University of Kentucky
UKnowledge

Theses and Dissertations--Computer Science

Computer Science

2020

Deep Neural Architectures for End-to-End Relation Extraction

Tung Tran
University of Kentucky, tran8749@gmail.com
Author ORCID Identifier:
https://orcid.org/0000-0001-8166-0681

Digital Object Identifier: https://doi.org/10.13023/etd.2020.153

Right click to open a feedback form in a new tab to let us know how this document benefits you.

Recommended Citation

Tran, Tung, "Deep Neural Architectures for End-to-End Relation Extraction" (2020). *Theses and Dissertations--Computer Science*. 97.
https://uknowledge.uky.edu/cs_etds/97

This Doctoral Dissertation is brought to you for free and open access by the Computer Science at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Computer Science by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student’s advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student’s thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Tung Tran, Student

Dr. Ramakanth Kavuluru, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

Deep Neural Architectures for End-to-End Relation Extraction

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Tung Tran
Lexington, Kentucky

Director: Dr. Ramakanth Kavuluru, Associate Professor of Biomedical Informatics
Lexington, Kentucky 2020

Copyright© Tung Tran 2020
https://orcid.org/0000-0001-8166-0681

ABSTRACT OF DISSERTATION

Deep Neural Architectures for End-to-End Relation Extraction

The rapid pace of scientific and technological advancements has led to a meteoric growth in knowledge, as evidenced by a sharp increase in the number of scholarly publications in recent years. PubMed, for example, archives more than 30 million biomedical articles across various domains and covers a wide range of topics including medicine, pharmacy, biology, and healthcare. Social media and digital journalism have similarly experienced their own accelerated growth in the age of big data. Hence, there is a compelling need for ways to organize and distill the vast, fragmented body of information (often unstructured in the form of natural human language) so that it can be assimilated, reasoned about, and ultimately harnessed. Relation extraction is an important natural language task toward that end. In relation extraction, semantic relationships are extracted from natural human language in the form of (*subject*, *object*, *predicate*) triples such that *subject* and *object* are mentions of discrete concepts and *predicate* indicates the type of relation between them. The difficulty of relation extraction becomes clear when we consider the myriad of ways the same relation can be expressed in natural language. Much of the current works in relation extraction assume that entities are known at extraction time, thus treating entity recognition as an entirely separate and independent task. However, recent studies have shown that entity recognition and relation extraction, when modeled together as interdependent tasks, can lead to overall improvements in extraction accuracy. When modeled in such a manner, the task is referred to as "end-to-end" relation extraction. In this work, we present four studies that introduce incrementally sophisticated architectures designed to tackle the task of end-to-end relation extraction. In the first study, we present a pipeline approach for extracting protein-protein interactions as affected by particular mutations. The pipeline system makes use of recurrent neural networks for protein detection, lexicons for gene normalization, and convolutional neural networks for relation extraction. In the second study, we show that a multi-task learning framework, with parameter sharing, can achieve state-of-the-art results for drug-drug interaction extraction. At its core, the model uses graph convolutions, with a novel attention-gating mechanism, over dependency parse trees. In the third study, we present a more efficient and general-purpose end-to-end neural architecture designed around the idea of the "table-filling" paradigm; for an input sentence of length n, all entities and relations are extracted in a single pass of the network in an indirect fashion by populating the cells of a corresponding $n \times n$ table using metric-based features. We show that this approach excels in both the general English and biomedical domains with extraction times that are up to an order of magnitude faster compared to the prior best. In the fourth and last study, we present an architecture for relation extraction that, in addition to being end-to-end, is able to handle cross-sentence and N-ary relations. Overall, our work contributes to the advancement of modern information extraction by exploring end-to-end solutions that are fast, accurate, and generalizable to many high-value domains.

KEYWORDS: Machine Learning, Deep Neural Networks, Natural Language Processing, Information Extraction, Relation Extraction

Author's signature: Tung Tran

Date: May 7, 2020

Deep Neural Architectures for End-to-End Relation Extraction

By
Tung Tran

Director of Dissertation: Ramakanth Kavuluru

Director of Graduate Studies: Mirosław Truszczyński

Date: May 7, 2020

Dedicated to my parents, Thu and Vũ Trần, to my sister Quyên Trần,
and to the memory of my cousin Thomas Lê.

ACKNOWLEDGMENTS

This dissertation is the result of the support and guidance of many people to whom I owe my gratitude. First and foremost, I would like to thank my dissertation advisor, Dr. Ramakanth Kavuluru, for his phenomenal and careful guidance during the course of my academic journey. His exemplary sense of integrity and commitment to scholarship has inspired me to set higher standards and expectations for myself. I would also like to thank Dr. Halil Kilicoglu for providing me the opportunity to do research at the U.S. National Library of Medicine, for his support and guidance, and for allowing me the time and freedom to work on projects of my choosing, including one that led to one of the studies presented in this dissertation. I am also grateful to Dr. Mirek Truszczyński for his early mentorship, which has had a strong influence in my decision to pursue academic research, and for his ongoing support. I would also thank Dr. Yuliya Lierler for seeing potential in me and for providing me with my first research opportunity in my early undergraduate years. I also wish to thank my peers (lab mates) for their help as a source of advice and constructive criticism. Finally, I would like to thank all members of my dissertation committee, and outside examiner, respectively: Dr. Ramakanth Kavuluru, Dr. Mirek Truszczyński, Dr. Jane Hayes, Dr. Subbarao Bondada, and Dr. Kunlei Liu.

Beyond the technical aspect of the dissertation, I would first and foremost like to thank my parents, Thu Trần and Vũ Trần, for their unwavering love, support, and encouragement throughout the course of my academic career. They taught me to be responsible for myself and to choose my own path, and provided me with full support without any pressure or expectations. Thanks to my sister Quyên Trần and her husband Jerry Lu for always having such a big presence in my life through all the highs and lows. I am also grateful to my uncle Bình Trần and his family for providing

what was essentially a second home to me, including the daily creature comforts of a home-cooked meal, during the last leg of my academic journey. Of course, I would also like to acknowledge the collective love and support from the many other figures in my life, including my grandparents, cousins, other close relatives, and life-long friends.

TABLE OF CONTENTS

Acknowledgments iii

Table of Contents v

List of Tables vii

List of Figures ix

Chapter 1 Introduction 1
- 1.1 Thesis Statement 3
- 1.2 Organization and Related Publications 3

Chapter 2 Background and Related Works 5
- 2.1 End-to-End Relation Extraction 5
  - 2.1.1 A Typical Pipelined Approach 6
  - 2.1.2 Evaluation Metrics 7
- 2.2 Literature Review 8
  - 2.2.1 Deep Neural Networks 8
  - 2.2.2 Relation Classification 10
  - 2.2.3 End-to-End Relation Extraction 11
- 2.3 Notations and Neural Building Blocks 14

Chapter 3 Deep Learning for Extracting Protein-Protein Interactions 17
- 3.1 Background and Related Work 19
  - 3.1.1 Biomedical Relation Extraction 19
  - 3.1.2 Top Performing PPIm Extraction Entry 20
- 3.2 Materials and Methods 21
  - 3.2.1 PPIm Dataset 21
  - 3.2.2 Gene Mention Identification (NER) 21
  - 3.2.3 Entrez Gene ID Normalization (GN) 27
  - 3.2.4 Relation Classification of Gene Pairs (RC) 29
- 3.3 Results and Discussion 30
- 3.4 Conclusion 36

Chapter 4 Graph Convolutions for Extracting Drug Interaction Information . 37
- 4.1 Background and Related Work 40
  - 4.1.1 Relation Extraction for DDI 40
  - 4.1.2 TAC 2018 DDI Track 41
- 4.2 Materials and Methods 42
  - 4.2.1 Task Description 42
  - 4.2.2 Joint Modeling Approach 43

4.2.3 Neural Network Architecture and Training Details 45
4.2.4 Transfer Learning with Network Pre-Training 52
4.2.5 Voting-based Ensembling 53
4.2.6 Model Evaluation 54
4.3 Results and Discussion 56
4.4 Error Analysis 58
4.5 Conclusion 61

Chapter 5 Neural Metric Learning for Fast End-to-End Relation Extraction . 62
5.1 Background and Related Work 64
5.2 Methodology 64
5.2.1 The Table-Filling Problem 64
5.2.2 Our Model: Relation-Metric Network 67
5.2.3 Decoding the Output 71
5.3 Experimental Setup 72
5.3.1 Evaluation Metrics 72
5.3.2 Datasets 72
5.3.3 Model Configuration 73
5.4 Results and Discussion 74
5.4.1 Ablation Analysis 76
5.4.2 Error Analyses 77
5.5 Conclusion 83

Chapter 6 End-to-End Extraction of Cross-Sentence N-ary Relations 85
6.1 Related Work 87
6.2 Methodology 88
6.2.1 Neural Network Architecture 88
6.2.2 Ternary Relation Extraction 89
6.2.3 Training Procedure 90
6.2.4 Model Ensembling 90
6.3 Experimental Setup 91
6.3.1 JAX-CKB Dataset 91
6.3.2 Evaluation Method 91
6.3.3 Model Configuration 92
6.4 Results and Discussion 93
6.5 Conclusion 101

Chapter 7 Conclusion 102
7.1 Contributions 102
7.2 Limitations and Future Work 103

Abbreviations 105

Bibliography 107

Vita 120

LIST OF TABLES

3.1 System performance on the official test set 31
3.2 Iterative component-level analysis on the official test set 34

4.1 Characteristics of various datasets 39
4.2 Example of the sequence labeling scheme for the sentence in Figure 4.1, where `LABELDRUG` is substitute for Adenocard. 43
4.3 Model configuration obtained through random search over 11-fold cross-validation of TR22 (training data). 51
4.4 Main results based on 95% confidence interval around mean precision, recall, and F1 based on evaluating N=100 ensembles for each model. . . . 55
4.5 Comparison of our method with comparable (based on training data) methods of teams in the top 5 trained on solely TR22 + NLM180. 56
4.6 Confusion matrix for interaction type 61

5.1 Model configuration as tuned on the CoNLL04 development set. 73
5.2 Results comparing to other methods on the CoNLL04 dataset. We report 95% confidence intervals around the mean F1 over 30 runs for models in the last two rows. Our model was tuned on the CoNLL04 development set corresponding to the configuration from Table 5.1. 74
5.3 Results comparing to other methods on the ADE dataset. We report the mean performance over 10-fold cross-validation for models in the last two rows. Our model was tuned on the CoNLL04 development set corresponding to the configuration from Table 5.1. 75
5.4 Ablation studies for relation extraction over the CoNLL04 and ADE dataset; each row after the first indicates removal of a particular feature/component. 76
5.5 Relation extraction performance partitioned based on “Entity Distance”, which is defined as the *number of characters* separating the subject and object entities (i.e., absolute character offset). 80
5.6 Relation extraction performance on the CoNLL04 dataset partitioned based on relation type. 81

6.1 Characteristics of the JAX-CKB dataset 92
6.2 Results on the test set for various models when learning and predicting at the sentence level, and when also evaluating at the sentence level. The “N-ary Relations” column are results from evaluating on a test set containing both binary and ternary relations, while the “Only Binary” and “Only Ternary” columns evaluates on binary relations and ternary relations exclusively. 94

6.3 Our main results on the test set for various models and various levels of discourse when evaluating on both intra- and inter-sentence relations (at the paragraph level). Results from Table 6.2, in which models learn and predict at the sentence level, are included after adjusting for inter-sentence relations — by penalizing recall based on missed inter-sentence relations — so that all displayed results are *directly comparable*. 96
6.4 Entity recognition results for the N-ary with ensembling model based on entity type. 98
6.5 Relation extraction results for the N-ary with ensembling model based on relation type. 98

LIST OF FIGURES

3.1 Deep neural network architecture of the NER model 22
3.2 Network architecture of the RC model (adapted from Tran and Kavuluru [1, Figure 1]) 29
3.3 Visualization of decisions made by the final system on article with PMID 23897824 33

4.1 An example illustrating the end-to-end DDI extraction task. We first (1) identify mentions including precipitants; for each precipitant, we (2) determine the type of interaction and, based on interaction type, (3) determine the interaction outcome. In the case of PD interactions, the outcome corresponds to one of the previously identified *effect* spans. 38
4.2 Overview of the neural network architecture for a simplified example from the drug label Adenocard. Here, the ground truth indicates that *digitalis* is a pharmacodynamic precipitant associated with the effect "ventricular fibrillation." The PK predictive component is omitted given there are no precipitants involved in a PK interaction. 44
4.3 An example sentence from the drug label for Savella along with the resulting prediction and ground truth labels. Red arrows indicate interaction outcome. 58
4.4 An example sentence from the drug label for Aubagio along with the resulting prediction and ground truth labels. Red arrows indicate interaction outcome, where C54357 is a PK label corresponding to the NCI Thesaurus code for "Increased Concomitant Drug Level." 59

5.1 A simple relation extraction example. 63
5.2 Table representation for the example in Figure 5.1. BILOU-encoded entity tags are assigned along the diagonal and relation tags are assigned where entity spans intersect. Empty cells are implicitly assigned the O tag. . . . 65
5.3 Overview of the network architecture for $\lambda = 2$. For simplicity, we ignore punctuation tokens. 66
5.4 2D convolution on 3D input with padding 70
5.5 Mean F1-score (over 10 runs) on CoNLL04 development set with respect to number of training epochs for various embedding training strategies. . 77
5.6 Entity and relation extraction performance with respective to change in maximum sentence length for CoNLL04. 78
5.7 Entity and relation extraction performance with respect to change in maximum sentence length for ADE. 79

5.8 Visualization of activity of pooling layers at various depths (L^i for $i = 1, \ldots, \lambda$), as tabular heatmaps, for a network with a depth of $\lambda = 8$ given the following input sentence: “In 1964, a jury in Dallas found Jack Ruby guilty of murdering Lee Harvey Oswald, the accused assassin of President Kennedy.” Here, we measure activity by sum-pooling the activations along the channel dimension of each hidden representation. For the prediction activity, we simply max-pool probabilities along the relation dimension thus ignoring the exact *type* of entity or relation. 82

6.1 Error analysis on an example paragraph, with recall-related errors, appearing in the article with PMID 27523909. The entire input with model-annotated entities appear on the left side, where mentions of the same concept share the same color. On the top right, we present binary predictions and highlight missed binary relations. On the bottom right, we show predictions of ternary candidates and the final output evaluated based on ground truth information. 99

Chapter 1 Introduction

As part of our transition into the information age, there is a growing wealth of information in the form of unstructured text. In news data, we have millions of articles relating people, places, and organizations published on a daily basis. Meanwhile, PubMed archives more than 30 million research articles and continues to archive an additional 500,000 articles per year. On the Twitter platform alone, participants share commentaries ("tweets") at a rate of *one-half of a billion* per day. These perpetually accumulating bodies of knowledge are rich in valuable information, encoded in natural human language, and remain relatively untapped. The sheer scale of textual data prevents any one person from being able to make meaningful use of encoded information in aggregate. Unlike human beings, machines are capable of such an endeavor. That is, we can build machines capable of organizing and distilling the vast, fragmented body of information so that it can be assimilated, reasoned about, and ultimately harnessed. Such efforts, under the broad theme of knowledge discovery, fundamentally rely on being able to recognize and capture semantic relations as conveyed in natural language — hence the importance of relation extraction (RE) systems. The task of RE is simple: given some textual input, extract (*subject*, *predicate*, *object*) triples where the *subject* and *object* are entities and the *predicate* is an instance of a semantic relation. For example, (*insulin*, *treat*, *diabetes type 1*) is a triple representing a relationship that can be extracted from the sentence:

> "Insulin is prescribed for the treatment of Diabetes Type 1."

