signatures from binaries”, in “26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017”, edited by E. Kirda and T. Ristenpart, pp. 99–116 (USENIX Association, 2017), URL `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/chua`.

Ciaramita, M. and Y. Altun, “Named-entity recognition in novel domains with external lexical knowledge”, in “Proceedings of the NIPS Workshop on Advances in Structured Learning for Text and Speech Processing”, vol. 2005 (2005).

Cifuentes, C., *Reverse Compilation Techniques*, Ph.D. thesis, Queensland University of Technology (1994).

Clark, P., I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick and O. Tafjord, “Think you have solved question answering? try arc, the ai2 reasoning challenge”, arXiv preprint arXiv:1803.05457 (2018).

Clark, P., O. Etzioni, T. Khot, A. Sabharwal, O. Tafjord, P. Turney and D. Khashabi, “Combining retrieval, statistics, and inference to answer elementary science questions”, in “Thirtieth AAAI Conference on Artificial Intelligence”, (2016).

Clark, P., O. Tafjord and K. Richardson, “Transformers as soft reasoners over language”, arXiv preprint arXiv:2002.05867 (2020).

Cohen, K. B. and L. Hunter, “Natural language processing and systems biology”, in “Artificial intelligence methods and tools for systems biology”, pp. 147–173 (Springer, 2004).

Conneau, A., D. Kiela, H. Schwenk, L. Barrault and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data”, in “Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing”, pp. 670–680 (Association for Computational Linguistics, Copenhagen, Denmark, 2017), URL `https://aclanthology.org/D17-1070`.

Copyright Office, U. S., “Copyright law of the united states, accessed: 2021-05-01”, `https://www.copyright.gov/title17/92chap1.html#107` (2016).

Crichton, G., S. Pyysalo, B. Chiu and A. Korhonen, “A neural network multi-task learning approach to biomedical named entity recognition”, BMC bioinformatics **18**, 1, 368 (2017).

CTFTime, “CTFTime, accessed: 2021-05-01”, `https://ctftime.org` (2021).

decompiler, T. G., “The Ghidra decompiler”, `https://ghidra-sre.org/` (2022).

Deliu, I., C. Leichter and K. Franke, “Extracting cyber threat intelligence from hacker forums: Support vector machines versus convolutional neural networks”, in “2017 IEEE International Conference on Big Data (Big Data)”, pp. 3648–3656 (IEEE, 2017).

Deliu, I., C. Leichter and K. Franke, “Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent dirichlet allocation”, in “2018 IEEE International Conference on Big Data (Big Data)”, pp. 5008–5013 (IEEE, 2018).

Delpech, E. and P. Saint-Dizier, “Investigating the structure of procedural texts for answering how-to questions”, in “Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)”, (European Language Resources Association (ELRA), Marrakech, Morocco, 2008), URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/20_paper.pdf.

Demszky, D., K. Guu and P. Liang, “Transforming question answering datasets into natural language inference datasets”, arXiv preprint arXiv:1809.02922 (2018).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in “2009 IEEE conference on computer vision and pattern recognition”, pp. 248–255 (Ieee, 2009).

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, “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)”, pp. 4171–4186 (Association for Computational Linguistics, Minneapolis, Minnesota, 2019a), URL https://www.aclweb.org/anthology/N19-1423.

Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, “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)”, pp. 4171–4186 (2019b).

DiMaggio, J., “The black vine cyberspionage group”, Tech. rep., Symantec, URL http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-black-vine-cyberespionage-group.pdf (2015).

DIRTY, “The log of a CI run for dirty on GitHub Actions”, https://github.com/CMUSTRUDEL/DIRTY/runs/6116940575 (2022).

Doğan, R. I., R. Leaman and Z. Lu, “Ncbi disease corpus: a resource for disease name recognition and concept normalization”, Journal of biomedical informatics **47**, 1–10 (2014).

Donahue, C., M. Lee and P. Liang, “Enabling language models to fill in the blanks”, arXiv preprint arXiv:2005.05339 (2020).

Dramko, L., J. Lacomis, P. Yin, E. J. Schwartz, M. Allamanis, G. Neubig, B. Vasilescu and C. Le Goues, "Dire and its data: Neural decompiled variable renamings with respect to software class", ACM Transactions on Software Engineering and Methodology (2022).

Dua, D., Y. Wang, P. Dasigi, G. Stanovsky, S. Singh and M. Gardner, "Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs", arXiv preprint arXiv:1903.00161 (2019a).

Dua, D., Y. Wang, P. Dasigi, G. Stanovsky, S. Singh and M. Gardner, "Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs", arXiv preprint arXiv:1903.00161 (2019b).

Duan, Y., X. Li, J. Wang and H. Yin, "Deepbindiff: Learning program-wide code representations for binary diffing", in "Network and Distributed System Security Symposium", (2020).

Ďurfina, L., J. Křoustek and P. Zemek, "Psybot malware: A step-by-step decompilation case study", in "2013 20th Working Conference on Reverse Engineering (WCRE)", pp. 449–456 (IEEE, 2013).

Ďurfina, L., J. Křoustek, P. Zemek, D. Kolář, T. Hruška, K. Masařík and A. Meduna, "Design of a retargetable decompiler for a static platform-independent malware analysis", in "International Conference on Information Security and Assurance", pp. 72–86 (Springer, 2011).

Efrat, A. and O. Levy, "The turking test: Can language models understand instructions?", arXiv preprint arXiv:2010.11982 (2020).

Engineering, M. S. R., "An exhaustively-analyzed IDB for ComRAT v4", https://www.msreverseengineering.com/blog/2020/8/31/an-exhaustively-analyzed-idb-for-comrat-v4 (2021).

Epure, E. V., P. Martín-Rodilla, C. Hug, R. Deneckère and C. Salinesi, "Automatic process model discovery from textual methodologies", in "2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)", pp. 19–30 (IEEE, 2015).

European Union, ., "Copyright in the eu, accessed: 2021-05-01", https://europa.eu/youreurope/business/running-business/intellectual-property/copyright/index_en.htm (2020).

Fey, M. and J. E. Lenssen, "Fast graph representation learning with pytorch geometric", arXiv preprint arXiv:1903.02428 (2019).

Fontan, L. and P. Saint-Dizier, "Analyzing the explanation structure of procedural texts: Dealing with advice and warnings", in "Semantics in Text Processing. STEP 2008 Conference Proceedings", pp. 115–127 (2008).

Fu, C., H. Chen, H. Liu, X. Chen, Y. Tian, F. Koushanfar and J. Zhao, "A neural-based program decompiler", arXiv:1906.12029 [cs] URL `http://arxiv.org/abs/1906.12029`, arXiv: 1906.12029 (2019).

Gao, H., S. Cheng, Y. Xue and W. Zhang, "A lightweight framework for function name reassignment based on large-scale stripped binaries", in "Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis", pp. 607–619 (2021).

Gardner, M., J. Berant, H. Hajishirzi, A. Talmor and S. Min, "Question answering is a format; when is it useful?", arXiv preprint arXiv:1909.11291 (2019).

Gaskell, A., Y. Miao, L. Specia and F. Toni, "Logically consistent adversarial attacks for soft theorem provers", arXiv preprint arXiv:2205.00047 (2022).

Georgescu, T., B. Iancu and M. Zurini, "Named-entity-recognition-based automated system for diagnosing cybersecurity situations in iot networks", Sensors **19**, 15, 3380, URL `https://doi.org/10.3390/s19153380` (2019).

Gerner, M., G. Nenadic and C. M. Bergman, "Linnaeus: a species name identification system for biomedical literature", BMC bioinformatics **11**, 1, 85 (2010).

Geva, M., A. Gupta and J. Berant, "Injecting numerical reasoning skills into language models", in "Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics", pp. 946–958 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.acl-main.89`.

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, "Neural message passing for quantum chemistry", arXiv preprint arXiv:1704.01212 (2017).

Giorgi, J. M. and G. D. Bader, "Towards reliable named entity recognition in the biomedical domain", Bioinformatics **36**, 1, 280–286 (2020).

Gontier, N., K. Sinha, S. Reddy and C. Pal, "Measuring systematic generalization in neural proof generation with transformers", arXiv preprint arXiv:2009.14786 (2020).

Gross, J. and C. S. team, "Operation dust storm", Tech. rep., Cylance, URL `https://www.cylance.com/hubfs/2015_cylance_website/assets/operation-dust-storm/Op_Dust_Storm_Report.pdf` (2016).

Gu, J., Z. Lu, H. Li and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning", in "Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)", pp. 1631–1640 (Association for Computational Linguistics, 2016), URL `http://aclweb.org/anthology/P16-1154`.

Gu, Y., Z. Zhang, X. Wang, Z. Liu and M. Sun, "Train no evil: Selective masking for task-guided pre-training", arXiv preprint arXiv:2004.09733 (2020).

Gubrud, M., "Nanotechnology and international security", Fifth Foresight Conference on Molecular Nanotechnology (1997).

Gupta, H., N. Varshney, S. Mishra, K. K. Pal, S. A. Sawant, K. Scaria, S. Goyal and C. Baral, “” john is 50 years old, can his son be 65?” evaluating nlp models’ understanding of feasibility”, arXiv preprint arXiv:2210.07471 (2022).

Gururangan, S., A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey and N. A. Smith, “Don’t stop pretraining: Adapt language models to domains and tasks”, arXiv preprint arXiv:2004.10964 (2020).

Gussoni, A., A. Di Federico, P. Fezzardi and G. Agosta, “A comb for decompiled c code”, in “Proceedings of the 15th ACM Asia Conference on Computer and Communications Security”, p. 637–651 (ACM, 2020), URL `https://dl.acm.org/doi/10.1145/3320269.3384766`.

Hamilton, W., Z. Ying and J. Leskovec, “Inductive representation learning on large graphs”, in “Advances in neural information processing systems”, pp. 1024–1034 (2017).

Han, S., H. Schoelkopf, Y. Zhao, Z. Qi, M. Riddell, L. Benson, L. Sun, E. Zubova, Y. Qiao, M. Burtell *et al.*, “Folio: Natural language reasoning with first-order logic”, arXiv preprint arXiv:2209.00840 (2022).

Hanga, K. M., Y. Kovalchuk and M. M. Gaber, “A graph-based approach to interpreting recurrent neural networks in process mining”, IEEE Access **8**, 172923–172938 (2020).

Haonan, L., S. H. Huang, T. Ye and G. Xiuyan, “Graph star net for generalized multi-task learning”, arXiv preprint arXiv:1906.12330 (2019).

He, J., P. Ivanov, P. Tsankov, V. Raychev and M. Vechev, “Debin: Predicting debug information in stripped binaries”, in “Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security”, pp. 1667–1680 (2018).

He, L., M. Lewis and L. Zettlemoyer, “Question-answer driven semantic role labeling: Using natural language to annotate natural language”, in “Proceedings of the 2015 conference on empirical methods in natural language processing”, pp. 643–653 (2015).

Hendrycks, D. and K. Gimpel, “Gaussian error linear units (gelus)”, arXiv preprint arXiv:1606.08415 (2016).

Hex-Rays, S., “Hex-rays decompiler”, (2013).

Hobbs, J. R., “Abduction in natural language understanding”, Handbook of pragmatics pp. 724–741 (2004).

Hobbs, J. R., M. E. Stickel, D. E. Appelt and P. Martin, “Interpretation as abduction”, Artificial intelligence **63**, 1-2, 69–142 (1993).

Hoffmann, J., S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark *et al.*, “Training compute-optimal large language models”, arXiv preprint arXiv:2203.15556 (2022).

Honkisz, K., K. Kluza and P. Wiśniewski, “A concept for generating business process models from natural language description”, in “International Conference on Knowledge Science, Engineering and Management”, pp. 91–103 (Springer, 2018).

Honnibal, M. and I. Montani, “spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing”, To appear (2017).

Hopper, “Hopper”, `https://www.hopperapp.com/` (2022).

Huang, K., J. Altosaar and R. Ranganath, “Clinicalbert: Modeling clinical notes and predicting hospital readmission”, CoRR **abs/1904.05342**, URL `http://arxiv.org/abs/1904.05342` (2019).

Huang, S. and Y. Wu, “POSTER: dynamic software vulnerabilities threat prediction through social media contextual analysis”, in “ASIA CCS ’20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020”, edited by H. Sun, S. Shieh, G. Gu and G. Ateniese, pp. 892–894 (ACM, 2020), URL `https://doi.org/10.1145/3320269.3405435`.

Huang, Y., M. Fang, Y. Cao, L. Wang and X. Liang, “Dagn: Discourse-aware graph network for logical reasoning”, arXiv preprint arXiv:2103.14349 (2021).

Jaffe, A., J. Lacomis, E. J. Schwartz, C. L. Goues and B. Vasilescu, “Meaningful variable names for decompiled code: A machine translation approach”, in “Proceedings of the 26th Conference on Program Comprehension”, pp. 20–30 (2018).

Jenkins, T., *Open Book Assessment in Computing Degree Programmes* (Citeseer, 1995).

Jeong, C., S. Jang, H. Shin, E. Park and S. Choi, “A context-aware citation recommendation model with bert and graph convolutional networks”, arXiv preprint arXiv:1903.06464 (2019).

Jermsurawong, J. and N. Habash, “Predicting the structure of cooking recipes”, in “Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing”, pp. 781–786 (2015).

Jiang, C., Z. Nian, K. Guo, S. Chu, Y. Zhao, L. Shen and K. Tu, “Learning numeral embeddings”, arXiv preprint arXiv:2001.00003 (2019).

Jiao, F., Y. Guo, X. Song and L. Nie, “MERIt: Meta-Path Guided Contrastive Learning for Logical Reasoning”, in “Findings of the Association for Computational Linguistics: ACL 2022”, pp. 3496–3509 (Association for Computational Linguistics, Dublin, Ireland, 2022), URL `https://aclanthology.org/2022.findings-acl.276`.

Johnson, D., D. Mak, A. Barker and L. Loessberg-Zahl, “Probing for multilingual numerical understanding in transformer-based language models”, in “Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP”, pp. 184–192 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.blackboxnlp-1.18`.

Joshi, M., E. Choi, D. S. Weld and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension", arXiv preprint arXiv:1705.03551 (2017).

Jung, J., S. Tong, H. Hu, J. Lim, Y. Jin and T. Kim, "Winnie: Fuzzing windows applications with harness synthesis and fast cloning", in "Proceedings of the 2021 Annual Network and Distributed System Security Symposium (NDSS), Virtual", (2021).

Kalle, S., N. Ameen, H. Yoo and I. Ahmed, "Clik on plcs! attacking control logic with decompilation and virtual plc", in "Binary Analysis Research (BAR) Workshop, Network and Distributed System Security Symposium (NDSS)", (2019).

Kashihara, K., K. K. Pal, C. Baral and R. P. Trevino, "Prompt-based learning for thread structure prediction in cybersecurity forums", arXiv preprint arXiv:2303.05400 (2023).

Kashihara, K., J. Shakarian and C. Baral, "Social structure construction from the forums using interaction coherence", in "Proceedings of the Future Technologies Conference", pp. 830–843 (2020).

Katiyar, A. and C. Cardie, "Nested named entity recognition revisited", in "Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)", pp. 861–871 (2018).

Katz, D. S., J. Ruchti and E. Schulte, "Using recurrent neural networks for decompilation", in "2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)", p. 346–356 (IEEE, 2018), URL http://ieeexplore.ieee.org/document/8330222/.

Kazama, J. and K. Torisawa, "Exploiting wikipedia as external knowledge for named entity recognition", in "Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)", pp. 698–707 (2007).

Khashabi, D., S. Chaturvedi, M. Roth, S. Upadhyay and D. Roth, "Looking beyond the surface: A challenge set for reading comprehension over multiple sentences", in "Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)", vol. 1, pp. 252–262 (2018).

Khashabi, D., T. Khot, A. Sabharwal and D. Roth, "Learning what is essential in questions", in "Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)", pp. 80–89 (2017).

Khashabi, D., T. Khot, A. Sabharwal, O. Tafjord, P. Clark and H. Hajishirzi, "Unifiedqa: Crossing format boundaries with a single qa system", arXiv preprint arXiv:2005.00700 (2020).

Kiddon, C., G. T. Ponnuraj, L. Zettlemoyer and Y. Choi, "Mise en place: Unsupervised interpretation of instructional recipes", in "Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing", pp. 982–992 (2015).

Kidmose, E., M. Stevanovic and J. M. Pedersen, "Detection of malicious domains through lexical analysis", in "2018 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2018, Glasgow, Scotland, United Kingdom, June 11-12, 2018", pp. 1–5 (IEEE, 2018), URL `https://doi.org/10.1109/CyberSecPODS.2018.8560665`.

Kim, J.-D., T. Ohta, S. Pyysalo, Y. Kano and J. Tsujii, "Overview of bionlp'09 shared task on event extraction", in "Proceedings of the BioNLP 2009 workshop companion volume for shared task", pp. 1–9 (2009).