A natural source of difficulty is how the same semantic relationship can be conveyed in a variety of complex ways while having the same intended meaning; e.g.,

> "Patients with Type 1 Diabetes are prescribed insulin."

Intuitively, semantic relations that are conveyed in active voice are easier for humans and rule-based machines to detect, while relations presented in passive voice are slightly more difficult. Sentences that are overly complex or where the entities of the relations are far apart can also prove to be difficult cases for a machine-based RE system. Another issue arises from how entities are represented, as it is not uncommon that entities are represented not as discrete words but rather spans of text. Moreover, a sentence may contain multiple relations and there may be relations that are expressed across sentence bounds; both cases add to the complexity of the problem.

Ultimately, the goal of relation extraction systems is to extract structured assertions relating entities corresponding to discrete concepts in a way that is ideally both human-readable and machine-processable. Once obtained, these so called relations can be valuable in and of themselves or as serve as valuable input to other user-end systems. For example, doctors may use such systems to survey for new drug-disease relations for off-label prescription of a particular drug. Or, this information may be used to construct new or populate existing knowledge databases and ontologies in certain specialized domains. This is important in precision medicine, for instance, wherein treatments are individualized based on a patient's genetic profile. In such cases, knowledge bases containing up-to-date information about relations between drugs, genes, and mutations, is an invaluable resource for medical researchers and professionals as they ponder treatment options. Freshly extracted relations can also serve as crucial input to end-systems including information retrieval, question-answering, automatic summarization, and knowledge discovery systems.

Typically, works in RE the presume that entities are pre-identified, and the relation extraction aspect is mostly limited to classification of these known entity pairs as either positive or negative for a relation — we refer to such tasks more concisely as *relation classification*. RE of the *end-to-end* ($\mathcal{E}2\mathcal{E}$RE) variety is a complex multi-stage natural-language task typically involving an additional named entity recognition (NER) step. That is, all entities are identified via NER before pairs of entities are classified for a relationship. Much of the current research in RE treat NER and relation classification as independent subtasks, with many focusing exclusively on the relation classification aspect with the simplifying assumption that entities are known during test time. Granted, RE systems resulting from such studies are modular with the benefit that innovations to each subtask can be achieved independently and later combined in an ad-hoc fashion. However, there is evidence to suggest that there are correlations between NER and relation classification — at a semantic level — that can be exploited for improved overall performance [2, 3, 4, 5]. In this dissertation, we present four studies that introduce incrementally sophisticated architectures designed to tackle the task of $\mathcal{E}2\mathcal{E}$RE across a wide-array of applications in the biomedical, healthcare, and general English domain. Overall, this work contributes to the advancement of modern information extraction by innovating on end-to-end solutions that are fast, accurate, and highly applicable to specialized domains.

1.1 Thesis Statement

Relation extraction is important because it facilitates knowledge discovery from the growing body of unstructured text encoded in natural language. We propose to study various deep learning architectures toward *end-to-end* relation extraction wherein both entities and their relationships are identified in a *coordinated* manner – for example, by modeling the subtasks jointly. We hypothesize that such an approach is able to better leverage inter-task correlations between NER and relation classification, which would otherwise be ignored in a traditional ad hoc pipeline system, resulting in non-trival gains in relation extraction tasks. Furthermore, we hypothesize that additionally extending end-to-end approaches to account for cross-sentence and N-ary relations will significantly improve recall and thus overall model accuracy.

1.2 Organization and Related Publications

The remaining chapters of this dissertation are organized as follows.

Chapter 2 introduces the task of relation extraction and presents pertinent background information important for staging the rest of the dissertation. Here, we concretely define the problem of end-to-end relation extraction, describe relevant nuances, and provide an example of a typical pipeline approach to such problems. This chapter additionally includes a review of the literature on deep neural networks in general, and its application to the problem of relation extraction. This chapter concludes with a description of mathematical notations used in remaining chapters of the dissertation; additionally provided are definitions of standard neural architectures referenced as abstract building blocks in the *methodology* section of studies described in subsequent chapters.

Chapter 3 presents the first study focused on extracting protein-protein interactions from biomedical literature. Here, we explore a pipeline approach that exemplifies the typical end-to-end modeling approach. The pipeline system makes use of recurrent neural networks for protein detection, lexicons for gene normalization, and convolutional neural networks for relation extraction. While we do not model NER and RC jointly, we show that our coordinated end-to-end approach improves over prior works that focus solely on optimizing relation classification and rely exclusively on existing tools for entity recognition.

Chapter 4 presents the second study focused on extracting drug-drug interactions from drug labels. Such problems are highly specialized, and require a complex multi-stage system for extracting not only drug entities and interactions, but also the outcome of each interaction. In this study, we proposed a multi-task learning framework, with joint learning via parameter sharing, involving a novel attention-gated graph convolution over dependency parse trees. We show that this approach is able to achieve state-of-the-art results over prior work on comparable FDA datasets.

Chapter 5 presents the third study of this thesis focused on devising an approach capable of jointly extracting both entities and relations without the need for local classifiers. Here, we present an efficient and general-purpose end-to-end neural architecture designed around the idea of the "table-filling" paradigm; for an input sentence of length N, all entities and relations are extracted in a single pass of the network in an indirect fashion by populating the cells of a corresponding $N \times N$ table using metric-based features. We show that this approach excels in both the general English and biomedical domains with extraction times that are an order of magnitude faster compared to prior methods.

Chapter 6 presents the fourth and final study of this dissertation. Here, we present an architecture for relation extraction that, in addition to being *end-to-end*, is able to handle relations that are N-arity and expressed over multiple sentences. We achieve this by leveraging the speed and efficiency of the model introduced in Chapter 5; we manage potential scalability issues by learning higher-arity relations by representing them through their constituent binary subrelations. We show that by additionally accounting for cross-sentence and higher-arity relations, we improve over baseline approaches predominantly owing to substantial gains in recall.

Chapter 7 concludes the dissertation by summarizing important contributions and highlighting potential avenues for future work.

This dissertation contains materials previously published in the following study:

T. Tran and R. Kavuluru. An End-to-End Deep Learning Architecture for Extracting Protein-Protein Interactions Affected by Genetic Mutations. *Database: Journal of Biological Databases and Curation*, 2018.

Chapter 2 Background and Related Works

2.1 End-to-End Relation Extraction

In this section, we concretely formulate the problem of RE. Let $\mathcal{E}$ be a set of entities and $\mathcal{R}$ be a set of semantic relation types for some target application domain. A directed binary relation is defined as a triple $(\alpha, \rho, \beta) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where $\alpha, \beta \in \mathcal{E}$ are the subject/object entities and $\rho \in \mathcal{R}$ is a predicate describing the relationship. As a digression, it is possible to have n-ary relationships where there are n participating entities. In most chapters of this work, we focus on the specific case where there are two participating entities. We explore N-ary relation extraction in Chapter 6 exclusively. In the literature, relation extraction can refer to one of two related but separate problems. In this dissertation, we make the distinction by referring to one as *relation classification* (RC) and the other as $\mathcal{E}2\mathcal{E}$RE. In RC, the input is a body of text pre-annotated with entity offsets and the output is a relation $r \in \mathcal{R}$. Here, the subject and object entities are known beforehand and included as input to the classification problem. In $\mathcal{E}2\mathcal{E}$RE, the input is a body of text and the goal is to exhaustively identify and return all triples corresponding to a relationship. In this case, the entities, if any, are not known beforehand. Typically, a solution to the RE problem will involve a combination of named entity recognition (NER) and relation classification (RC). Thus, in some cases, an RC system may serve as a subcomponent of an $\mathcal{E}2\mathcal{E}$RE system. In some specialized domains, there will also be an additional entity normalization component responsible for linking an entity mention to a unique identifier; for example, mentions of genes may be mapped to unique gene IDs. In this work, we generally do not address the problem of entity normalization except in cases where it is an integral part of the domain problem being studied. This is due to the fact that entity normalization can exist as an ad-hoc and independent post-processing step with no bearing on the $\mathcal{E}2\mathcal{E}$RE system. Interestingly, many studies on relation extraction focus solely on the RC aspect. However, when treated as a sub-task of $\mathcal{E}2\mathcal{E}$RE, RC will typically include an additional *null* relation type indicating the lack of a relationship. In RC, the *null* relation type may or may not be included. In Section 2.1.1, we describe a typical pipeline setup for document-level relation extraction. Lastly, in Section 2.1.2, we describe standard evaluation metrics for $\mathcal{E}2\mathcal{E}$RE.

2.1.1 A Typical Pipelined Approach

A typical relation extraction pipeline is composed of an NER and an RC component. Many $\mathcal{E}2\mathcal{E}$RE tasks in technical domains also mandate an entity normalization (EN) component as part of the NER process. EN refers to the mapping of entity mentions to domain-specific identifiers. Clearly, the entity normalization aspect adds another source of error that can propagate to the pipeline output. If we map an entity to an incorrect identifier, all triples extracted involving said entity will be treated as false positives regardless of how well the NER component performs. We proceed to describe a common $\mathcal{E}2\mathcal{E}$RE approach with a few simplifying assumptions in mind. First, that EN is either perfect or irrelevant. Second, that no cross-sentence relationships exist. And lastly, we assume that each entity corresponds to a single contiguous span in the input text; this is usually — but not always — the case.

In NER, the goal is to identify spans of text corresponding to *named entities*, or proper names of people, places, or objects such as "George Washington" or "Futurama". NER has been traditionally modeled as a sequence-to-sequence tagging problem such as in part-of-speech tagging and machine translation. The input is a sequence of tokens (these are symbols/words forming a sentence) and the output is a corresponding sequence of labels demarcating entity bounds. It is important to note that it is common for a single named entity to span multiple words, so a binary tagging scheme is insufficient. There has been many tagging schemes proposed, with the most popular among them being the IOB (inside, outside, beginning) and IOBES (inside, outside, beginning, end, and single) scheme. These sequence-tagging labels usually serve as prefixes to an entity type such as a person or location. In the IOB tagging scheme, the B label indicates the beginning of a tag, the I label indicates the inside of a tag, and the O label indicates the outside of a tag. As a rule, the B label is only used to indicate the beginning of a new tag when following a tag of the same entity type. This allows for the distinction of entity spans that are adjacent but share the same entity type. The IOBES scheme is a more fine-grained extension of IOB that also happens to be more intuitive and has generally been shown to perform better.

For an input document x, we extract a set of entities $\mathcal{S}_x \subset \mathcal{E}$ using an NER model. Once all entities are identified (and mapped to their IDs, if applicable), the next step is to classify pairs according to their relation type. To that end, a classifier is build to assign each pair in $\mathcal{P} = \{(\alpha, \beta) | (\alpha, \beta) \in \mathcal{S}_x \times \mathcal{S}_x\}$ a relation from $\mathcal{R}$. In this problem scenario, $\mathcal{R}$ includes the *null* relation type which indicates the lack of a relationship. The output for x are all pairs in $\mathcal{P}$, along with their relation assignment, that were not assigned the *null* relation.

2.1.2 Evaluation Metrics

In end-to-end relation extraction, we use evaluate entity recognition and relation extraction separately. However, relation extraction performance directly subsumes entity recognition performance, and thus can be viewed as a holistic measure, given the heavy reliance of relation extraction on entity recognition performance. In both cases, however, we use the standard F1 score as a single evaluation metric as opposed to the more popular *accuracy* metric. While accuracy is an acceptable metric in cases where the number of positive to negative examples is balanced, he F1 score is preferable when there is sparsity in the prediction space as is associated with information extraction tasks. That is, the number of *candidate* entities and relations are typically high for any input example, and the number of any of those entity or relation being *relevant* is much smaller by comparison. In such cases, a model that predicts nothing (that is, no entities or no relations) will have a misleadingly high accuracy owing to a high number of true negatives. The F1 score is not inflated by true negatives, and is thus a better choice for information extraction problems. The F1 score is actually a summary metric that is based on an ideal balance of the *precision* and *recall* measures often seen in NLP and information retrieval tasks. Precision (or *positive predictive value*) is the fraction of predicted entity or relation that are correctly predicted, or

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \ ,$$

while recall (or *sensitivity*) is the fraction of all correct entity or relation appearing in the prediction set, or

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \ ,$$

where TP, FP, and FN are true positives, false positives, and false negatives. Concretely, the F1 score is defined as the harmonic mean of the precision and recall,

$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Thus, precision and recall, and thereby the F1 score, ignores true negatives completely.

Micro- and macro-averaged F1. In the studies explored in this work, the F1 scores are implicitly *micro*-averaged over classes. That is, we compute F1 in way that weights all discrete predictions equally; relations or entities of different types are weighted proportion to the frequency of each type. This is opposed to *macro*-averaging, wherein each type is weighted equally without regard for their relative frequency. Concretely, for a set of relations $\mathcal{R}$ relations, the micro-averaged F1 score is the harmonic mean of

$$\text{micro precision} = \frac{\sum_r \mathrm{TP}_r}{\sum_r \mathrm{TP}_r + \sum_r \mathrm{FP}_r} \quad \text{and} \quad \text{micro recall} = \frac{\sum_r \mathrm{TP}_r}{\sum_r \mathrm{TP}_r + \sum_r \mathrm{FN}_r},$$

where TP_r, FP_r, and FN_r are the true positive, false positive, and false negative counts, respectively, for relation r. Or more concretely,

$$\text{micro F1} = \frac{2 \cdot \text{micro precision} \cdot \text{micro recall}}{\text{micro precision} + \text{micro recall}}. \tag{2.1}$$

With *macro*-averaging, F1 scores are computed per class, and then averaged.

While the above describes a generic formulation for the micro-F1 score, each self-contained study in this work will typically contain a more tailored definition depending on the problem or dataset.

2.2 Literature Review

This section serves as a literature review of the research space on the topic of deep learning as well as both relation classification (RC) and end-to-end relation extraction ($\mathcal{E}2\mathcal{E}$RE). We first provide an overview of advances in deep neural networks for natural language tasks in Section 2.2.1. In Section 2.2.2, we discuss early works on relation classification as well as recent state-of-the-art neural models. In Section 2.2.3, we review studies that focus on $\mathcal{E}2\mathcal{E}$RE; here, we explore early works using inference mechanisms as well as recent end-to-end neural models.