Kim, J.-D., T. Ohta, Y. Tateisi and J. Tsujii, "Genia corpus—a semantically annotated corpus for bio-textmining", Bioinformatics **19**, suppl_1, i180–i182 (2003).

Kim, J.-D., T. Ohta, Y. Tsuruoka, Y. Tateisi and N. Collier, "Introduction to the bio-entity recognition task at jnlpba", in "Proceedings of the international joint workshop on natural language processing in biomedicine and its applications", pp. 70–75 (Citeseer, 2004).

Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", arXiv preprint arXiv:1412.6980 (2014).

Kipf, T. N. and M. Welling, "Semi-supervised classification with graph convolutional networks", arXiv preprint arXiv:1609.02907 (2016).

Kirillov, I., D. Beck, P. Chase and R. Martin, "Malware attribute enumeration and characterization", The MITRE Corporation [online, accessed Apr. 8, 2019] (2011).

Koupaee, M. and W. Y. Wang, "Wikihow: A large scale text summarization dataset", arXiv preprint arXiv:1810.09305 (2018).

Krallinger, M., F. Leitner, O. Rabal, M. Vazquez, J. Oyarzabal and A. Valencia, "Chemdner: The drugs and chemical names extraction challenge", Journal of cheminformatics **7**, S1, S1 (2015).

Kumar, S., S. Jat, K. Saxena and P. Talukdar, "Zero-shot word sense disambiguation using sense definition embeddings", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 5670–5681 (Association for Computational Linguistics, Florence, Italy, 2019), URL `https://aclanthology.org/P19-1568`.

Kwiatkowski, T., J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, "Natural questions: a benchmark for question answering research", Transactions of the Association for Computational Linguistics **7**, 453–466 (2019).

Lacomis, J., P. Yin, E. Schwartz, M. Allamanis, C. Le Goues, G. Neubig and B. Vasilescu, "DIRE: A neural approach to decompiled identifier naming", in "2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)", pp. 628–639 (IEEE, 2019).

Lai, G., Q. Xie, H. Liu, Y. Yang and E. Hovy, "Race: Large-scale reading comprehension dataset from examinations", in "Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing", pp. 785–794 (Association for Computational Linguistics, 2017), URL http://aclweb.org/anthology/D17-1082.

Landsberger, J., "Study guides and strategies.", URL Http://www.studygs.net/tsttak7.htm. (1996).

Lee, J., T. Avgerinos and D. Brumley, "Tie: Principled reverse engineering of types in binary programs", in "Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS'11)", p. 18 (2011).

Lee, J., W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining", Bioinformatics **36**, 4, 1234–1240 (2020a).

Lee, J., W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining", Bioinform. **36**, 4, 1234–1240, URL https://doi.org/10.1093/bioinformatics/btz682 (2020b).

Lhoest, Q., A. Villanova del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Šaško, G. Chhablani, B. Malik, S. Brandeis, T. Le Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussière, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. Rush and T. Wolf, "Datasets: A community library for natural language processing", in "Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations", pp. 175–184 (Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021), URL https://aclanthology.org/2021.emnlp-demo.21.

L'Huillier, G., A. Hevia, R. Weber and S. A. Ríos, "Latent semantic analysis and keyword extraction for phishing classification", in "IEEE International Conference on Intelligence and Security Informatics, ISI 2010, Vancouver, BC, Canada, May 23-26, 2010, Proceedings", edited by C. C. Yang, D. Zeng, K. Wang, A. Sanfilippo, H. H. Tsang, M. Day, U. Glässer, P. L. Brantingham and H. Chen, pp. 129–131 (IEEE, 2010), URL https://doi.org/10.1109/ISI.2010.5484762.

Li, J.-c., D.-l. Zhao, B.-F. Ge, K.-W. Yang and Y.-W. Chen, "A link prediction method for heterogeneous networks based on bp neural network", Physica A: Statistical Mechanics and its Applications **495**, 1–17 (2018).

Li, X., J. Feng, Y. Meng, Q. Han, F. Wu and J. Li, "A unified MRC framework for named entity recognition", in "Proceedings of the 58th Annual Meeting of the

Association for Computational Linguistics", pp. 5849–5859 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.acl-main.519`.

Li, Y., J. Cheng, C. Huang, Z. Chen and W. Niu, "Nedetector: Automatically extracting cybersecurity neologisms from hacker forums", J. Inf. Secur. Appl. **58**, 102784, URL `https://doi.org/10.1016/j.jisa.2021.102784` (2021).

Lieber, O., O. Sharir, B. Lenz and Y. Shoham, "Jurassic-1: Technical details and evaluation", Tech. rep., AI21 Labs (2021).

Lifschitz, V., *Answer set programming* (Springer Berlin, 2019).

Lin, B. Y., X. Chen, J. Chen and X. Ren, "Kagnet: Knowledge-aware graph networks for commonsense reasoning", 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)", pp. 2822–2832 (2019a).

Lin, B. Y., S. Lee, R. Khanna and X. Ren, "Birds have four legs?! numersense: Probing numerical commonsense knowledge of pre-trained language models", in "Proceedings of EMNLP", (2020), to appear.

Lin, B. Y., W. Zhou, M. Shen, P. Zhou, C. Bhagavatula, Y. Choi and X. Ren, "Commongen: A constrained text generation challenge for generative commonsense reasoning", arXiv preprint arXiv:1911.03705 (2019b).

Lin, H., Y. Lu, X. Han and L. Sun, "Sequence-to-nuggets: Nested entity mention detection via anchor-region networks", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 5182–5192 (2019c).

Liu, H., Z.-Z. Hu, M. Torii, C. Wu and C. Friedman, "Quantitative assessment of dictionary-based protein named entity tagging", Journal of the American Medical Informatics Association **13**, 5, 497–507 (2006).

Liu, J., L. Cui, H. Liu, D. Huang, Y. Wang and Y. Zhang, "Logiqa: A challenge dataset for machine reading comprehension with logical reasoning", arXiv preprint arXiv:2007.08124 (2020a).

Liu, P., W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing", (2021).

Liu, T., J.-G. Yao and C.-Y. Lin, "Towards improving neural named entity recognition with gazetteers", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 5301–5307 (Association for Computational Linguistics, Florence, Italy, 2019a), URL `https://www.aclweb.org/anthology/P19-1524`.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach", arXiv preprint arXiv:1907.11692 (2019b).

Liu, Z., D. Huang, K. Huang, Z. Li and J. Zhao, “Finbert: A pre-trained financial language representation model for financial text mining”, in “Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020”, edited by C. Bessiere, pp. 4513–4519 (ijcai.org, 2020b), URL https://doi.org/10.24963/ijcai.2020/622.

Loshchilov, I. and F. Hutter, “Decoupled weight decay regularization”, arXiv preprint arXiv:1711.05101 (2017).

Lourie, N., R. L. Bras, C. Bhagavatula and Y. Choi, “UNICORN on RAINBOW: A universal commonsense reasoning model on a new multitask benchmark”, in “Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021”, pp. 13480–13488 (AAAI Press, 2021), URL https://ojs.aaai.org/index.php/AAAI/article/view/17590.

Lu, Z., P. Du and J.-Y. Nie, “Vgcn-bert: Augmenting bert with graph embedding for text classification”, in “European Conference on Information Retrieval”, pp. 369–382 (Springer, 2020).

Luan, Y., D. Wadden, L. He, A. Shah, M. Ostendorf and H. Hajishirzi, “A general framework for information extraction using dynamic span graphs”, 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)”, pp. 3036–3046 (2019).

Lv, S., D. Guo, J. Xu, D. Tang, N. Duan, M. Gong, L. Shou, D. Jiang, G. Cao and S. Hu, “Graph-based reasoning over heterogeneous external knowledge for commonsense question answering.”, in “AAAI”, pp. 8449–8456 (2020).

Madsen, A. and A. R. Johansen, “Neural arithmetic units”, arXiv preprint arXiv:2001.05016 (2020).

Maeta, H., T. Sasada and S. Mori, “A framework for procedural text understanding”, in “Proceedings of the 14th International Conference on Parsing Technologies”, pp. 50–60 (2015).

Malaviya, C., C. Bhagavatula, A. Bosselut and Y. Choi, “Exploiting structural and semantic context for commonsense knowledge base completion”, arXiv preprint arXiv:1910.02915 (2019).

Malmaud, J., E. Wagner, N. Chang and K. Murphy, “Cooking with semantics”, in “Proceedings of the ACL 2014 Workshop on Semantic Parsing”, pp. 33–38 (2014).

Mantovani, A., S. Aonzo, Y. Fratantonio and D. Balzarotti, “RE-Mind: a First Look Inside the Mind of a Reverse Engineer”, in “USENIX Security Symposium”, pp. 2727–2745 (2022a).

Mantovani, A., L. Compagna, Y. Shoshitaishvili and D. Balzarotti, “The convergence of source code and binary vulnerability discovery – a case study”, in “Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS22)”, ASIACCS 22 (2022b).

Marchal, S., J. François, R. State and T. Engel, “Phishstorm: Detecting phishing with streaming analytics”, IEEE Transactions on Network and Service Management **11**, 458–471 (2014).

Marcus, G. and E. Davis, *Rebooting AI: Building artificial intelligence we can trust* (Pantheon, 2019).

McCann, B., J. Bradbury, C. Xiong and R. Socher, “Learned in translation: Contextualized word vectors”, arXiv preprint arXiv:1708.00107 (2017).

McCann, B., N. S. Keskar, C. Xiong and R. Socher, “The natural language decathlon: Multitask learning as question answering”, arXiv preprint arXiv:1806.08730 (2018).

McCarthy, J., *Programs with common sense* (RLE and MIT computation center, 1959).

McDonnell, S., O. Nada, M. R. Abid and E. Amjadian, “Cyberbert: A deep dynamic-state session-based recommender system for cyber threat recognition”, 2021 IEEE Aerospace Conference (50100) pp. 1–12 (2021).

Merity, S., C. Xiong, J. Bradbury and R. Socher, “Pointer sentinel mixture models”, (2016).

Mihaylov, T., P. Clark, T. Khot and A. Sabharwal, “Can a suit of armor conduct electricity? a new dataset for open book question answering”, in “EMNLP”, (2018).

Mirzaei, O., R. Vasilenko, E. Kirda, L. Lu and A. Kharraz, “Scrutinizer: Detecting code reuse in malware via decompilation and machine learning”, in “International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment”, pp. 130–150 (Springer, 2021).

Mishra, S., D. Khashabi, C. Baral and H. Hajishirzi, “Cross-task generalization via natural language crowdsourcing instructions”, arXiv preprint arXiv:2104.08773 (2021a).

Mishra, S., D. Khashabi, C. Baral and H. Hajishirzi, “Natural instructions: Benchmarking generalization to new tasks from natural language instructions”, arXiv preprint arXiv:2104.08773 (2021b).

Mishra, S., A. Mitra, N. Varshney, B. Sachdeva and C. Baral, “Towards question format independent numerical reasoning: A set of prerequisite tasks”, arXiv preprint arXiv:2005.08516 (2020).

Mitra, A., P. Clark, O. Tafjord and C. Baral, “Declarative question answering over knowledge bases containing natural language text with answer set programming”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 33, pp. 3003–3010 (2019).

Mohasseb, A., B. Aziz and A. Kanavos, "SMS spam identification and risk assessment evaluations", in "Proceedings of the 16th International Conference on Web Information Systems and Technologies, WEBIST 2020, Budapest, Hungary, November 3-5, 2020", edited by M. Marchiori, F. J. D. Mayo and J. Filipe, pp. 417–424 (SCITEPRESS, 2020), URL `https://doi.org/10.5220/0010022404170424`.

Mori, S., H. Maeta, Y. Yamakata and T. Sasada, "Flow graph corpus from recipe texts.", in "LREC", pp. 2370–2377 (2014).

Mostafazadeh, N., N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli and J. Allen, "A corpus and evaluation framework for deeper understanding of commonsense stories", arXiv preprint arXiv:1604.01696 (2016).

Musa, R., X. Wang, A. Fokoue, N. Mattei, M. Chang, P. Kapanipathi, B. Makni, K. Talamadupula and M. Witbrock, "Answering science exam questions using query rewriting with background knowledge", arXiv preprint arXiv:1809.05726 (2018).

Mysore, S., Z. Jensen, E. Kim, K. Huang, H.-S. Chang, E. Strubell, J. Flanigan, A. McCallum and E. Olivetti, "The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures", arXiv preprint arXiv:1905.06939 (2019).

Naik, A., A. Ravichander, C. Rose and E. Hovy, "Exploring numeracy in word embeddings", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 3374–3380 (2019).

Nakayama, H., "seqeval: A python framework for sequence labeling evaluation", URL `https://github.com/chakki-works/seqeval`, software available from https://github.com/chakki-works/seqeval (2018).

Nédellec, C., R. Bossy, J.-D. Kim, J.-J. Kim, T. Ohta, S. Pyysalo and P. Zweigenbaum, "Overview of bionlp shared task 2013", in "Proceedings of the BioNLP shared task 2013 workshop", pp. 1–7 (2013).

Nguyen, T. O.-R. B. N., J. T. J.-D. Kim and S. Pyysalo, "Overview of bionlp shared task 2011", in "Proceedings of BioNLP Shared Task 2011 Workshop", pp. 1–6 (2011).

Ni, J., C. Zhu, W. Chen and J. McAuley, "Learning to attend on essential terms: An enhanced retriever-reader model for scientific question answering", arXiv preprint arXiv:1808.09492 (2018).

Ninja, B., "Binary Ninja", `https://binary.ninja/` (2022).

Nitin, V., A. Saieva, B. Ray and G. Kaiser, "DIRECT: A Transformer-based Model for Decompiled Variable Name Recovery", Workshop on Natural Language Processing for Programming (NLP4Prog) (2021).

Nogueira, R., Z. Jiang and J. Li, "Investigating the limitations of the transformers with simple arithmetic tasks", arXiv preprint arXiv:2102.13019 (2021).

Noonan, M., A. Loginov and D. Cok, "Polymorphic type inference for machine code", in "Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation", p. 27–41 (2016), URL http://arxiv.org/abs/1603.05495.

Norvig, P., "Frame activated inferences in a story understanding program.", in "IJCAI", pp. 624–626 (1983).

Norvig, P., "Inference in text understanding.", in "AAAI", pp. 561–565 (1987).

OpenAI, "Gpt-4 technical report", arXiv (2023).

Ott, M., S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling", in "Proceedings of NAACL-HLT 2019: Demonstrations", (2019).

Ouyang, S., Z. Zhang and H. Zhao, "Fact-driven logical reasoning", arXiv preprint arXiv:2105.10334 (2021).

Pal, K. K. and C. Baral, "Investigating numeracy learning ability of a text-to-text transfer model", in "Findings of the Association for Computational Linguistics: EMNLP 2021", pp. 3095–3101 (Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021), URL https://aclanthology.org/2021.findings-emnlp.265.

Pal, K. K., K. Kashihara, U. Anantheswaran, K. C. Kuznia, S. Jagtap and C. Baral, "Exploring the limits of transfer learning with unified model in the cybersecurity domain", arXiv preprint arXiv:2302.10346 (2023).

Pal, K. K., K. Kashihara, P. Banerjee, S. Mishra, R. Wang and C. Baral, "Constructing flow graphs from procedural cybersecurity texts", arXiv preprint arXiv:2105.14357 (2021a).

Pal, K. K., K. Kashihara, P. Banerjee, S. Mishra, R. Wang and C. Baral, "Constructing flow graphs from procedural cybersecurity texts", in "Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021", edited by C. Zong, F. Xia, W. Li and R. Navigli, vol. ACL/IJCNLP 2021 of *Findings of ACL*, pp. 3945–3957 (Association for Computational Linguistics, 2021b), URL https://doi.org/10.18653/v1/2021.findings-acl.345.

Pan, L.-M., J. Chen, J. Wu, S. Liu, C.-W. Ngo, M.-Y. Kan, Y. Jiang and T.-S. Chua, "Multi-modal cooking workflow construction for food recipes", in "Proceedings of the 28th ACM International Conference on Multimedia", pp. 1132–1141 (2020).

Pan, S., J. Wu, X. Zhu, G. Long and C. Zhang, "Task sensitive feature exploration and learning for multitask graph classification", IEEE transactions on cybernetics **47**, 3, 744–758 (2016).

Pan, S., J. Wu, X. Zhu, C. Zhang and S. Y. Philip, "Joint structure feature exploration and regularization for multi-task graph classification", IEEE Transactions on Knowledge and Data Engineering **28**, 3, 715–728 (2015).

Pandey, B., P. K. Bhanodia, A. Khamparia and D. K. Pandey, “A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges”, Expert Systems with Applications **124**, 164–181 (2019).

Park, H. and H. R. Motahari Nezhad, “Learning procedures from text: Codifying how-to procedures in deep neural networks”, in “Companion Proceedings of the The Web Conference 2018”, pp. 351–358 (2018).