2.2.1 Deep Neural Networks

Recent progress in natural language processing (NLP) in general has mostly been a consequence of advances in *deep* neural networks – neural networks with at least two layers between the input and output layer and capable of composing useful intermediate representations. Relevant deep neural architectures are introduced in the remainder of this section.

Neural Word Embeddings and Convolutional Neural Networks. Convolutional neural networks (CNNs) in particular were originally developed for image recognition tasks [6] and has been successfully applied to the text domain by exploiting so called neural word embeddings [7, 8]. These word embeddings represent words as vectors and can be pre-trained using unsupervised methods and further trained when learning on a specific task. CNNs exhibit translational invariance, which allows them to detect contextual features while being insensitive to changes of a translational nature. In the context of computer vision, this feature allows a model to recognize an object regardless of whether it has been rotated or rescaled, for example. In text classification, this translates to being able to recognize meaningful contextual terms or phrases while accounting for the many distinct ways the same sentence can be expressed syntactically. Moreover, the so called pooling operation (more later) that is intrinsic to CNNs makes it possible to deal with variable-length nature of text. Using CNNs along with neural word embeddings has been shown to be effective in many natural language tasks (including text classification and relation extraction) since they naturally capture syntactic and semantic information [9, 10, 11].

Recurrent Neural Networks. Unlike CNNs, which are a feedforward type of network, recurrent neural networks (RNNs) have also been successful in NLP tasks involving sequence data such as part-of-speech tagging, named entity recognition (NER), and machine translation [12, 13]. RNNs are a natural architecture for modeling sequences via cyclical connections in the network such that outputs from previous timesteps are fed back as input to the network. It is typical to compose RNNs in both the forward and the backward direction as this allows the sequence to be modeled in both directions in a joint architecture called a bidirectional RNN (Bi-RNN). In a typical Bi-RNN architecture, both the forward and backward RNN receive the same input and are composed independently; once composed, the output vector is concatenated at each corresponding timestep. Bi-RNNs are important for sequence labeling tasks as the full context is taken into account when assigning a label for each timestep of the input sequence. Bi-RNNs in a text classification setting can also be viewed intuitively as a composition of the entire context (in vector form) centered at some word with as many "context vectors" as there are words in the sequence.

Long Short-Term Memory Networks. A significant issue with traditional RNNs is the problem of *vanishing gradients* [14] where the back propagated errors that are needed to update the parameters become extremely small for earlier layers (in the cyclical layer unfolding) due to the application of the familiar chain rule in computing derivates that involve functions of functions. The effect of this learning becomes extremely slow and may be ineffective overall. This effect increases the deeper the network is, which in case of RNNs amounts to the maximum length of the input sequences. To counter this in RNNs, one popular idea is to use a more involved hidden layer with the so called *long short-term memory* (LSTM) units [15, 16]. Unlike in a traditional RNN, in LSTMs, the state representation includes an explicit *memory cell* access to and use of which is controlled through three *gates* – first to control how much of the next input to incorporate in the memory (input gate), second to determine to what extent the current memory is to be forgotten (forget gate), and third to limit the extent of information from the current step's output to propagate to the next step (output gate). These three gates control the flow of information based on the previous output and cell state via sigmoid outputs in $[0, 1]$. We encourage readers to refer to Graves [17, Chapter 4] and Goldberg [18, Section 11] for thorough details of LSTMs and the corresponding derivations of gradients. In this study, we used Bi-RNNs with LSTM units in the hidden layer which are simply termed Bi-LSTMs.

2.2.2 Relation Classification

The majority of past and current efforts in relation extraction treat the problem as a simpler *relation classification* problem where pairs of entities are known during test time; the goal is to classify the pair of entities, given the context, as being either positive or negative for a particular type of relation. Many early works on relation classification preprocess the input as a dependency parse tree [19, 20] and exploit features corresponding to the shortest dependency path between candidate entities; this general approach has also been successfully applied in the biomedical domain [21, 22, 23, 24], where they typically involve a graph kernel based Support Vector Machine (SVM) classifier [23, 25]. Such models are limited in that they rely on a capable dependency parser to pre-process the input and are prone to errors that can propagate from input pre-processing step. The concept of network centrality has also been applied [24] such that gene networks were created with respect to a specific disease; genes are then ranked according to network centrality metrics where highly ranked genes were considered more likely to be associated with the disease. Other studies, such as the effort by Frunza et al. [26], apply the more traditional *bag-of-words* approach focusing on syntactic and lexical features while exploring a wide variety of classification algorithms including decision trees, SVMs, and Naïve Bayes.

Deep Neural Models. More recently, innovations in relation extraction have centered around designing meaningful deep learning architectures. Liu et al. [27] proposed a dependency-based CNN architecture wherein the convolution is applied over words adjacent according to the shortest path connecting the entities in the dependency tree, rather than words adjacent with respect to the order expressed, to detect drug-drug interactions (DDIs). In Kavuluru et al. [28], ensembling of both character-level and word-level RNNs is further proposed for improved performance in DDI extraction. Raj et al. [29] proposed a deep learning architecture such that word representations are first processed by a bidirectional RNN layer followed by a standard CNN, with an optional attention mechanism towards the output layer. Luo et al. [30] proposed convolving over not only the sentence, but rather over the five segments of a sentence: before the first entity mention, the first entity mention, in between the entity mentions, the second entity mention, and after the second entity mention. A single representation of the candidate relation and its context are then composed via simple concatenation of the CNN outputs. Zhang et al. [31] showed that relation extraction performance can be improved by applying graph convolutions over a pruned version of the dependency tree.

2.2.3 End-to-End Relation Extraction

Early efforts in $\mathcal{E}2\mathcal{E}$RE, as covered in this section, assume that entity bounds are known during test time. Hence, the NER aspect of these methods is limited to classifying entity type (e.g., is "President Kennedy" a person, place, or organization?). In an early seminal study, Roth and Yih [2] proposed an integer linear programming (LP) approach to tackle the end-to-end relation extraction problem[1]. The LP component, which takes independent local classifier probability outputs as input, is used to enhance classification outputs by enforcing relational constraints at a global level through a so called *global inference* mechanism. The task of joint extraction is then reduced to finding a most-probable joint assignment of entities and relations. Hence, it is possible for the results of an relation classifier to affect the results of the NER component, which is otherwise impossible in a traditional pipeline approach. A benefit to formulating RE as an LP problem is that it is relatively fast using commercial LP packages despite hardware limitations at the time; moreover, it is more efficient

---

[1] In early works, end-to-end relation extraction for the most part assume that entity bounds are known during test time. The joint modeling aspect is really then to 1. assign entity types to known entity mentions along and 2. assign relation types for each candidate pair of entities. It is therefore a much easier task than “end-to-end relation extraction” as defined in this study, where we must additionally identify entity bounds.

than computing these relationships independently. They experimented on the TREC dataset (corpus from Wall Street Journal, Associate Press, etc.) that were annotated with entities and their relationships. They discovered that the LP component was effective in enhancing classifier results by reducing semantic inconsistencies in the predictions compared to a traditional pipeline approach wherein the outputs of an NER are passed as features into the relation classification component. This indicates that there are mutual inter-dependencies between NER and RC as subtasks which can be exploited. The LP technique has been also been successfully applied in jointly model entities and relations with respect to opinion recognition by Choi et al. [32]. Kate and Mooney [33] proposed a similar approach but presented a global inference mechanism induced by building a graph resembling a card-pyramid structure. A dynamic programming algorithm, similar to CYK [34] parsing, called *card-pyramid* parsing is applied along with beam search to identify the most probable joint assignment of entities and their relations based on outputs of local classifiers. Other efforts to this end involve the use of a probabilistic graphical model [35, 36]. These methods work on the assumption that entity mention boundaries are known during test time.

Li and Ji [4] proposed the first model wherein entities (including entity mention bounds) and their relations are predicted using a single joint model. The predictive goal is to obtain a *structure* output for some input sentence given arbitrary (non-task specific) features and constraints, such that a structure contains both entity mentions and their relations. Structured perceptrons [37], as a learning framework, is used to estimate feature weights while beam-search is used to explore partial solutions to incrementally arrive at the most probable structure. Miwa and Sasaki [3] followed up by being the first to propose the idea of using a table representation which simplifies the task into a table-filling problem such that NER and relation labels are assigned to cells of the table; the goal is to predict a most-probable assignment to the table out of all possible assignments using beam search. The table is symmetric so only cells in the lower triangle are assigned a relation label, while cells along the diagonal are assigned entity labels. While the representation is in table form, the beam search is performed linearly, one cell-assignment per step. Many search orders are explored as part of the study and they discovered that assigning entity labels first resulted in slight gains that are statistically significant. They also deployed an enhanced margin-based version of the structured perceptron by Li and Ji [4] such that wrong assignments with small differences from groundtruth are penalized. Moreover, label dependencies are strictly enforced based on the previous label assignment, at the cost of additional computation time, so that illegal assignments are not possible.

Deep Neural Models. Miwa and Bansal [38] proposed the first deep neural network based model for end-to-end relation extraction. The model proposed consists of a sequential bi-LSTM layer for encoding contextual semantic information about the sentence as well as predicting NER tags. Each pair of identified entity are passed to a mention-level classifier based on a tree-structured LSTM where each pair of entities is represented by the shortest-path dependency subtree that connects them. Such a model trained in conjunction with entity pre-training and scheduled sampling was found to be highly effective. Katiyar and Cardie [39] proposed a deep neural model for jointly extracting entities and relations without the need to rely on dependency trees. They propose a multi-layered LSTM model that *jointly* assigns NER and relation labels at each timestep. For relation labeling, they use an attention layer as in pointer networks [40] to assign relatedness measures to tokens identified as entities in previous timesteps. A drawback of this approach is that, while NER tagging decisions are conditioned on the tag assigned to the previous token, pointer decisions are not conditioned similarly as in a true pointer network. Moreover, the model as proposed only permits entities to participate in at most one relation, which restricts its usefulness for real-world problems.

Zheng et al. [41] proposed a "hybrid" neural network based on an LSTM module for NER tagging and CNN for relation classification; the networks are trained separately despite such that intermediate representations of the NER module are passed as input to the relation classification module. A major drawback of such a setup is that the weights of the NER module are not affected during back-propagation when training the relation classification component. In this sense, the predictive capability of the NER component does not benefit from global knowledge about the relation component. In a separate but related work, Zheng et al. [42] proposes the idea of a novel tagging scheme for identifying entities as well as relations based on the BIES (Begin, Inside, End, Single) tags traditionally reserved for NER. In this proposed tagging scheme, there is an instance of the BIES tag for each relation type and for one of the two designations (subject or object). For example, a tag "B-Affects-Subject" indicates that a token is the beginning of an entity tag and the subject of an "Affects" relation type. The study suggests a layered encoder-decoder LSTM model typically observed for NER. This approach suffers from the same drawback as Katiyar and Cardie [39] wherein overlapping relations are not possible.

Pawar et al. [43] proposed a so called *All Word Pairs* (AWP) neural network model that, similar to Miwa and Sasaki [3], uses mention-level subnetworks to label pairs of words such that the end-to-end relation extraction reduces to a table filling problem. The model is advertised as a single joint network in the sense that network parameters are shared; however, the use of features that are specific to each pair of words (such as shortest dependency path) for mention-level prediction suggests a reliance on local classifiers as in past work. Since label predictions are independent of another, the authors used Markov Logic Networks (MLNs) as a global inference mechanism to correct label assignments. Li et al. [5] proposes a similar "joint" type model involving two sub-models; a neural network with CNNs at the character level and Bi-LSTM at the word level serves as the NER sub-model while a Bi-LSTM over the shortest dependency path serves as the relation classification sub-model. The sub-models share parameters but are trained separately in alternation.

Verga et al. [44] proposes a deep neural network model that simultaneously predicts relationships between all mention pairs in a document based on finding pairwise scores between entity mentions, using a bi-affine similarity function, and aggregates them to arrive at a single relation score for each entity pair. Katiyar and Cardie [39] and Bekoulis et al. [45] specifically use attention mechanisms for the RE component without the need for dependency parse features. Zheng et al. [46] operate by reducing the problem to a sequence-labeling task that relies on a novel tagging scheme. Zeng et al. [47] use an encoder-decoder network such that the input sentence is encoded as fixed-length vector and decoded to relation triples directly. Most recently, Bekoulis et al. [48] found that adversarial training (AT) is an effective regularization approach for $\mathcal{E}2\mathcal{E}$RE performance.

2.3 Notations and Neural Building Blocks

Herein, we adhere to the following mathematical notations for consistency. Matrices are denoted as uppercase letters, such as X, Y, and Z. We use standard subscript indexing notation to represent matrix indexing operations on rows and columns respectively. For example, $X_{i,j}$ corresponds to the element at the i^{th} row and j^{th} column of X. Furthermore, we use subscripted square brackets following a matrix term to denote a row-indexing operation; for example, $X_{[i]}$ corresponds to the vector at row i of X. The operation for selecting a range of rows is denoted similarly by making use of the colon; for example, $X_{[i:j]}$ corresponds to the sub-matrix consisting of rows i to j of X. Vectors are denoted as bolded lowercase letters such as $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, with standard subscript as an indexing notation. For example, $\mathbf{x}_i$ corresponds to the i^{th} element of $\mathbf{x}$. Moreover, vectors can be represented as a sequence using square brackets such that $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ supposing $\mathbf{x}$ is a vector of size n. The vector concatenation operation is denoted using the $\|$ symbol. Scalars and other types of variables are denoted as plain lowercase letters such as x, y, and z. Lastly, sets are represented as calligraphic uppercase letters such as $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$.

We next provide a generic definition of the canonical CNN and BiLSTM networks that are later used as building blocks in model construction. For ease of notation, we assume fixed sentence length n and word length $\hat{n}$; in practice, we set n and $\hat{n}$ to be the maximum sentence/word length and zero-pad shorter sentences/words.