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library”, in “Advances in Neural Information Processing Systems 32”, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, pp. 8024–8035 (Curran Associates, Inc., 2019), URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, “Scikit-learn: Machine learning in Python”, Journal of Machine Learning Research **12**, 2825–2830 (2011).

Pei, K., J. Guan, M. Broughton, Z. Chen, S. Yao, D. Williams-King, V. Ummadisetty, J. Yang, B. Ray and S. Jana, “StateFormer: Fine-Grained Type Recovery from Binaries using Generative State Modeling”, in “Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering”, pp. 690–702 (2021a).

Pei, K., J. Guan, M. Broughton, Z. Chen, S. Yao, D. Williams-King, V. Ummadisetty, J. Yang, B. Ray and S. Jana, “Stateformer: fine-grained type recovery from binaries using generative state modeling”, in “ESEC/FSE ’21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021”, edited by D. Spinellis, G. Gousios, M. Chechik and M. D. Penta, pp. 690–702 (ACM, 2021b), URL https://doi.org/10.1145/3468264.3468607.

Peng, Y., S. Yan and Z. Lu, “Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets”, in “Proceedings of the 2019 Workshop on Biomedical Natural Language Processing (BioNLP 2019)”, pp. 58–65 (2019).

Pennington, J., R. Socher and C. D. Manning, “Glove: Global vectors for word representation”, in “EMNLP (2014)”, pp. 1532–1543 (2014), URL http://www.aclweb.org/anthology/D14-1162.

Perera, I., J. D. Hwang, K. Bayas, B. J. Dorr and Y. Wilks, “Cyberattack prediction through public text analysis and mini-theories”, in “IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018”, edited by N. Abe, H. Liu, C. Pu, X. Hu, N. K. Ahmed, M. Qiao, Y. Song, D. Kossmann,

B. Liu, K. Lee, J. Tang, J. He and J. S. Saltz, pp. 3001–3010 (IEEE, 2018), URL https://doi.org/10.1109/BigData.2018.8622106.

Phan, L. N., J. T. Anibal, H. Tran, S. Chanana, E. Bahadroglu, A. Peltekian and G. Altan-Bonnet, "Scifive: a text-to-text transformer model for biomedical literature", arXiv preprint arXiv:2106.03598 (2021).

Phandi, P., A. Silva and W. Lu, "SemEval-2018 task 8: Semantic extraction from CybersecUrity REports using natural language processing (SecureNLP)", in "Proceedings of the 12th International Workshop on Semantic Evaluation", pp. 697–706 (Association for Computational Linguistics, New Orleans, Louisiana, 2018), URL https://aclanthology.org/S18-1113.

Portnoff, R. S., S. Afroz, G. Durrett, J. K. Kummerfeld, T. Berg-Kirkpatrick, D. McCoy, K. Levchenko and V. Paxson, "Tools for automated analysis of cybercriminal markets", in "Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017", edited by R. Barrett, R. Cummings, E. Agichtein and E. Gabrilovich, pp. 657–666 (ACM, 2017), URL https://doi.org/10.1145/3038912.3052600.

Prakash, A., A. Sharma, A. Mitra and C. Baral, "Combining knowledge hunting and neural language models to solve the winograd schema challenge", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 6110–6119 (2019).

Pyysalo, S. and S. Ananiadou, "Anatomical entity mention recognition at literature scale", Bioinformatics **30**, 6, 868–875 (2014).

Pyysalo, S., T. Ohta, M. Miwa and J. Tsujii, "Towards exhaustive protein modification event extraction", in "Proceedings of BioNLP 2011 Workshop", pp. 114–123 (2011).

Qian, C., L. Wen, A. Kumar, L. Lin, L. Lin, Z. Zong, J. Wang *et al.*, "An approach for process model extraction by multi-grained text classification", in "International Conference on Advanced Information Systems Engineering", pp. 268–282 (Springer, 2020).

Queiroz, A. L., S. Mckeever and B. Keegan, "Detecting hacker threats: Performance of word and sentence embedding models in identifying hacker communications", in "AICS", (2019).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", arXiv preprint arXiv:1910.10683 (2019).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", The Journal of Machine Learning Research **21**, 1, 5485–5551 (2020a).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", Journal of Machine Learning Research **21**, 140, 1–67, URL `http://jmlr.org/papers/v21/20-074.html` (2020b).

Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, "Squad: 100,000+ questions for machine comprehension of text", in "Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing", pp. 2383–2392 (2016a).

Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text", in "Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing", pp. 2383–2392 (Association for Computational Linguistics, Austin, Texas, 2016b), URL `https://aclanthology.org/D16-1264`.

Ran, Q., Y. Lin, P. Li, J. Zhou and Z. Liu, "Numnet: Machine reading comprehension with numerical reasoning", arXiv preprint arXiv:1910.06701 (2019).

Reddy, S., D. Chen and C. D. Manning, "Coqa: A conversational question answering challenge", arXiv preprint arXiv:1808.07042 (2018).

Reiter, P., H. J. Tay, W. Weimer, A. Doupé, R. Wang and S. Forrest, "Automatically mitigating vulnerabilities in x86 binary programs via partially recompilable decompilation", arXiv preprint arXiv:2202.12336 (2022).

Reitz, K., "Requests: Http for humans, accessed: 2020-10-23", `https://requests.readthedocs.io/en/master/` (2020).

Richardson, K. and A. Sabharwal, "What does my qa model know? devising controlled probes using expert knowledge", Transactions of the Association for Computational Linguistics **8**, 572–588 (2020).

Richardson, L., "Beautiful soup documentation", April (2007).

Richardson, M., C. J. Burges and E. Renshaw, "Mctest: A challenge dataset for the open-domain machine comprehension of text", in "Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing", pp. 193–203 (2013).

Ritsko, J. and S. D.I., "Deduplicating training data makes language models better", IBM Systems Journal, Preface, 3(3) pp. 449—-450 (2004).

Robertson, S. E. and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval", in "SIGIR'94", pp. 232–241 (Springer, 1994).

Roemmele, M., C. A. Bejan and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning", in "2011 AAAI Spring Symposium Series", (2011).

Sabottke, C., O. Suciu and T. Dumitras, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits", in "24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015", edited by J. Jung and T. Holz, pp. 1041–1056 (USENIX Association, 2015), URL https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sabottke.

Saganowski, S., "Cybersecurity NER corpus 2019", in "Harvard Dataverse, V1", (Harvard Dataverse, 2020), URL https://doi.org/10.7910/DVN/1TCFII.

Sakaguchi, K., R. L. Bras, C. Bhagavatula and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale", Commun. ACM **64**, 9, 99–106, URL https://doi.org/10.1145/3474381 (2021).

Sanyal, S., Z. Liao and X. Ren, "Robustlr: Evaluating robustness to logical perturbation in deductive reasoning", arXiv preprint arXiv:2205.12598 (2022a).

Sanyal, S., Y. Xu, S. Wang, Z. Yang, R. Pryzant, W. Yu, C. Zhu and X. Ren, "Apollo: A simple approach for adaptive pretraining of language models for logical reasoning", arXiv preprint arXiv:2212.09282 (2022b).

Sap, M., R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith and Y. Choi, "Atomic: an atlas of machine commonsense for if-then reasoning", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 33, pp. 3027–3035 (2019a).

Sap, M., H. Rashkin, D. Chen, R. LeBras and Y. Choi, "Social iqa: Commonsense reasoning about social interactions", in "EMNLP", (2019b).

Satyapanich, T., F. Ferraro and T. Finin, "CASIE: extracting cybersecurity event information from text", in "The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020", pp. 8749–8757 (AAAI Press, 2020), URL https://aaai.org/ojs/index.php/AAAI/article/view/6401.

Savery, M. E., W. J. Rogers, M. Pillai, J. G. Mork and D. Demner-Fushman, "Chemical entity recognition for medline indexing", AMIA Summits on Translational Science Proceedings **2020**, 561 (2020).

Schankin, A., A. Berger, D. V. Holt, J. C. Hofmeister, T. Riedel and M. Beigl, "Descriptive compound identifier names improve source code comprehension", in "2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)", pp. 31–3109 (IEEE, 2018).

Schlichtkrull, M., T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov and M. Welling, "Modeling relational data with graph convolutional networks", in "European Semantic Web Conference", pp. 593–607 (Springer, 2018).

Schwartz, E. J., J. Lee, M. Woo and D. Brumley, "Native x86 decompilation using semantics-preserving structural analysis and iterative control-flow structuring", in "22nd USENIX Security Symposium (USENIX Security 13)", p. 17 (USENIX Association, 2013).

Seo, M., A. Kembhavi, A. Farhadi and H. Hajishirzi, "Bidirectional attention flow for machine comprehension", arXiv preprint arXiv:1611.01603 (2016).

Shahid, M. R. and H. Debar, "Cvss-bert: Explainable natural language processing to determine the severity of a computer security vulnerability from its description", in "2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)", pp. 1600–1607 (2021).

Shang, J., T. Ma, C. Xiao and J. Sun, "Pre-training of graph augmented transformers for medication recommendation", arXiv preprint arXiv:1906.00346 (2019).

Shibuya, T. and E. Hovy, "Nested named entity recognition via second-best sequence learning and decoding", Transactions of the Association for Computational Linguistics **8**, 605–620 (2020).

Shoshitaishvili, Y., R. Wang, C. Hauser, C. Kruegel and G. Vigna, "Firmalice - automatic detection of authentication bypass vulnerabilities in binary firmware", in "Proceedings 2015 Network and Distributed System Security Symposium", (Internet Society, 2015), URL `https://www.ndss-symposium.org/ndss2015/ndss-2015-programme/firmalice-automatic-detection-authentication-bypass-vulnerabilities-binary-firmware/`.

Shoshitaishvili, Y., R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel *et al.*, "Sok:(state of) the art of war: Offensive techniques in binary analysis", in "2016 IEEE Symposium on Security and Privacy (SP)", pp. 138–157 (IEEE, 2016).

Shoshitaishvili, Y., M. Weissbacher, L. Dresel, C. Salls, R. Wang, C. Kruegel and G. Vigna, "Rise of the hacrs: Augmenting autonomous cyber reasoning systems with human assistance", in "Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security", p. 347–362 (ACM, 2017), URL `https://dl.acm.org/doi/10.1145/3133956.3134105`.

Shu, K., A. Sliva, J. Sampson and H. Liu, "Understanding cyber attack behaviors with sentiment information on social media", in "Social, Cultural, and Behavioral Modeling - 11th International Conference, SBP-BRiMS 2018, Washington, DC, USA, July 10-13, 2018, Proceedings", edited by R. Thomson, C. L. Dancy, A. Hyder and H. Bisgin, vol. 10899 of *Lecture Notes in Computer Science*, pp. 377–388 (Springer, 2018), URL `https://doi.org/10.1007/978-3-319-93372-6_41`.

Si, Y., J. Wang, H. Xu and K. Roberts, "Enhancing clinical concept extraction with contextual embeddings", Journal of the American Medical Informatics Association **26**, 11, 1297–1304 (2019).

Simran, K., S. Sriram, R. Vinayakumar and K. Soman, “Deep learning approach for intelligent named entity recognition of cyber security”, in “International Symposium on Signal Processing and Intelligent Recognition Systems”, pp. 163–172 (Springer, 2019).

Singh, A. K. B., M. Guntu, A. R. Bhimireddy, J. W. Gichoya and S. Purkayastha, “Multi-label natural language processing to identify diagnosis and procedure codes from MIMIC-III inpatient notes”, CoRR **abs/2003.07507**, URL `https://arxiv.org/abs/2003.07507` (2020).

Slowinska, A., T. Stancescu and H. Bos, “Howard: a dynamic excavator for reverse engineering data structures”, in “Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS’11)”, (2011).

Smith, L., L. K. Tanabe, R. J. nee Ando, C.-J. Kuo, I.-F. Chung, C.-N. Hsu, Y.-S. Lin, R. Klinger, C. M. Friedrich, K. Ganchev *et al.*, “Overview of biocreative ii gene mention recognition”, Genome biology **9**, S2, S2 (2008).

Song, H.-J., B.-C. Jo, C.-Y. Park, J.-D. Kim and Y.-S. Kim, “Comparison of named entity recognition methodologies in biomedical documents”, Biomedical engineering online **17**, 2, 158 (2018).

Song, S.-k., H.-s. Oh, S. H. Myaeng, S.-P. Choi, H.-W. Chun, Y.-S. Choi and C.-H. Jeong, “Procedural knowledge extraction on medline abstracts”, in “International Conference on Active Media Technology”, pp. 345–354 (Springer, 2011).

spaCy, “spacy v2.0”, `https://spacy.io/models/en#en_core_web_md` (2017).

Straková, J., M. Straka and J. Hajic, “Neural architectures for nested ner through linearization”, in “Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics”, pp. 5326–5331 (2019).

Suciu, O., C. Nelson, Z. Lyu, T. Bao and T. Dumitras, “Expected exploitability: Predicting the development of functional vulnerability exploits”, CoRR **abs/2102.07869**, URL `https://arxiv.org/abs/2102.07869` (2021).

Sun, K., D. Yu, D. Yu and C. Cardie, “Improving machine reading comprehension with general reading strategies”, CoRR **abs/1810.13441** (2018).

Sun, W., A. Rumshisky and O. Uzuner, “Evaluating temporal relations in clinical text: 2012 i2b2 challenge”, Journal of the American Medical Informatics Association **20**, 5, 806–813 (2013).

Sundararaman, D., S. Si, V. Subramanian, G. Wang, D. Hazarika and L. Carin, “Methods for numeracy-preserving word embeddings”, in “Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)”, pp. 4742–4753 (2020).

Švábenskỳ, V., P. Čeleda, J. Vykopal and S. Brišáková, “Cybersecurity knowledge and skills taught in capture the flag challenges”, Computers & Security **102**, 102154 (2021).

Tabassum, J., M. Maddela, W. Xu and A. Ritter, “Code and named entity recognition in stackoverflow”, in “The Annual Meeting of the Association for Computational Linguistics (ACL)”, (2020).

Tafjord, O., P. Clark, M. Gardner, W.-t. Yih and A. Sabharwal, “Quarel: A dataset and models for answering questions about qualitative relationships”, arXiv preprint arXiv:1811.08048 (2018).

Talmor, A., J. Herzig, N. Lourie and J. Berant, “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)”, pp. 4149–4158 (Association for Computational Linguistics, Minneapolis, Minnesota, 2019), URL `https://www.aclweb.org/anthology/N19-1421`.

Talmor, A., O. Tafjord, P. Clark, Y. Goldberg and J. Berant, “Teaching pre-trained models to systematically reason over implicit knowledge”, arXiv preprint arXiv:2006.06609 (2020).

Thawani, A., J. Pujara, P. A. Szekely and F. Ilievski, “Representing numbers in nlp: a survey and a vision”, arXiv preprint arXiv:2103.13136 (2021).

Tian, J., Y. Li, W. Chen, L. Xiao, H. He and Y. Jin, “Diagnosing the first-order logical reasoning ability through logicnli”, in “Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing”, pp. 3738–3747 (2021).

Touvron, H., T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “Llama: Open and efficient foundation language models”, arXiv preprint arXiv:2302.13971 (2023).

Trask, A., F. Hill, S. E. Reed, J. Rae, C. Dyer and P. Blunsom, “Neural arithmetic logic units”, 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., 2018), URL `https://proceedings.neurips.cc/paper/2018/file/0e64a7b00c83e3d22ce6b3acf2c582b6-Paper.pdf`.

Uzuner, O., A. Bodnari, S. Shen, T. Forbush, J. Pestian and B. R. South, “Evaluating the state of the art in coreference resolution for electronic medical records”, Journal of the American Medical Informatics Association **19**, 5, 786–791 (2012).

Uzuner, Ö., I. Solti and E. Cadag, “Extracting medication information from clinical text”, Journal of the American Medical Informatics Association **17**, 5, 514–518 (2010).

Uzuner, Ö., B. R. South, S. Shen and S. L. DuVall, “2010 i2b2/va challenge on concepts, assertions, and relations in clinical text”, Journal of the American Medical Informatics Association **18**, 5, 552–556 (2011).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, “Attention is all you need”, in “Advances in neural information processing systems”, pp. 5998–6008 (2017).

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph attention networks", arXiv preprint arXiv:1710.10903 (2017).

Votipka, D., S. Rabin, K. Micinski, J. S. Foster and M. L. Mazurek, "An observational investigation of reverse engineers' processes", in "29th USENIX Security Symposium (USENIX Security 20)", p. 1875–1892 (2020).

Wallace, E., Y. Wang, S. Li, S. Singh and M. Gardner, "Do NLP models know numbers? probing numeracy in embeddings", 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)", pp. 5307–5315 (Association for Computational Linguistics, Hong Kong, China, 2019), URL `https://www.aclweb.org/anthology/D19-1534`.

Wang, A., A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding", arXiv preprint arXiv:1804.07461 (2018a).

Wang, C. and H. Jiang, "Explicit utilization of general knowledge in machine reading comprehension", in "Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics", pp. 2263–2272 (2019).