CNN. Henceforth, the abstract function $f_{\text{CNN}}^{w,d_{\text{out}}}(\cdot) : \mathbb{R}^{n \times d_{\text{in}}} \mapsto \mathbb{R}^{d_{\text{out}}}$ is used to represent the CNN that convolves on a window of size w in a sentence with n words, mapping an $n \times d_{\text{in}}$ matrix to a vector representation of length d_{out}, where d_{in} is the word embedding size. This is an abstraction of the canonical CNN for NLP first proposed by Kim [7] and is defined as follows. First, we denote the convolution operation $\star$ as the sum of the element-wise products of two matrices. That is, for two matrices A and B of same dimensions, $A \star B = \sum_j \sum_k A_{j,k} \cdot B_{j,k}$. Suppose the input is a sequence of vector representations $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^{d_{\text{in}}}$; the output representation $\mathbf{g} \in \mathbb{R}^{d_{\text{out}}}$ is defined such that

$$\begin{aligned} \mathbf{g}_k = \max(\, & f_{\text{convolve}}(k, \mathbf{x}^1, \ldots, \mathbf{x}^w)\,,\, \ldots, \\ & f_{\text{convolve}}(k, \mathbf{x}^{n-w+1}, \ldots, \mathbf{x}^n)\,) \\ & \text{for } k = 1, \ldots, d_{\text{out}}, \end{aligned}$$

given a convolution function f_{convolve} that *convolves* over a contiguous window of size $w \leq n$, defined as

$$f_{\text{convolve}}(k, \mathbf{v}^1, \ldots, \mathbf{v}^w) = \text{ReLU}\left(W^k \star \begin{pmatrix} \mathbf{v}^1 \\ \vdots \\ \mathbf{v}^w \end{pmatrix} + b^k \right)$$

where $\mathbf{v}^1, \ldots, \mathbf{v}^w \in \mathbb{R}^{d_{\text{in}}}$ are input vectors, $W^k \in \mathbb{R}^{w \times d_{\text{in}}}$ and $b^k \in \mathbb{R}$ for $k = 1, \ldots, d_{\text{out}}$, are network parameters (corresponding to a set of d_{out} convolutional filters), $\text{ReLU(x)} = \max(0, x)$ is the linear rectifier activation function. Here, d_{out} is a hyperparameter that determines the number of convolutional filters and thus the size of the final feature vector. In the study, we denote the convolution as an abstract function $f_{\text{CNN}}^{w,d_{\text{out}}}(\cdot) : \mathbb{R}^{n \times d_{\text{in}}} \mapsto \mathbb{R}^{d_{\text{out}}}$ that convolves on a window of size w and maps an $n \times d_{\text{in}}$ matrix to a vector representation of length d_{out}.

BiLSTM. Likewise, we represent the BiLSTM network as an abstract function $f_{\text{BLSTM}}^{d_{\text{out}}}(\cdot) : \mathbb{R}^{n \times d_{\text{in}}} \mapsto \mathbb{R}^{n \times d_{\text{out}}}$ that maps a sequence of n input vectors (e.g., word embeddings) of d_{in} size (as an $n \times d_{\text{in}}$ matrix) to a corresponding sequence of n output context vectors of d_{out} size (as an $n \times d_{\text{out}}$ matrix). Let $\overrightarrow{\text{LSTM}}$ and $\overleftarrow{\text{LSTM}}$ represent an LSTM composition in the forward and backward direction. Suppose the input is a sequence of vector representations $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^{d_{\text{in}}}$; the output of a standard bidirectional LSTM network (BiLSTM) is a matrix $H \in \mathbb{R}^{n \times d_{\text{out}}} = (\mathbf{h}^1, \ldots, \mathbf{h}^n)^\top$ such that

$$\begin{aligned} \overrightarrow{\mathbf{h}}^i &= \overrightarrow{\text{LSTM}}(\mathbf{x}^i), \\ \overleftarrow{\mathbf{h}}^i &= \overleftarrow{\text{LSTM}}(\mathbf{x}^i), \\ \mathbf{h}^i &= \overrightarrow{\mathbf{h}}^i \parallel \overleftarrow{\mathbf{h}}^i, \qquad \text{for } i = 1, \ldots, n, \end{aligned}$$

where $\parallel$ is the vector concatenation operator and $\mathbf{h}^i \in \mathbb{R}^{d_{\text{out}}}$ represents the context centered at the i^{th} word. Here, d_{out} is a hyperparameter that determines the size of the the context embeddings. In the study, we denote the BiLSTM network as an abstract function $f^{d_{\text{out}}}_{\text{BLSTM}}(\cdot) : \mathbb{R}^{n \times d_{\text{in}}} \mapsto \mathbb{R}^{n \times d_{\text{out}}}$ that maps a sequence of n input vectors (e.g., word embeddings) of d_{in} size (as an $n \times d_{\text{in}}$ matrix) to a corresponding sequence of n output context vectors of d_{out} size (as an $n \times d_{\text{out}}$ matrix).

Chapter 3 Deep Learning for Extracting Protein-Protein Interactions

Precision medicine is an emerging disease treatment paradigm in which healthcare is customized to each individual patient. To support this effort, it is important to be able to extract useful translational information such as mentions of relationships between genes[1], mutations, and diseases. BioCreative (Critical Assessment of Information Extraction in Biology) [49] is an initiative with the aims of providing a standard evaluation framework for assessing text mining systems with respect to relevant problems in the biomedical domain. The related challenges are important as they provide an avenue for introducing new gold standard datasets to the research community that are hand-annotated by human domain experts. The precision medicine track of BioCreative VI, specifically, was organized to identify and study mutations and their effect on molecular interactions. Concretely, this track focuses on mining biomedical literature for protein-protein interactions (PPIs) that are affected by the presence of a genetic mutation. As an example, consider the following sentence: "We found that dominant-negative mutants of PML blocked AXIN-induced p53 activation, and that AXIN promotes PML SUMOylation, a modification necessary for PML functions." Here, we see that *AXIN* and *PML* are proteins that interact, as indicated by the assertion that *AXIN* promotes SUMOylation in *PML*; moreover, a mutation of *PML* is involved. Based on this observation, we can deduce that *AXIN* and *PML* are interesting pairs of proteins to study. We refer to this particular type of relation, where the participants of a PPI are also affected by a mutation, as a PPIm relation. This challenge is important as there has been a lack of tools that allows for the extraction of such interactions from biomedical literature despite its potential to support approaches in precision medicine.

The precision medicine track involves two distinct tasks: *document triage* and *relation extraction*. In the first task, participants are asked to build systems able to determine whether a PubMed citation is *relevant* or *not relevant* with respect to the relation extraction task; that is, whether or not it contains any PPIm relations to be extracted. In the second task, we are asked to build systems that take as input a PubMed citation and output any PPIm relations along with the Entrez Gene

---

[1] Given proteins are biochemical materials resulting from expression of corresponding genes, the terms gene and protein are used interchangeably in this study and the exact meaning is dependent on the context

IDs[2] of the participating genes. Thus for the second task, besides the input text, no additional information is provided making it a true end-to-end requirement where gene spotting, normalization, and interaction detection are all required.

In this study, we exclusively focus on the PPIm extraction task and propose a pipeline of three modular components: named entity recognition (NER), gene mention normalization (GN), and relation classification (RC). The input to the pipeline is a PubMed article and the output is a set of extracted PPIm pairs. The first component identifies spans of text corresponding to gene mentions. As an aside, we do not recognize pronouns referring to gene mentions, only the mentions themselves. The second component maps the gene mentions to their normalized Entrez Gene IDs. Lastly, the third component classifies all pairs of unique gene IDs found in the article as either positive or negative for the PPIm relation. The system we present here is an improved version of our original challenge entry [50] with three major changes. First, we use GNormPlus [51] to augment the original training corpus with additional gene annotations. For the NER component, this has the effect of reducing mixed signals stemming from the lack of annotations in the original training data. For the RC component, this provides many more meaningful negative examples such that the label imbalance more accurately reflects real-world situations. Second, during testing, we tag sequences of tokens that are missed by the NER component but appear in a gene lexicon (provided with the BioCreative II Gene Normalization training data [52]) to boost overall recall. Third, we consult PubTator [53] as a secondary reference (in addition to the gene database lookup; more later) for document-level gene annotations when mapping genes to their Entrez Gene IDs. We find that these changes drastically improve recall while retaining high precision.

The PPIm extraction task differs from a typical relation extraction task in three notable ways. First, a protein may interact with itself which implies that a protein can participate simultaneously as both the *subject* and the *object* of a PPIm relation. Second, directionality of a protein pair is immaterial which implies that (A, B) and (B, A) are equivalent for the sake of system evaluation. Here, the interaction type is also not important as in other PPI tasks so each relation can sufficiently be represented as a pair instead the usual (*subject*, *predicate*, *object*) triple. Lastly, it is possible for relations to be expressed across sentence bounds such that the subject and object mentions of a PPIm pair are in different sentences. Hence, we believe it is better to make relation classification decisions (i.e., extract protein-pairs) at the document level for this particular task. This is opposed to sentence-level relation

---

[2] `https://www.ncbi.nlm.nih.gov/gene/`

extraction where sentences are assumed to be mutually independent when extracting relations; and only pairs mentioned in the same sentence are considered as valid candidates for extraction. Document-level extraction has an additional advantage in that it takes into account sentence-level correlations such as order of sentences expressed.

In the rest of the manuscript, we discuss other approaches to this task and provide an overview of deep neural network architectures in Section 3.1. We present our main methods in Section 3.2 and discuss system performance and comparisons in Section 3.3.

3.1 Background and Related Work

In this section we cover prior efforts in biomedical relation extraction and the top performer of the PPIm extraction task we address in this manuscript.

3.1.1 Biomedical Relation Extraction

Many early works on relation extraction preprocess the input as a dependency parse tree [19, 20] and exploit features corresponding to the shortest dependency path between candidate entities; this general approach has also been successfully applied in the biomedical domain [21, 22, 23, 25], where they typically involve a graph kernel based Support Vector Machine (SVM) classifier [23, 25]. The concept of network centrality has also been applied [24] such that gene networks were created with respect to a specific disease; genes are then ranked according to network centrality metrics where highly ranked genes were considered more likely to be associated with the disease. Other studies, such as the effort by Frunza et al. [26], apply the more traditional *bag-of-words* approach focusing on syntactic and lexical features while exploring a wide variety of classification algorithms including decision trees, SVMs, and Naïve Bayes. More recently, innovations in relation extraction have centered around designing meaningful deep learning architectures. Liu et al. [27] proposed a dependency-based CNN architecture wherein the convolution is applied over words adjacent according to the shortest path connecting the entities in the dependency tree, rather than words adjacent with respect to the order expressed, to detect drug-drug interactions (DDIs). In Kavuluru et al. [28], ensembling of both character-level and word-level RNNs is further proposed for improved performance in DDI extraction. Raj et al. [29] proposed a deep learning architecture such that word representations are first processed by a bidirectional RNN layer followed by a standard CNN, with an optional attention mechanism towards the output layer. Luo et al. [30]

proposed convolving over not only the sentence, but rather over the five segments of a sentence: before the first entity mention, the first entity mention, in between the entity mentions, the second entity mention, and after the second entity mention. A single representation of the candidate relation and its context are then composed via simple concatenation of the CNN outputs. Recent studies have also explored joint modeling of both NER and relation extraction in an end-to-end fashion via deep neural networks [39, 38, 41].

3.1.2 Top Performing PPIm Extraction Entry

Chen et al. [54] produced the best micro-F1 scores during the BioCreative VI PPIm extraction challenge. They used the GNormPlus [51] tool as an "out-of-the-box" solution for recognizing and normalizing gene mentions. The main contribution lies in the relation classification aspect in which two different approaches are explored. The first is based on a rule-based system using the heuristic that if a protein-protein pair occur together in more than N sentences then it is considered positive for a PPIm relation. This works surprisingly well which is likely due to the observation that an article that has already been deemed relevant during document triage phase is likely topically-focused on a specific PPIm relation. It is reasonable to assume that two proteins mentioned together multiple times are more likely to be part of a relation than not. They found that $N = 2$ was optimal during validation. The second approach is based on traditional SVM with a graph kernel where the input is a dependency graph. Syntactic dependency graphs generated for each sentence are used as classifier features. In case a protein-pair is mentioned across two sentences, an artificial root node is generated connecting the roots of the two sentences to form a single larger graph to be used as input. They additionally experimented with introducing hand-picked mutation-context binary features in the form of 30 interaction terms including *interact*, *complex*, *bound*, *bind*, and *regulate*. From the 5-fold cross validation results on the training set, they found that SVM with these mutation features worked best at 27.5% F1. This is contrary to the test results, in which the rule-based approach was superior at 37.67% on the official test set. The authors note an end-to-end performance ceiling of 56% F1 when using GNormPlus for protein recognition and normalization. This aligns with our observation that improving the gene annotation aspect plays a key role in improving overall performance. The system we propose in this study uses more elaborate heuristics for the NER and gene normalization components and exploits recent advances in deep neural networks for natural language processing. Our current results improve upon Chen et al. [54] by 3 micro-F1 points in exact matching and by over 8 micro-F1 points in homolog-level matching strongly indicating that our end-to-end formulation is more suitable for this task.

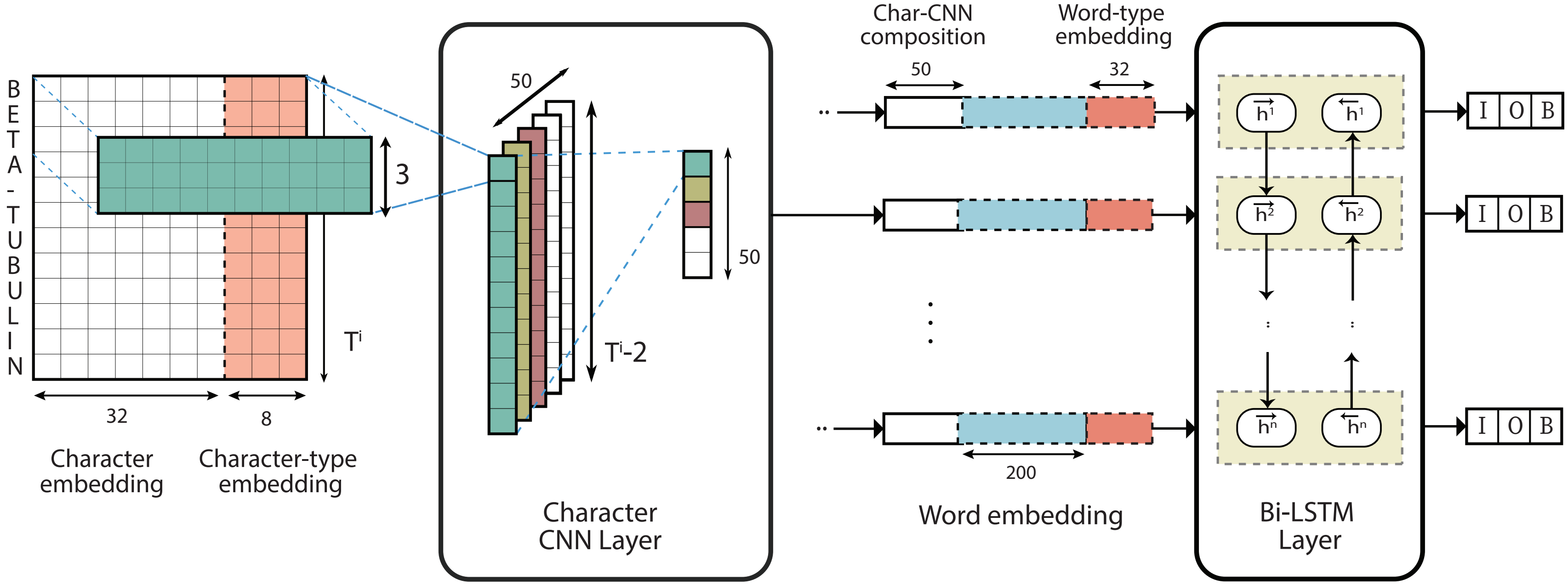
3.2 Materials and Methods

For the relation extraction subtask, we propose a pipeline system that consists of three components: supervised NER for gene mention detection, knowledge-based gene normalization, and supervised relation classification to predict each pair of genes found as either positive or negative for an interaction. It is possible to use an "out-of-the-box" solution such as GNormPlus that identifies both gene mentions and their corresponding gene identifier directly; however, we opted for a supervised approach that lets us leverage the generous gene annotations provided with the training corpus for this task. In the rest of this section, we first describe the dataset to be used in Section 3.2.1. We describe the NER system used to identify spans of text corresponding to a gene mention in Section 3.2.2. We then describe our knowledge-based method for gene normalization in Section 3.2.3 and relation classification model in Section 3.2.4.

3.2.1 PPIm Dataset

The PPIm dataset consists of 597 article title and abstracts each of which is annotated with gene mentions and interacting relevant protein pairs (at least one per citation) identified by their Entrez Gene IDs. In total, there are 752 pairs such that each article contains 1.26 relevant PPIm pairs *on average*. It is important to note that a gene is only annotated with mention-level offsets if it exists as part of a PPIm relation in the ground truth; hence, these gene annotations are incomplete for the sole purpose of training an NER model to identify gene mentions. The test has 632 articles each with at least one PPIm pair and a total of 868 PPIm pairs over the full test set; here we observe a similar distribution to the training set with an average of 1.37 pairs per article. Systems designed for this task are officially evaluated using standard metrics such as micro and macro F1/precision/recall; additionally, evaluations can be performed using exact or homologous gene matching. Further details of system evaluation are discussed in Section 3.3.

Figure 3.1: Deep neural network architecture of the NER model



3.2.2 Gene Mention Identification (NER)

The aim of the first component in the pipeline is to identify spans of text corresponding to gene mentions. To that end, we propose the use of a deep neural network system based on a CNN-LSTM hybrid model initially proposed by Chiu and Nichols [55] for NER. This sequence-to-sequence model composes word representations with CNNs by convolving over character n-grams. At the word level, contextual word representations are composed using a bi-directional LSTM layer. A separate fully-connected softmax output layer is present at the output of each LSTM unit such that an IOB[3] [56] label prediction can be made for each token. A visualization of the architecture can be observed in Figure 3.1.

Herein, we formulate the model from the bottom up. In this formulation, a word at position i for $i = 1, \dots, n$ is treated as a lowercased character sequence $c^i_1, \dots, c^i_{T^i}$ represented by their index into the character vocabulary $\mathcal{V}^{\text{char}}$. The corresponding character embedding matrix $E^{\text{char}} \in \mathbb{R}^{|\mathcal{V}^{\text{char}}| \times \alpha}$ embeds each character as a vector of length α (a hyper-parameter). Embedding matrices can be initialized to random or pretrained values; in either case, the word vectors are (further) modified via backward propagation. We use the same embedding setup to produce *character type* embedding vectors of length 8 indicating the type of character: lowercase, uppercase, punctuation, or other. Suppose the embedding matrix for *character type* is $E^{\text{ctype}} \in \mathbb{R}^{4 \times 8}$ and $z^i_1, \dots, z^i_{T^i}$ represents the sequence of enumerated character types for the word at position i. The word at position i can then be represented as a matrix composition B^i of its character embeddings, or concretely

$$B^i = \begin{pmatrix} E^{\text{char}}_{[c^i_1]} \parallel E^{\text{ctype}}_{[z^i_1]} \\ \vdots \\ E^{\text{char}}_{[c^i_{T^i}]} \parallel E^{\text{ctype}}_{[z^i_{T^i}]} \end{pmatrix}$$

where $E^{\text{char}}_{[j]}$, $E^{\text{ctype}}_{[j]}$ is the j^{th} row of E^{char}, E^{ctype} respectively and $\parallel$ is the vector concatenation operator. The central idea in CNNs is the so called *convolution* operation over the document matrix (or in this case, the “word” matrix) to produce a feature map representation using a *convolution filter* (CF). The convolution operation $*$ is formally defined as the sum of the element-wise products of two matrices. That is, for two matrices A and B of same dimensions, $A * B = \sum_j \sum_k A_{j,k} \cdot B_{j,k}$. We

---

[3] The Inside-Outside-Beginning (IOB) format is a tagging scheme commonly used in NER and sequence labeling tasks. The Inside and Beginning tags indicate that the tag is inside and at the beginning of a typed span respectively while Outside indicates that the tag is outside of a span. Typically, and in our model, the Beginning tag is only used when a tag is followed by a tag of the same type to indicate the start of a new span.

perform a convolution operation over B^i of window size three to obtain the feature map $\mathbf{v}^i = [v_1^i, \ldots, v_{T^i-2}^i]$ such that

$$v_j^i = \mathrm{ReLU}(W^{\mathrm{char}} * B^i_{[j:j+2]} + b^{\mathrm{char}})$$

where $B^i_{[j:j+2]}$ is a window of matrix B^i spanning from row j to row j+2, W^{char} and b^{char} are network parameters representing a CF, and the linear rectifier activation function $\mathrm{ReLU}(x) = \max(0, x)$. The goal is to learn multiple CFs that can collectively capture diverse representations of the same word. Here, specifically, we learn κ filters to obtain κ corresponding feature maps denoted as $\mathbf{v}^{i,1}, \ldots, \mathbf{v}^{i,\kappa}$. As a crucial step with CNNs, we select the most distinctive feature of each feature map using a max-over-time pooling operation [57]. Let $\mathbf{v}_k^{i,j}$ be the k^{th} value of $\mathbf{v}^{i,j}$, then the word representation at position i is $\mathbf{u}^i = [\hat{v}^{i,1}, \ldots, \hat{v}^{i,\kappa}]$ where $\hat{v}^{i,j} = \max(\mathbf{v}_1^{i,j}, \ldots, \mathbf{v}_{T^i-2}^{i,j})$. Conceptually, we can roughly equate this to composing a word representation using the traditional *bag-of-words* model, except here the features consist of character tri-grams. Because of the way max-pooling is applied, the order of tri-grams is immaterial.

Once a representation is composed for each word, we then use a bi-directional LSTM to model the word sequence. It is important that we also include actual word embeddings (in addition to those obtained through character embedding compositions) as well as *word type* embeddings as input. The latter embeddings serve a similar purpose to that of the *character types* and can correspond to one of the five following types: all lowercase, mixed-cased, capitalized first letter, all uppercase, or other. We now transition to a word-level perspective. Formally, the input to the network is a sequence of word indexes $w_1, \ldots, w_n$ into the word vocabulary $\mathcal{V}^{\mathrm{word}}$ and the corresponding embedding matrix is denoted as $E^{\mathrm{word}} \in \mathbb{R}^{|\mathcal{V}^{\mathrm{word}}| \times d}$. In addition, we denote $\bar{z}_1, \ldots, \bar{z}_n$ as a sequence of enumerated *word types* corresponding to the embedding matrix $E^{\mathrm{wtype}} \in \mathbb{R}^{5 \times \alpha}$. The bi-directional LSTM with a hidden/output unit size of π can then be composed as

$$\mathbf{h}^i = f^{\rho}_{\mathrm{BLSTM}}(\ \mathbf{u}^i \parallel E^{\mathrm{word}}_{[w_i]} \parallel E^{\mathrm{wtype}}_{[\bar{z}_i]}\ )_{[i]}$$

where $\mathbf{u}^i$ is character based embedding for w_i, $E^{\mathrm{word}}_{[j]}$ and $E^{\mathrm{wtype}}_{[j]}$ are j^{th} rows of E^{word} and E^{wtype} respectively, and $\mathrm{LSTM}^{\rightarrow}$ and $\mathrm{LSTM}^{\leftarrow}$ represent an LSTM unit composition in the forward and backward directions respectively. The concatenated output vector $\mathbf{h}^i \in \mathbb{R}^{2\pi}$ represents the entire context centered at the i^{th} word. The output at each timestep necessarily has its own softmax output layer in order for the network to be able to tag each word with an IOB label typically used for NER. The output at each position $i = 1, \ldots, n$ is

$$\mathbf{q}^i = W^{out}\mathbf{h}^i + b^{out}$$

where $W^{out} \in \mathbb{R}^{m \times 2\pi}$ and $b^{out} \in \mathbb{R}^m$ are network parameters and $m = 3$, the number of NER tags (*B-GENE*, *I-GENE*, and *O*). In order to get a categorical distribution, we apply the softmax formulation to $\mathbf{q}^i$ such that

$$\mathbf{p}^i_j = \frac{e^{\mathbf{q}^i_j}}{\sum_{l=1}^{m} e^{\mathbf{q}^i_l}}$$

where $\mathbf{p}^i$ is the vector of probability estimates serving as a categorical distribution over gene IOB tags for the word at position i. We optimize by computing the standard categorical cross-entropy loss at each output layer. Since each instance may be of a different sequence length, the final loss is computed as the *mean* over all n losses, one per word. The per-example loss ℓ is therefore computed as

$$\ell = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{y}^i_j \log(\mathbf{p}^i_j)$$

where $\mathbf{y}^i \in \mathbb{R}^m$ is the correct label for word i encoded as a one-hot vector. Next, we discuss the training procedure and model configuration.

Training and Model Configuration The NER model is trained on the training data and additionally on the GNormPlus corpus which includes re-annotations of the BioCreative II GM/GN corpus [52]. The core training data consists of 5668 sentence-level training examples while the GNormPlus corpus constitutes an additional 6389. We chose an embedding size of $\alpha = 32$ with $\kappa = 50$ filters for the character-based CNN composition. These hyperparameters were chosen based on the results of a hyperparameter search conducted by Chiu and Nichols [55] and further tweaked during initial experiments. At the word level, we used word embedding vectors of size $d = 200$ pre-trained on the PubMed corpus [58]. The forward and backward LSTM are each implemented with a hidden unit size of $\pi = 200$. The network was trained using SGD with an exponential decay rate of 0.95 for a maximum of 10,000 iterations. On each iteration, we trained the network using a mini-batch [59] of 20 random examples. We check-pointed every 100 iterations and saved only the checkpoint with the best F1 on the development set. We also deployed *early stopping* such that training is stopped if there are no improvements for ten checkpoints. We train ten such models (each with a different seed) as part of an ensemble where each model is trained on a smaller random subset of only 50% of the original training set. We observed that the ensemble was less prone to over-fitting (during initial experiments) when each model of the ensemble was only exposed to a smaller subset of the training data.

Augmented Gene Annotations An issue with the gene annotations in the training data is that they are not comprehensive. In fact, only genes participating in at least one relationship are annotated with mention-level offsets and gene IDs. This issue manifests in two distinct ways.

1. Mixed signals are introduced during learning (for the NER model) given it is possible for the same entity to appear as a target (annotated with *I-GENE*) for identification in one training example but not others (they are instead annotated with O) where it may not participate in an interaction. Due to the nature of a pipeline system, downstream bottlenecks can often occur as a result of low recall at the front-end of a pipeline. If we fail to identify a gene mention, for example, we will miss any relations it may participate in regardless of the competency of the relation classification component.

2. Data generated to train the relation classification component will not contain enough meaningful negative examples given gene mentions in the original training dataset are limited to those participating in interactions. From a manual observation of the data, we find that most examples generated are positive with many of the negative instances resulting from self-interactions.

From our original system submission [50], we found that models trained on only the provided annotations worked reasonably well despite the highlighted issues. As a strategy to overcome these issues and to improve end-to-end recall, we augment gene annotations provided in the training set using the PubTator tool [53] (which uses GNormPlus [51] as the backend for gene annotations). We simply run PubTator on the training corpus and insert genes it finds to corresponding spans of text in the training data that have consecutive O labels. The augmented corpus is instead used for training the supervised model (not only for NER, but relation classification as well). When doing this, we make sure to apply corrections such that the label sequence conforms to IOB rules.

Post-processing step Before proceeding to the gene normalization component, we perform a post-processing step to the output of the NER system in an attempt to maximize recall. Specifically, we use the gene lexicon provided with the BioCreative II Gene Normalization training data [52] as a knowledge source. The gene lexicon provides mappings of gene mentions to potential Entrez Gene IDs (keeping in mind that a gene mention may map to more than one unique ID). For a document input, we search for occurrences of gene mentions from the lexicon (note that we prioritize longer gene mentions over shorter ones) and add them as additional mentions to our supervised NER system's annotations barring those that overlap with our NER gene spans. In the gene normalization step (to be discussed next), we filter out gene mentions for which there are no plausible gene ID mappings. As such, the lower precision at the NER level due to this recall oriented post-processing step is reconcilable as we can weed out obviously bad gene mentions (i.e., gene mentions with no valid mapping) during gene normalization. Hence, precision can be compromised for the sake of improved recall for the NER component.

3.2.3 Entrez Gene ID Normalization (GN)

For the gene normalization component, we initially experimented with a naive look-up approach using the gene lexicon from BioCreative II normalization task [52] as well as mappings provided with the training corpus. This served as a reasonable baseline; however, it does not take context into consideration during the mapping process. A gene mention may be incorrectly mapped to one of its many homologs resulting in increased false positives. The final version of our gene normalization system is knowledge-based and more sophisticated in that it takes into consideration both the gene mention and the context. This system relies on the NCBI gene database [60] to identify the candidate gene IDs for a particular mention and further narrows it down to a "best guess" based on the document in which it occurred. We define two utility functions that serve as the basis for this system. Before we proceed, we recall that the full citation (title and abstract) represents a single input instance for our task. Hence the context for confirming the mapping is the Medline citation of the full article.

The first function, *gene_name_lookup*, takes as input a mention span and returns a list of candidate gene IDs sorted by relevance. This is achieved by querying the NCBI gene database via the E-utilities API[4]. This provides a ranked list of candidate genes for a given gene mention. The intuition here is that the top few in this list are either the correct gene or at least homologs of the correct gene. We now define the second function, *gene_pmid_lookup*, which takes as input a PubMed article ID (PMID) and returns a list of candidate gene IDs for the article. We achieve this by

---

[4] An example query for the gene span "Utp21": `https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=gene&term=Utp21&retmax=100&sort=relevance`

Algorithm 1 Gene Normalization

```
Input $a$: gene mention
Input $b$: document PMID
  $X \leftarrow$ gene_name_lookup($a$)
  $Y \leftarrow$ gene_pmid_lookup($b$)
  $Z \leftarrow$ pubtator_pmid_lookup($b$)

  for $x \in X$ do
    if $x \in Y$ then
      return $x$
    end if
  end for

  for $x \in X$ do
    if $x \in Z$ then
      return $x$
    end if
  end for

  return $NULL$
```

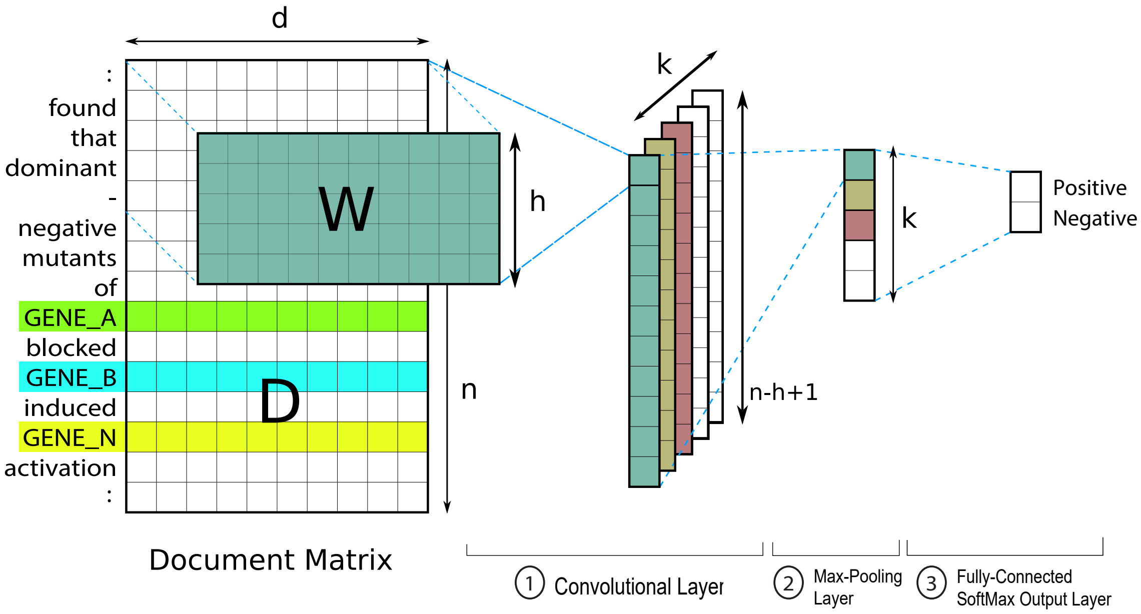
making another query to the NCBI gene database using the PMID of the current document as query input[5]. This allows us to narrow down the list of candidate gene IDs to ones that have already been identified as appearing in the document.

The final *gene_normalization* algorithm takes as input a gene mention and a PMID and returns either a gene ID or *NULL*. The latter indicates that no match can be found, in which case we simply ignore the span entirely for the remainder of the pipeline. From initial experiments, we found that relying only on the NCBI gene database to inform us of the possible genes for a document is too limiting and hurts recall considerably. This is because, while it is very precise, the database is not a comprehensive source of knowledge (at least for our purpose) and should not be relied upon as such. Hence, there is reason to believe that augmenting it with another high-precision system such as PubTator would improve overall recall. Let *pubtator_pmid_lookup* be a function that takes as input a PMID and returns a list of candidate genes for an article — not unlike *gene_pmid_lookup*. The difference is that *pubtator_pmid_lookup* returns the output of PubTator for the article without any information about word-level offsets; in other words, only a list of document-level gene IDs is returned. A natural union works well in our experiments, but we find a slight advantage in using *gene_pmid_lookup* as the primary source of knowledge

---

[5] An example query for the PMID 18725399: `https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=gene&term=18725399[PMID]`

Figure 3.2: Network architecture of the RC model (adapted from Tran and Kavuluru [1, Figure 1])



with *pubtator_pmid_lookup* serving as a secondary fallback. The final version of the procedure is defined in Algorithm 1. For example, suppose the document in question has PMID 18725399 and we wish to map the gene mention "Utp21" to a gene ID. The above algorithm will correctly return 851125 as the gene ID which can be verified via footnotes 4 and 5.

3.2.4 Relation Classification of Gene Pairs (RC)

To extract PPIm pairs, we propose using a deep neural network architecture based on CNNs for relation classification. The proposed model was originally introduced by Kim [7] for text classification and later adapted by us for narrative-based prediction of mental conditions [1]. An overview of the architecture modified to suit the relation classification task is presented in Figure 3.2. Since the architecture is identical (with exception of the output layer) to our prior work [1], we simply refer readers to the original study for the exact model formulation; the remainder of this section will instead focus on the training and configuration aspect of the model.

Training and Model Configuration When generating training examples for this model, we use the *augmented* training corpus as described in Section 3.2.2 with the additional gene mentions. For this task, each pair of candidate genes in an article constitutes a separate candidate interaction. Hence for each pair of candidate genes, we generate a distinct training instance by performing well-known entity-binding – we replace mentions of the pair with special tokens GENE_A and GENE_B (with their own embeddings) in the corresponding document text. We adapt this idea of entity-binding from prior efforts [28, 27] on classifying drug-drug interactions (DDIs), which obtained competitive results on a popular DDI dataset. For a gene pair (A, B), we also generate an additional instance for the reverse case (B, A) given directionality does not matter. Note that we run both cases of a candidate pair during testing and take the average output score for classification. We also generate examples for the exception case when the candidate pair involves the same gene, i.e. $A = B$, in which case GENE_S is used for entity binding of the single gene ID. We also replace mentions of other genes in the narrative with a special GENE_N token in either case. In total, we generated 2972 instances from the 597 articles in the training set. At test time, we only predict pairs as positive where the mean probability is above 50% for the instance generated from (A, B) and its reverse case (B, A). In case no pairs meet the threshold, we make a single positive prediction by choosing the pair with the highest probability (even if it is ≤ 50%).

We now describe the configuration of the RC model. As with the NER model, we used word embeddings of size 200 pre-trained on the PubMed corpus [58]. For the convolutional component, we used window sizes of 3, 4, and 5 with 200 convolutional filters. The model was trained for 30 epochs using RMSProp [61] (an SGD variant) using mini-batches [59] with a batch size of 8 and a learning rate of 0.001. Since each instance is a collection of sentences and the window size is at most 5, we pad four zero-vectors at the beginning and the end of the input text as well as between sentences. We additionally apply dropout at a rate of 50%. During training, we checkpoint model parameters at each epoch and only keep the checkpoint resulting in the highest F1 on the development set. We train 10 such models as part of an ensemble. Each model of the ensemble is trained and tuned on a random split of 80% to 20% and seeded with a different value for random parameter initialization. The neural network was configured based on insights from our prior work [1] with this particular architecture and further tuned during initial experiments.

3.3 Results and Discussion

Officially, systems submitted for this task are evaluated on micro-F1 with macro-F1 being a secondary metric introduced after the original challenge. There are two

Table 3.1: System performance on the official test set

	HomoloGene Matching	Method	Micro-P (%)	Micro-R (%)	Micro-F (%)	Macro-P (%)	Macro-R (%)	Macro-F (%)
1	✗	Task baseline	10.91	47.41	17.74	19.29	47.16	23.21
2		Tran and Kavuluru [50]	37.39	25.09	30.03	26.86	27.35	25.87
3		Chen et al. [54]	40.00	30.84	34.83	28.68	33.53	28.90
4		Our system	38.22	37.34	**37.78**	39.68	40.94	38.46
5	✓	Task baseline	14.68	51.97	22.90	21.36	51.57	26.02
6		Tran and Kavuluru [50]	46.53	31.09	37.27	32.87	34.15	31.94
7		Chen et al. [54]	43.18	33.41	37.67	30.87	35.86	31.09
8		Our system	46.67	45.69	**46.17**	48.53	49.94	47.03

matching criteria that are considered when evaluating: exact gene ID matching and HomoloGene Gene ID matching. In the latter case, genes of the same homology group are considered equivalent for the purpose of evaluation. This allows room for errors in the gene mapping aspect of the system and is therefore a less stringent measure compared to "exact matches." To identify homologous genes, the NCBI HomoloGene[6] database is used as a reference. In this context, the macro-F1 metric is based on computing the example-based F1 for each test article and averaging it over all test articles; this is different from the standard macro-F1 in a multi-class setting where it is the average of the F1 score over all classes.

The end-to-end performance of our system on the official test set is recorded in Table 3.1. Results of the top-performing participants of the original challenge are displayed in order of ascending micro-F1. Our original system submission [50] during the challenge placed second on exact matching (Table 3.1; row 2) and on HomoloGene ID matching (Table 3.1; row 6) at a micro-F1 of 30.03% and 37.27% respectively. As observed in Table 3.1, we were able to improve drastically on previous results by at least 7 points in micro-F1 for both exact and HomoloGene ID matching. The gains are almost entirely due to the improved recall of the new system although minor gains in precision were also observed. We also included the results of Chen et al. [54] for comparison as their system placed first on both matching criteria. Our improved system attains competitive test results for this dataset at 37.78% micro-F1 on exact matching and 46.17% micro-F1 on HomoloGene ID matching.

In Table 3.2, we study the iterative gains achieved by incrementally applying proposed changes to our original system [50]. In order to draw conclusions based on statistical significance, we apply the following experiment. First, we train a set of 30 models each with randomly initialized weights for both the NER and the relation classification component. Recall that both components make predictions based on

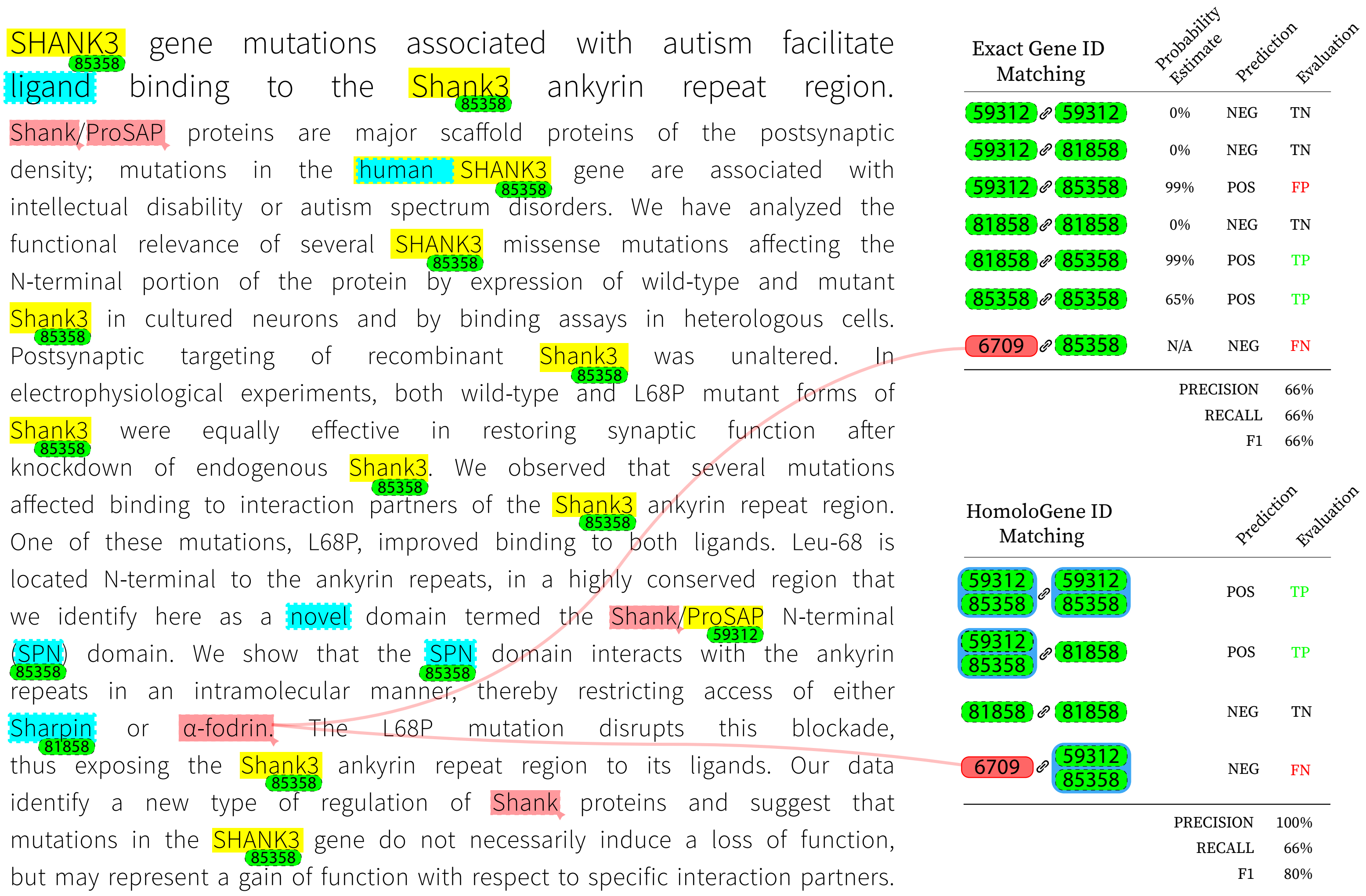
---

[6] https://www.ncbi.nlm.nih.gov/homologene

ten-model ensembles. We repeatedly evaluate the end-to-end system on the test set 30 times; each evaluation run involves a different ten-model ensemble for each component sampled from their respective pool of 30 trained models. We record the mean-F1 and 95% confidence intervals from these experiments in Table 3.2. Based on the results of this experiment, we can conclude that performance gains from the proposed changes are statistically significant (with exception of the retrained NER/RC component on HomoloGene ID matching). Next, we discuss these changes in detail.

We start by implementing changes to the NER and RC components such that they are trained on the augmented training corpus (recall that this corpus includes the original gene annotations as well as genes identified by GNormPlus). This results in fewer mixed signals for NER component while supplying the RC component with meaningful negative examples. From this, we see a notable improvement in micro-precision of at least 5 points on exact matching and 6 points on HomoloGene ID matching at a minor cost of recall in either case (rows 2 and 8 of Table 3.2); due to the nature of harmonic means and the fact that the performance already skews toward precision, improvements to micro-F1 are marginal. Next, we change the NER component by adding an NER post-processor that takes the output of the NER component and annotates unmatched gene names using the gene lexicon as a dictionary. From this we observe minor improvements (rows 3 and 9 of Table 3.2) to both precision and recall corresponding to an increase of at least 1 micro-F1 that is consistent for either matching criteria. A suspected bottleneck of our system is that it has an overly strict gene mapping criterion in that only genes that are annotated in the NCBI gene database for a particular PMID are allowed. The system is precise, but does not comprehensively cover all genes at the document level. Hence, we implemented a final change such that document-level PubTator (GNormPlus) annotations are used as a secondary recourse when considering the scope of genes to allow for a particular article. This final change is responsible for the most dramatic improvement (rows 4 and 10 of Table 3.2) to micro-recall at 12 points on exact matching and 14 points on HomoloGene ID matching. This comes with a cost to micro-precision at 5 points on exact matching and 6 points on HomoloGene ID matching. We arrive at relatively balanced precision and recall measures, an observation that is consistent on either matching criteria, resulting in an increase of at about 6 points on exact matching and 7 points on HomoloGene ID matching with respect to micro-F1.

Figure 3.3: Visualization of decisions made by the final system on article with PMID 23897824



SHANK3 gene mutations associated with autism facilitate ligand binding to the Shank3 ankyrin repeat region.

Shank/ProSAP proteins are major scaffold proteins of the postsynaptic density; mutations in the human SHANK3 gene are associated with intellectual disability or autism spectrum disorders. We have analyzed the functional relevance of several SHANK3 missense mutations affecting the N-terminal portion of the protein by expression of wild-type and mutant Shank3 in cultured neurons and by binding assays in heterologous cells. Postsynaptic targeting of recombinant Shank3 was unaltered. In electrophysiological experiments, both wild-type and L68P mutant forms of Shank3 were equally effective in restoring synaptic function after knockdown of endogenous Shank3. We observed that several mutations affected binding to interaction partners of the Shank3 ankyrin repeat region. One of these mutations, L68P, improved binding to both ligands. Leu-68 is located N-terminal to the ankyrin repeats, in a highly conserved region that we identify here as a novel domain termed the Shank/ProSAP N-terminal (SPN) domain. We show that the SPN domain interacts with the ankyrin repeats in an intramolecular manner, thereby restricting access of either Sharpin or α-fodrin. The L68P mutation disrupts this blockade, thus exposing the Shank3 ankyrin repeat region to its ligands. Our data identify a new type of regulation of Shank proteins and suggest that mutations in the SHANK3 gene do not necessarily induce a loss of function, but may represent a gain of function with respect to specific interaction partners.

Exact Gene ID Matching	Probability Estimate	Prediction	Evaluation
59312 🔗 59312	0%	NEG	TN
59312 🔗 81858	0%	NEG	TN
59312 🔗 85358	99%	POS	FP
81858 🔗 81858	0%	NEG	TN
81858 🔗 85358	99%	POS	TP
85358 🔗 85358	65%	POS	TP
6709 🔗 85358	N/A	NEG	FN

PRECISION 66%
RECALL 66%
F1 66%

HomoloGene ID Matching	Prediction	Evaluation
59312 85358 🔗 59312 85358	POS	TP
59312 85358 🔗 81858	POS	TP
81858 🔗 81858	NEG	TN
6709 🔗 59312 85358	NEG	FN

PRECISION 100%
RECALL 66%
F1 80%

Table 3.2: Iterative component-level analysis on the official test set

	HomoloGene ID Matching	Method	Micro-P (%)	Micro-R (%)	Micro-F (%)
1	✗	Our base system [50]	35.115 ± 0.488	25.380 ± 0.551	29.461 ± 0.540
2		+ Retrained NER/RC	40.848 ± 0.148	24.211 ± 0.094	30.403 ± 0.112
3		+ Improved NER	42.368 ± 0.126	25.210 ± 0.079	31.611 ± 0.090
4		+ Improved GN	37.425 ± 0.303	37.221 ± 0.205	**37.317** ± 0.194
5		Lexicon-based GN + Our NER/RC	12.149 ± 0.156	13.826 ± 0.132	12.925 ± 0.106
6		GNormPlus-based NER/GN + Our RC	37.069 ± 0.206	35.637 ± 0.176	36.333 ± 0.082
7	✓	Our base system [50]	44.335 ± 0.684	31.871 ± 0.708	37.077 ± 0.713
8		+ Retrained NER/RC	50.406 ± 0.161	29.991 ± 0.092	37.543 ± 0.113
9		+ Improved NER	52.393 ± 0.139	31.186 ± 0.081	39.099 ± 0.094
10		+ Improved GN	45.989 ± 0.365	45.863 ± 0.278	**45.927** ± 0.251
11		Lexicon-based GN + Our NER/RC	13.592 ± 0.183	15.594 ± 0.141	14.517 ± 0.121
12		GNormPlus-based NER/GN + Our RC	40.067 ± 0.178	38.632 ± 0.231	39.329 ± 0.095

We additionally include results based on other variants of our system for comparison. For example, we report performance for a variant in which the NER and RC component are fixed while replacing the GN component with a method based on gene lexicon mapping and a fuzzy string matching that allowed genes to be mapped to gene IDs within a 90% similarity threshold. This corresponds to rows 5 and 11 of Table 3.2 in which we observe very poor performance. For HomoloGene ID matching, the result is worse than the baseline reported in Table 3.1. This is expected as article context is not used to infer the correct gene ID from many possible gene IDs that are homologous in nature. On the other hand, using GNormPlus with the retrained RC component results in surprisingly high performance. This is contrary to our initial experiments on a held-out validation set wherein GNormPlus performs much worse at a micro-F1 of 26.75% – granted this was prior to system improvements as described in this study. This could be an indicator that GNormPlus is better at annotating genes on the test set than the training set. Nevertheless, relying on GNormPlus as the core NER and GN component would result in 36.33% and 39.33% micro-F1 scores on exact and homologous matching respectively(rows 6 and 12 of Table 3.2); while these scores are high, this restricts any further improvement to strictly the RC component and the pipeline wide improvements achieved by our system are still superior (rows 4 and 10 of Table 3.2).

To gain further insight on the inner workings of the final system, we provide a visualization of intermediate decisions made on a concrete example in Figure 3.3. The target article, identified by PMID 23897824, was manually chosen from the set of test examples based on its potential for discussion as well as practical considerations (such as length). Highlighted in yellow are spans of text initially identified by the NER system; further corrections to these annotations by consulting the gene lexicon are highlighted in blue. Gene ID annotations are tagged (in green) for each named entity span for which the gene normalization component finds a suitable match. The color red is reserved for spans and genes which were missed entirely by the system. We also include an example-based evaluation on both matching criteria for the final prediction. For HomoloGene ID matching, we group genes that are homologous accordingly.

One clear observation to be made is that most occurrences of the gene *Shank3* are captured by the supervised NER system. Since *Shank3* and its variants do not occur in the training set, this example demonstrates the ability of the system to generalize to unseen examples. Occurrences of the same gene without the numeric suffix are not captured however, which can be an indication that the character-level composition plays an influential role and that there is bias for word tokens that are a mix of alphabetic and numeric characters. We can also observe that the NER component was unable to detect the gene *α-fodrin*, more commonly known as *SPTAN1*. This is due to the system's lack of support for non-ASCII characters; here, we believe a simple preprocessing step to convert non-ASCII characters to a more processable form prior to training and testing will alleviate such issues. The final evaluation of this example shows that missing such genes can be detrimental to overall recall. The post-NER correction step introduces its share of false positives including *ligand* and *novel*; nonetheless, it is responsible for detecting the only mention of the gene *Sharpin*, which is a participant of a PPIm relation according to the groundtruth. The result is a net-gain as the false positives introduced are not normalized by the gene normalization component at this stage and are therefore ignored for the rest of the pipeline. Another observation is that *Shank/ProSAP* individually refer to protein names but in this context may refer to a group of proteins; the first instance of this mention is ignored while the NER system detects only *ProSAP* in the second mention. In this case, *ProSAP* appears to be a source of error as it is ultimately mapped to gene ID 59312, which is *Shank3* but of the variety that occurs in the Norwegian rat. This is in contrast to other instances of *Shank3* which are correctly identified as of the human variety (gene ID 85358). Despite genes 59312 and 85358 being homologous, incorrectly identifying the precise gene ID predictably results in a false positive when evaluating on exact matches. This issue disappears when matching on HomoloGene IDs, as shown in the right panel of Figure 3.3. To bridge the gap between exact and homologous gene ID matching performance, one option to reduce false positives is by consolidating the gene ID mappings for subsets of unique gene IDs that are homologous; for example, the use of a voting mechanism for deciding the correct variant for all members of the subset. However, it is necessary to consider the trade-off since such a system would not perform well on articles without narrow focus on any particular species of animal.

3.4 Conclusion

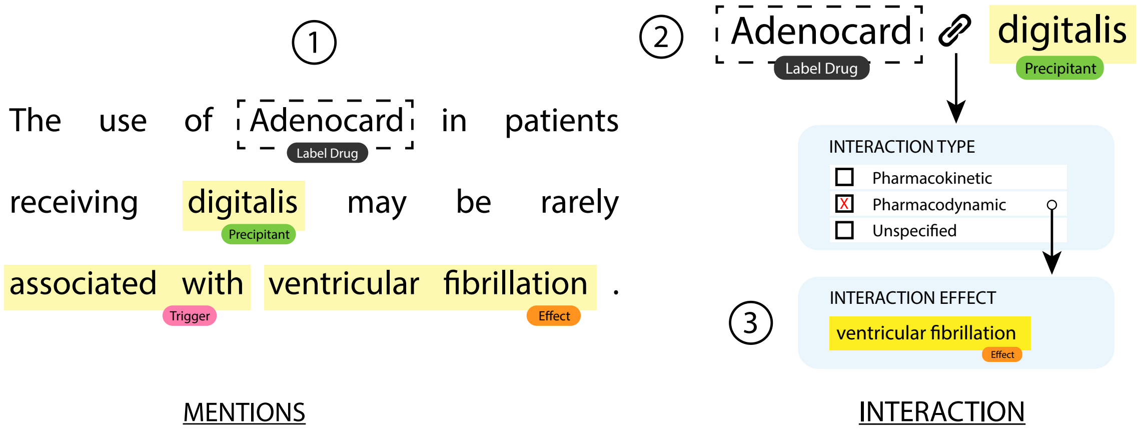
In this chapter, we proposed an end-to-end deep learning system that consists of named entity recognition, gene normalization, and relation classification for the BioCreative VI Precision Medicine track's task on relation extraction. We proposed changes to our original system entry for the challenge and analyzed the incremental performance gains of these changes. Furthermore, we demonstrated that the proposed system performs competitively for this task by significantly improving upon top results achieved in the original challenge. We believe this is an important progression in supporting efforts in precision medicine. A drawback of the system is the lack of built-in mechanisms for interpretability of decisions, which can be rectified by adding an attention layer to highlight contextual words or phrases that are central to this new problem domain. On the other hand, the lack of comprehensive gene annotations also poses a non-trivial challenge when attempting to build an end-to-end system for this task. The system as proposed relies heavily on numerous external tools and knowledge bases to circumvent the lack of comprehensive gene annotations. As human-expert annotations are expensive and time consuming, this aspect may continue to surface in future datasets of a similar nature. The next chapters will focus on dealing with this aspect in a more direct fashion while realizing a true end-to-end deep neural network that is able to model all components jointly.

Chapter 4 Graph Convolutions for Extracting Drug Interaction Information

Preventable adverse events (AE) are negative consequences of medical care resulting in injury or illness in a way that is generally considered avoidable. According to a report [62] by the Department of Human and Health Services, based on an analysis of hospital visits by over a million Medicare beneficiaries, about *one in seven* hospital visits were associated with an AE with 44% being considered clearly or likely preventable. Overall, AEs were responsible for an estimated US $324 million in Medicare spending for the studied month of October 2008. Preventable AEs thus introduce a growing concern in the modern healthcare system as they represent a significant fraction of hospital admissions and play a significant role in increased health care costs. Alarmingly, preventable AEs have been cited as the eighth leading cause of death in the U.S., with an estimated fatality rate of between 44,000 and 98,000 each year [63]. As drug-drug interactions (DDIs) may lead to to variety of preventable AEs, being able to extract DDIs from prescription drug labels is an important effort toward effective dissemination of drug safety information. This includes extracting information such as adverse drug reactions and drug-drug interactions as indicated by drug labels. The U.S. Food and Drug Administration (FDA), for example, has recently begun to transform Structured Product Labeling (SPL) documents into a computer-readable format, encoded in national standard terminologies, that will be made available to the the medical community and the public [64]. The initiative to develop a database of structured drug safety information that can be indexed, searched, and sorted is an important milestone toward a *fully-automated* health information exchange system.

To aid in this effort, we propose a supervised deep learning model able to tackle the problem of drug-drug interaction extraction in an *end-to-end* fashion. While most prior efforts assume all drug entities are known ahead of time (more in Section 4.1), and the drug-drug interaction extraction task reduces to a simpler *binary relation classification* task of known drug pairs, we propose a system able to identify drug mentions in addition to their interactions. Concretely, the system takes as input the textual content of the label (indicating dosage and drug safety precautions) of a target drug and, as output, identifies mentions of other drugs that interact with the target drug. Thus only one of the two interacting drugs is known beforehand (i.e., the "label drug"), while the other (i.e., the "precipitating drug", or simply precipitant) is an unknown that our model is expected to extract. Along with identifying precipitants,

Figure 4.1: An example illustrating the end-to-end DDI extraction task. We first (1) identify mentions including precipitants; for each precipitant, we (2) determine the type of interaction and, based on interaction type, (3) determine the interaction outcome. In the case of PD interactions, the outcome corresponds to one of the previously identified *effect* spans.



we also determine the type of interaction associated with each precipitant; that is, whether the interaction is designated as being pharmacodynamic (PD) or pharmacokinetic (PK). In pharmacology, PD interactions are associated with a consequence on the organism while PK interactions are associated with changes in how one or both of the interacting drugs is absorbed, transported, distributed, metabolized, and excreted when used jointly. Beyond identifying the interaction type, it is also important to identify the outcome or consequence of an interaction. As defined, PK consequence can be captured using a small fixed vocabulary, while identifying PD effects is a much more contrived process. The latter involves additionally identifying spans of text correspond to a mention of a PD effect and linking each identified PD precipitants to one or more PD effects. We provide a more formal description of the task in Section 4.2.1. Figure 4.1 features a simple example of a PD interaction that is extracted from the drug label for **Adenocard**, where the precipitant is *digitalis* and the effect is “ventricular fibrillation.”

To address this end-to-end variant of DDI extraction, we propose a multi-task joint-learning architecture wherein various intermediate hidden representations, including sequence-based and graph-based contextual representations based on bidirectional Long Short-Term Memory (BiLSTM) networks and graph convolution (GC) networks respectively, are composed and are then combined in clever ways to produce predictions for each subtask. GCs over dependency parse trees are useful for

Table 4.1: Characteristics of various datasets

	*DDI2013	*NLM180	TR22	Test Set 1	Test Set 2
Number of Drug Labels	715	180	22	57	66
Total number of sentences	6489	5757	603	8195	4256
Number of sentences per Drug Label (Average)	9	32	27	144	64
Number of words per sentence (Average)	21	23	24	22	23
Proportion of sentences with *annotations*	70%	27%	51%	23%	23%
Number of mentions per *annotated* sentence (Average)	2.3	4.0	3.8	3.7	3.6
Proportion of mentions that are Precipitant	100%	57%	53%	56%	55%
Proportion of mentions that are Trigger	-	20%	28%	30%	33%
Proportion of mentions that are Effect	-	23%	19%	14%	12%
Proportion of interactions that are Pharmacodynamic	14%	47%	49%	33%	28%
Proportion of interactions that are Pharmacokinetic	9%	25%	21%	28%	47%
Proportion of interactions that are Unspecified	77%	28%	30%	39%	25%

* Statistics for NLM180 and DDI2013 were computed on mapped examples (based on our own annotation mapping scheme) and not based on the original data.

capturing long-distance syntactic dependencies. We innovate on conventional GCs with a sigmoid gating mechanism derived via additive attention, referred to as Graph Convolution with Attention-Gating (GCA), which determines whether or not (and to what extent) information propagates between source and target nodes corresponding to edges in the dependency tree. The attention component controls information flow by producing a sigmoid gate (corresponding to a value in $[0, 1]$) for each edge based on an attention-like mechanism that measures relevance between node pairs. Intuitively, some dependency edges are more relevant than others; for example, *negations* or *adjectives* linked to important nouns via dependency edges may have a large influence on the overall meaning of a sentence while *articles*, such as “the”, “a”, and “an”, have little or no influence comparatively. A standard GC would compose all source nodes with equal weighting, while the GCA would be more selective by possibly assigning a higher sigmoid value to *negations*/*adjectives* and a lower sigmoid value to *articles*.

We train and evaluate our model on the Text Analysis Conference (TAC) 2018 dataset for drug-drug interaction extraction from drug labels [64]. The training data contains 22 drug labels, referred to as TR22, with gold standard annotations. As training data is scarce, we additionally propose a transfer learning step whereby the model is first trained on external data for extracting DDIs including the NLM-DDI CD corpus[1] and SemEval-2013 Task 9 dataset [65]; we refer to these as NLM180 and DDI2013 respectively. Two official test sets of 57 and 66 drug labels, referred

---

[1] https://lhce-brat.nlm.nih.gov/NLMDDICorpus.htm

to as Test Set 1 and 2 respectively, with gold standard annotations are used strictly for evaluation. Table 4.1 contains more information about these datasets and their characteristics. In this study, we show that the GCA improves over the standard GC and that our GCA based model with transfer learning by pretraining on external data improves over our the best model [66] from a prior study[2], that is based solely on BiLSTMs, by 4 absolute F1 points in overall performance. Furthermore, we show that our GCA based model complements our prior BiLSTM model; that is, by combining the two via ensembling, we improve over the prior best by 6 absolute F1 points in overall performance. Among comparable methods, our GCA based method exhibits state-of-the-art performance on all metrics after controlling for available training data.

4.1 Background and Related Work

Prior studies on DDI extraction have focused primarily on binary relation extraction where drug entities are known during test time and the learning objective is reduced to a simpler *relation classification* (RC) task. In RC, pairs of known drug entities occurring in the same sentence are assigned a label, from a fixed set of labels, indicating relation type (including the *none* or *null* relation). Typically, no preliminary drug entity recognition or additional consequence prediction step is required. In this section, we cover prior relation extraction methods for DDI as well as participants of the initial TAC DDI challenge.

4.1.1 Relation Extraction for DDI

State-of-the-art methods for DDI extraction typically involve some variant of convolutional neural networks (CNNs) or recurrent neural networks (RNNs), or a hybrid of the two. Many studies utilize the dependency parse structure of an input sentence to capture long-distance dependencies, which has previously been shown to improve performance in general relation extraction tasks [31] and those in the biomedical domain [67, 27]. Liu et al. [68] first proposed the use of standard CNNs for DDI extraction. Their approach involved convolving over an input sentence with drug entities bound to generic tokens in conjunction with so called *position vectors*. Position vectors are used to indicate the offset between a word and each drug of the pair and provide additional spatial features. Improvements were attained, in a follow-up study, by instead convolving over the shortest dependency path between the candi-

---

[2] Tran et al. [66] was published as part of the non-refereed Text Analysis Conference (TAC); this study is an extension of our original report.

date drug pair [27]. Zhao et al. [69] introduced an enhanced version of the CNN based method by deploying word embeddings that were pretrained on syntactic parses, part-of-speech embeddings, and traditional handcrafted features. Suárez-Paniagua et al. [70] instead focused on fine-tuning various hyperparameter settings including word and position vector dimensions and convolution filter sizes for improved performance. Kavuluru et al. [28] introduced the first neural architecture for DDI extraction based on hierarchical RNNs, wherein hidden intermediate representations are composed in a sequential fashion with cyclic connections, with character and word-level input. Sahu and Anand [71] experimented with various ways of composing the output of a bidirectional LSTM network including max-pooling and attention pooling. Lim et al. [72] proposed a *recursive* neural network architecture using recurrent units called TreeLSTMs to produce meaningful intermediate representations that are composed based on the structure of the dependency parse tree of a sentence. Asada et al. [73] demonstrated that combining representations of a CNN over the input text and graph convolutions over the molecular structure of the target drug pair (as informed by an external drug database) can result in improved DDI extraction performance. More recently, Sun et al. [74] proposed a hybrid RNN/CNN method by convolving over the contextual representations produced by a preceding recurrent neural network.

4.1.2 TAC 2018 DDI Track

TAC is a series of workshops organized by NIST aimed at encouraging research in natural language processing (NLP) by providing large test collections along with a standard evaluation procedure. The "DDI Extraction from Drug Labels" track [64] is established with the goal of transforming the contents of drug labels into a machine-processable format with linkage to standard terminologies. Tang et al. [75] placed first in the challenge using an encoder/decoder architecture to jointly identify precipitants and their interaction types and a rule-based system to determine interaction outcome. In addition to the provided training data, they downloaded and manually annotated a collection of 1148 sentences to be used as external training data. Tran et al. [66] placed second in the challenge using a BiLSTM for joint entity recognition and interaction type prediction, followed by a CNN with two separate dense output layers (one of PK and one for PD) for outcome prediction. Dandala et al. [76] placed third in the challenge using a BiLSTM (with CRFs) with part-of-speech and dependency features as input for entity recognition. Next, an Attention-LSTM model was used to detect relations between recognized entities. The embeddings were pretrained on a corpus of FDA-released drug labels and used to initialized the model. NLM180 was used for training with TR22 serving as the development set. Other participants proposed systems involving similar approaches including BiLSTMs and CNNs as well as traditional linear and rule-based methods.

4.2 Materials and Methods

We begin by formally describing the end-to-end task in Section 4.2.1. Next, we describe our approach to framing and modeling the problem (Section 4.2.2), the proposed network architecture (Section 4.2.3), the data used for transfer learning (Section 4.2.4), and our model-ensembling approach (Section 4.2.5). Finally, in Section 4.2.6, we describe the method for model evaluation.

4.2.1 Task Description

Herein, we describe the end-to-end task of automatically detecting drugs and their interactions, including the outcome of identified interactions, as conveyed in drug labels. We first define drug label as a collection of sections (e.g., `DOSAGE & ADMINISTRATION`, `CONTRAINDICATIONS`, and `WARNINGS`) where each section contains one or more sentences. The overall task, in essence, involve fundamental language processing techniques including named entity recognition (NER) and relation extraction (RE). The first subtask of NER is focused on identifying **mentions** in the text corresponding to precipitants, interaction triggers, and interaction effects. Precipitating drugs (or simply precipitants) are defined as substances, drugs, or a drug class involved in an interaction. The second subtask of RE is focused on identifying sentence-level interactions; specifically, the goal is to identify the interacting precipitant, the type of the interaction, and outcome of the interaction. The interaction outcome depends on the interaction type as follows. Pharmacodynamic (PD) interactions are associated with a specified *effect* corresponding to a span within the text that describes the outcome of the interaction. Figure 4.1 features a simple example of a PD interaction that is extracted from the drug label for **Adenocard**, where the precipitant is *digitalis* and the effect is “ventricular fibrillation.” Naturally, it is possible for a precipitant to be involved in multiple PD interactions. Pharmacokinetic (PK) interactions, on the other hand, are associated with a label from a fixed vocabulary of National Cancer Institute (NCI) Thesaurus codes indicating various levels of increase/decrease in functional measurements. For example, consider the sentence: “There is evidence that treatment with `phenytoin` leads to decrease intestinal absorption of `furosemide`, and consequently to lower peak serum `furosemide` concentrations.” Here, *phenytoin* is

Table 4.2: Example of the sequence labeling scheme for the sentence in Figure 4.1, where `LABELDRUG` is substitute for Adenocard.

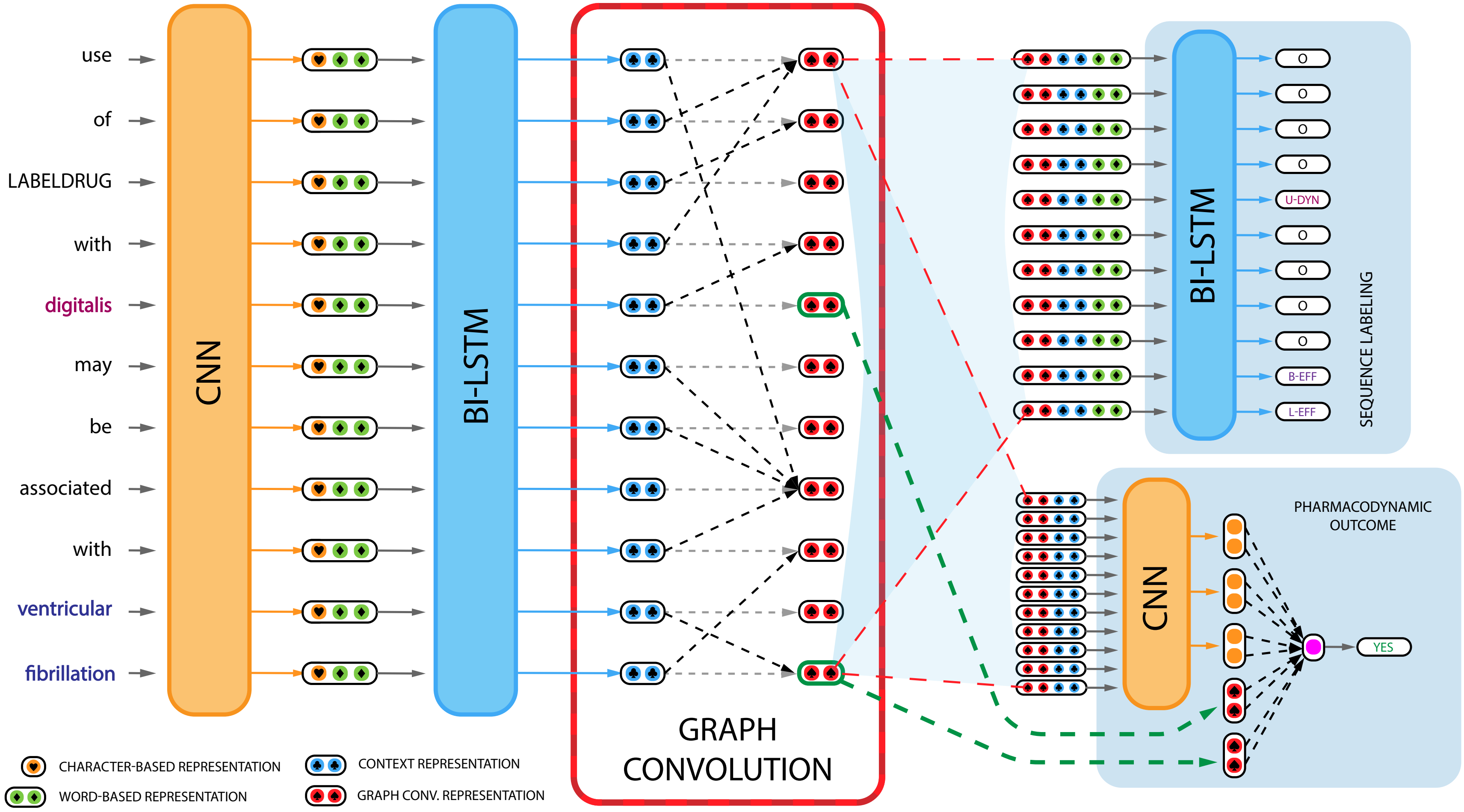
`O`	`O`	`O`	`O`	`O`	`O`	`O`	`U-DYN`
The	use	of	LABELDRUG	in	patients	receiving	digitalis
`O`	`O`	`O`	`B-TRI`	`L-TRI`	`B-EFF`	`L-EFF`	`O`
may	be	rarely	associated	with	ventricular	fibrillation	.

involved in a PK interaction with the label drug, *furosemide*, and the type of PK interaction is indicated by the NCI Thesaurus code C54615 which describes a decrease in the maximum serum concentration ($C_{\max}$) of the label drug. Lastly, *unspecified* (UN) interactions are interactions with an outcome that is not explicitly stated in the text and is typically indicated through cautionary remarks.

4.2.2 Joint Modeling Approach

Since only precipitants are annotated in the ground truth, we model the task of precipitant recognition and interaction type prediction jointly. We accomplish this by reducing the problem to a sequence tagging problem via a novel NER tagging scheme. That is, for each precipitant drug, we additionally encode the associated interaction type. Hence, there are three possible precipitant tags: **DYN**, **KIN**, and **UN** for precipitants with pharmaco**dyn**amic, pharmaco**kin**etic, and **un**specified interactions respectively. Two more tags, **TRI** and **EFF**, are added to further identify mentions of *triggers* and *effects* concurrently. To properly identify boundaries, we employ the BILOU encoding scheme [77]. In the BILOU scheme, *B*, *I*, and *L* tags are used to indicate the beginning, inside, and last token of a multi-token entity respectively. The *U* tag is used for unit-length entities while the *O* tag indicates that the token is outside of an entity span. As a preprocessing step, we identify the label drug in the sentence, if it is mentioned, and bind it to a generic entity token (e.g., "LABELDRUG"). We also account for indirect mentions of the label drug, such as the generic version of a brand-name drug, or cases where the label drug is referred to by its drug class. To that end, we built a lexicon of drug names mapped to alias using NLM's Medical Subject Heading (MeSH) tree as a reference. Table 4.2 shows how the tagging scheme is applied to a simple example.

Figure 4.2: Overview of the neural network architecture for a simplified example from the drug label Adenocard. Here, the ground truth indicates that *digitalis* is a pharmacodynamic precipitant associated with the effect “ventricular fibrillation.” The PK predictive component is omitted given there are no precipitants involved in a PK interaction.



Once we have identified the precipitant (as well as triggers/effects) and the interaction type for each precipitant, we subsequently predict the outcome or consequence of the interaction (if any). To that end, we consider all entity spans annotated with **KIN** tags and assign them a label from a static vocabulary of 20 NCI concept codes corresponding to PK consequence (i.e., multi-class classification). Likewise, we consider all entity spans annotated with **DYN** tags and link them to mention spans annotated with **EFF** tags; we accomplish this via binary classification of all pairwise combinations. For entity spans with **UN** tags, no outcome prediction is needed.

4.2.3 Neural Network Architecture and Training Details

We begin by describing how the three types of intermediate representations are composed. The construction of word, context, and graph-based representations are described in the remainder of this section. We note that this section references neural building blocks originally described in Section 2.3. Next, we describe the predictive components of the network that share and utilize the intermediate representations. Afterwards, we describe the sequence-labeling component of the network used to extract drugs and their interactions. Then, we describe the component for predicting interaction outcome. An overview of the architecture is shown in Figure 4.2. Lastly, we describe the model configuration and training process.

Word-level Representation

Suppose the input is a sentence of length n represented by a sequence of word indices $w_1, \ldots, w_n$ into the vocabulary $\mathcal{V}^{\text{Word}}$. Each word is mapped to a word embedding vector via embedding matrices $E^{\text{Word}} \in \mathbb{R}^{|\mathcal{V}^{\text{Word}}| \times \delta}$ such that δ is a hyperparameter that determines the size of word embeddings. In