Wang, L., R. Li, Y. Yan, Y. Yan, S. Wang, W. Wu and W. Xu, "Instructionner: A multi-task instruction-based generative framework for few-shot NER", CoRR **abs/2203.03903**, URL `https://arxiv.org/abs/2203.03903` (2022a).

Wang, R., Y. Shoshitaishvili, A. Bianchi, A. Machiry, J. Grosen, P. Grosen, C. Kruegel and G. Vigna, "Ramblr: Making reassembly great again", in "Proceedings 2017 Network and Distributed System Security Symposium", (Internet Society, 2017), URL `https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/ramblr-making-reassembly-great-again/`.

Wang, S., W. Zhong, D. Tang, Z. Wei, Z. Fan, D. Jiang, M. Zhou and N. Duan, "Logic-driven context extension and data augmentation for logical reasoning of text", in "Findings of the Association for Computational Linguistics: ACL 2022", pp. 1619–1629 (Association for Computational Linguistics, Dublin, Ireland, 2022b), URL `https://aclanthology.org/2022.findings-acl.127`.

Wang, X., J. Lyu, L. Dong and K. Xu, "Multitask learning for biomedical named entity recognition with cross-sharing structure", BMC bioinformatics **20**, 1, 427 (2019a).

Wang, X., Y. Zhang, Q. Li, X. Ren, J. Shang and J. Han, "Distantly supervised biomedical named entity recognition with dictionary expansion", in "2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)", pp. 496–503 (IEEE, 2019b).

Wang, X., Y. Zhang, X. Ren, Y. Zhang, M. Zitnik, J. Shang, C. Langlotz and J. Han, "Cross-type biomedical named entity recognition with deep multi-task learning", Bioinformatics **35**, 10, 1745–1752 (2018b).

Wang, Y., S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Naik, A. Ashok, A. S. Dhanasekaran, A. Arunkumar, D. Stap *et al.*, "Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks", in "Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing", pp. 5085–5109 (2022c).

Wang, Y., W. Wang, S. R. Joty and S. C. H. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation", in "Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021", edited by M. Moens, X. Huang, L. Specia and S. W. Yih, pp. 8696–8708 (Association for Computational Linguistics, 2021).

Wei, C.-H., Y. Peng, R. Leaman, A. P. Davis, C. J. Mattingly, J. Li, T. C. Wiegers and Z. Lu, "Overview of the biocreative v chemical disease relation (cdr) task", in "Proceedings of the fifth BioCreative challenge evaluation workshop", vol. 14 (2015).

Wei, J., M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai and Q. V. Le, "Finetuned language models are zero-shot learners", arXiv preprint arXiv:2109.01652 (2021).

Wilensky, R., "Planning and understanding: A computational approach to human reasoning", (1983).

Wilensky, R., D. N. Chin, M. Luria, J. Martin, J. Mayfield and D. Wu, "The berkeley unix consultant project", in "Intelligent Help Systems for UNIX", pp. 49–94 (Springer, 2000).

Wilson, E. B., "Probable inference, the law of succession, and statistical inference", Journal of the American Statistical Association **22**, 158, 209–212 (1927).

Winograd, T., "Understanding natural language", Cognitive psychology **3**, 1, 1–191 (1972).

Wolf, T., J. Chaumond, L. Debut, V. Sanh, C. Delangue, A. Moi, P. Cistac, M. Funtowicz, J. Davison, S. Shleifer *et al.*, "Transformers: State-of-the-art natural language processing", in "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations", pp. 38–45 (2020a).

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing", ArXiv **abs/1910.03771** (2019).

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest and A. M. Rush, "Transformers: State-of-the-art natural language processing", in "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations", pp. 38–45 (Association for Computational Linguistics, Online, 2020b), URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Wu, S., K. Roberts, S. Datta, J. Du, Z. Ji, Y. Si, S. Soni, Q. Wang, Q. Wei, Y. Xiang *et al.*, "Deep learning in clinical natural language processing: a methodical review", Journal of the American Medical Informatics Association **27**, 3, 457–470 (2020).

Xiao, Y., Y. Qu, L. Qiu, H. Zhou, L. Li, W. Zhang and Y. Yu, "Dynamically fused graph network for multi-hop reasoning", arXiv preprint arXiv:1905.06933 (2019).

Xu, F. F., L. Ji, B. Shi, J. Du, G. Neubig, Y. Bisk and N. Duan, "A benchmark for structured procedural knowledge extraction from cooking videos", arXiv preprint arXiv:2005.00706 (2020).

Yadav, V. and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models", in "Proceedings of the 27th International Conference on Computational Linguistics", pp. 2145–2158 (Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018), URL `https://www.aclweb.org/anthology/C18-1182`.

Yadav, V., R. Sharp and S. Bethard, "Deep affix features improve neural named entity recognizers", in "Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics", pp. 167–172 (2018).

Yadegari, B., B. Johannesmeyer, B. Whitely and S. Debray, "A generic approach to automatic deobfuscation of executable code", in "2015 IEEE Symposium on Security and Privacy", pp. 674–691 (IEEE, 2015).

Yakdan, K., S. Dechand, E. Gerhards-Padilla and M. Smith, "Helping johnny to analyze malware: A usability-optimized decompiler and malware analysis user study", in "2016 IEEE Symposium on Security and Privacy (SP)", pp. 158–177 (IEEE, 2016).

Yakdan, K., S. Eschweiler, E. Gerhards-Padilla and M. Smith, "No more gotos: Decompilation using pattern-independent control-flow structuring and semantic-preserving transformations.", in "NDSS", (Citeseer, 2015a).

Yakdan, K., S. Eschweiler, E. Gerhards-padilla and M. Smith, "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantics-Preserving Transformations", in "Network and Distributed System Security", No. February, pp. 8–11 (2015b).

Yamakata, Y., S. Mori and J. A. Carroll, "English recipe flow graph corpus", in "Proceedings of The 12th Language Resources and Evaluation Conference", pp. 5187–5194 (2020).

Yan, H., T. Gui, J. Dai, Q. Guo, Z. Zhang and X. Qiu, "A unified generative framework for various NER subtasks", in "Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021", edited by C. Zong, F. Xia, W. Li and R. Navigli, pp. 5808–5822 (Association for Computational Linguistics, 2021), URL `https://doi.org/10.18653/v1/2021.acl-long.451`.

Yang, L., C. Fang, H. Jin, W. Chang and D. Estrin, “Creative procedural-knowledge extraction from web design tutorials”, arXiv preprint arXiv:1904.08587 (2019a).

Yang, P.-J., Y. T. Chen, Y. Chen and D. Cer, “Nt5?! training t5 to perform numerical reasoning”, arXiv preprint arXiv:2104.07307 (2021).

Yang, Z., Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding”, in “Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada”, edited by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox and R. Garnett, pp. 5754–5764 (2019b), URL https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov and C. D. Manning, “Hotpotqa: A dataset for diverse, explainable multi-hop question answering”, in “Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing”, pp. 2369–2380 (2018a).

Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov and C. D. Manning, “HotpotQA: A dataset for diverse, explainable multi-hop question answering”, in “Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing”, pp. 2369–2380 (Association for Computational Linguistics, Brussels, Belgium, 2018b), URL https://aclanthology.org/D18-1259.

Ye, Z., G. Jiang, Y. Liu, Z. Li and J. Yuan, “Document and word representations generated by graph convolutional network and bert for short text classification”, in “ECAI 2020”, pp. 2275–2281 (IOS Press, 2020).

Ying, Z., J. You, C. Morris, X. Ren, W. Hamilton and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling”, in “Advances in neural information processing systems”, pp. 4800–4810 (2018).

Yoon, W., C. H. So, J. Lee and J. Kang, “Collabonet: collaboration of deep neural networks for biomedical named entity recognition”, BMC bioinformatics **20**, 10, 249 (2019).

Yosinski, J., J. Clune, Y. Bengio and H. Lipson, “How transferable are features in deep neural networks?”, arXiv preprint arXiv:1411.1792 (2014).

Yu, J., B. Bohnet and M. Poesio, “Named entity recognition as dependency parsing”, in “Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics”, pp. 6470–6476 (Association for Computational Linguistics, Online, 2020a), URL https://www.aclweb.org/anthology/2020.acl-main.577.

Yu, J., W. Liu, S. Qiu, Q. Su, K. Wang, X. Quan and J. Yin, “Low-resource generation of multi-hop reasoning questions”, in “Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics”, pp. 6729–6739 (Association for Computational Linguistics, Online, 2020b), URL https://aclanthology.org/2020.acl-main.601.

Yu, W., Z. Jiang, Y. Dong and J. Feng, “Reclor: A reading comprehension dataset requiring logical reasoning”, arXiv preprint arXiv:2002.04326 (2020c).

Zellers, R., Y. Bisk, R. Schwartz and Y. Choi, “Swag: A large-scale adversarial dataset for grounded commonsense inference”, arXiv preprint arXiv:1808.05326 (2018).

Zenebe, A., M. Shumba, A. Carillo and S. Cuenca, “Cyber threat discovery from dark web”, in “Proceedings of 28th International Conference”, vol. 64, pp. 174–183 (2019).

Zhang, J., H. Zhang, L. Sun and C. Xia, “Graph-bert: Only attention is needed for learning graph representations”, arXiv preprint arXiv:2001.05140 (2020a).

Zhang, M. and Y. Chen, “Link prediction based on graph neural networks”, in “Advances in Neural Information Processing Systems”, pp. 5165–5175 (2018).

Zhang, M., Z. Cui, M. Neumann and Y. Chen, “An end-to-end deep learning architecture for graph classification”, in “Thirty-Second AAAI Conference on Artificial Intelligence”, (2018a).

Zhang, X., D. Ramachandran, I. Tenney, Y. Elazar and D. Roth, “Do language embeddings capture scales?”, in “Findings of the Association for Computational Linguistics: EMNLP 2020”, pp. 4889–4896 (Association for Computational Linguistics, Online, 2020b), URL https://www.aclweb.org/anthology/2020.findings-emnlp.439.

Zhang, Y., H. Dai, K. Toraman and L. Song, “Kgˆ 2: Learning to reason science exam questions with contextual knowledge graph embeddings”, arXiv preprint arXiv:1805.12393 (2018b).

Zhang, Z., Y. Ye, W. You, G. Tao, W. Lee, Y. Kwon, Y. Aafer and X. Zhang, “OSPREY: recovery of variable and data structure via probabilistic analysis for stripped binary”, in “42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021”, pp. 813–832 (IEEE, 2021a), URL https://doi.org/10.1109/SP40001.2021.00051.

Zhang, Z., Y. Ye, W. You, G. Tao, W.-c. Lee, Y. Kwon, Y. Aafer and X. Zhang, “Osprey: Recovery of variable and data structure via probabilistic analysis for stripped binary”, in “2021 IEEE Symposium on Security and Privacy (SP)”, p. 813–832 (IEEE, 2021b), URL https://ieeexplore.ieee.org/document/9519451/.

Zhao, J., Q. Yan, J. Li, M. Shao, Z. He and B. Li, “Timiner: Automatically extracting and analyzing categorized cyber threat intelligence from social data”, Comput. Secur. **95**, 101867, URL https://doi.org/10.1016/j.cose.2020.101867 (2020).

Zhao, Y., H. Wan, J. Gao and Y. Lin, “Improving relation classification by entity pair graph”, in “Asian Conference on Machine Learning”, pp. 1156–1171 (2019).

Zheng, B., H. Wen, Y. Liang, N. Duan, W. Che, D. Jiang, M. Zhou and T. Liu, “Document modeling with graph attention networks for multi-grained machine reading comprehension”, in “Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics", pp. 6708–6718 (Association for Computational Linguistics, Online, 2020), URL `https://www.aclweb.org/anthology/2020.acl-main.599`.

Zhong, W., D. Tang, N. Duan, M. Zhou, J. Wang and J. Yin, "Improving question answering by commonsense-based pre-training", in "CCF International Conference on Natural Language Processing and Chinese Computing", pp. 16–28 (Springer, 2019).

Zhu, Y., L. Pang, Y. Lan and X. Cheng, "L2r2: Leveraging ranking for abductive reasoning", arXiv preprint arXiv:2005.11223 (2020).

APPENDIX A

DISSERTATION CONTRIBUTIONS

In Chapter 2, we show how entity-specific explicit knowledge can be extracted from natural language biomedical texts using transformer-based language models with definitions of the entities and their examples learning from 18 publicly available biomedical datasets. In this project, my contributions are: (1) collecting and pre-processing 18 biomedical datasets (2) preparing the collected sequence-labeling NER datasets into single question-answering format (3) all experiments with their 10 different seed values (4) thorough related literature study and understanding the domain of problem (5) all the ablation experiments (5) visualization of the model performance of why our problem formulation can achieve state of the art (6) experiments on nested-entity predictions (7) thorough error-analysis and case-study collections (8) preparation of multiple sections of the manuscript like Introduction, Related works, Methods, Datasets, Template Creation, NER model, Experiments, Results and Discussions.

In Chapter 3, we show how transformer-based language models in combination with Graph Neural Networks can be used to create a structured graph from unstructured public forum texts in the cybersecurity domain. We introduced a novel cybersecurity dataset and pose two challenging tasks: Relevant information extraction and Flow Structure Prediction. In this project, my contributions are: (1) extraction and processing of public forum texts for the annotation process (2) planning and developing the scripts for the collection of annotation for the two tasks (3) post-processing and preparing for the data for the models (4) development of all model architectures (5) execution of all experiments including baseline and our models on COR and CTFW datasets (6) error analysis and model analysis along with collections of case studies (7) thorough related literature study and understanding the domain of problem (8) preparation of multiple sections of the manuscript like Introduction, approach, dataset creation, model description, experiments, analysis, and ablations studies

In Chapter 5, we show that transformer-based language models could use explicitly retrieved instance-specific knowledge to perform multi-step reasoning on Open Book Question-Answering effectively. In this project, my contributions are (1) development and running all experiments on the Copynet Sequence-to-Sequence model in natural language abduction and IR. (2) development, implementation and analysis of novel overlap_score for copynet seq-to-seq approach (3) formal representation of word-symmetric difference approach (4) error analysis and categorization, extracting examples and their presentation (5) preparation of the block diagram of our approach (6) thorough related literature study and understanding the domain of problem (7) Hypothesis generation from question-answering format for each question and answer-options (8) preparation of multiple sections of the manuscript like Hypothesis Generation, Related Works, Copynet Seq-to-Seq model, QA model descriptions, Error Analysis.

In Chapter 6, we demonstrate four modes of knowledge infusion into two transformer-based language models through three training strategies. We also show that infusing external targeted knowledge can help language models to perform commonsense reasoning. In this project, my contributions are (1) a thorough analysis of the Physical IQA (PIQA) commonsense dataset and drawing insights helpful for the projects including bias and distributions (2) finding and pre-processing external knowledge sources relevant to the PIQA dataset (3) setting up local server search engines for PIQA (4) knowledge retrieval for each instance of the dataset (5) execution and analysis of model on PIQA dataset for each of 3 training strategies and 4 knowledge infusion methods. (6) manual error analysis to draw important insights from 200 test samples of PIQA datasets (7) development of end-to-end approach illustration and weights vs lexical overlap figures (8) preparation of multiple sections of the manuscript like Introduction, PIQA Dataset description, Knowledge source preparation, Knowledge retrieval, and results and discussions

APPENDIX B

RELATED PUBLICATION DETAILS

Most ideas in this dissertation have appeared in the following publications, with the consent of my co-authors.

- Banerjee, Pratyay, Kuntal Kumar Pal, Murthy Devarakonda, and Chitta Baral. "Biomedical named entity recognition via knowledge guidance and question answering." ACM Transactions on Computing for Healthcare 2, no. 4 (2021): 1-24.

- Pal, Kuntal Kumar, Kazuaki Kashihara, Pratyay Banerjee, Swaroop Mishra, Ruoyu Wang, and Chitta Baral. "Constructing Flow Graphs from Procedural Cybersecurity Texts." In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 3945-3957. 2021.

- Banerjee, Pratyay, Swaroop Mishra, Kuntal Kumar Pal, Arindam Mitra, and Chitta Baral. "Commonsense reasoning with implicit knowledge in natural language." In 3rd Conference on Automated Knowledge Base Construction. 2021.

- Banerjee, Pratyay, Kuntal Kumar Pal, Arindam Mitra, and Chitta Baral. "Careful Selection of Knowledge to Solve Open Book Question Answering." In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 6120-6129. 2019.

- Pal, Kuntal Kumar, Kazuaki Kashihara, Ujjwala Anantheswaran, Kirby C. Kuznia, Siddhesh Jagtap, and Chitta Baral. "Exploring the Limits of Transfer Learning with Unified Model in the Cybersecurity Domain." arXiv preprint arXiv:2302.10346 (2023).

- Pal, Kuntal Kumar, and Chitta Baral. "Investigating Numeracy Learning Ability of a Text-to-Text Transfer Model." In Findings of the Association for Computational Linguistics: EMNLP 2021, pp. 3095-3101. 2021.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA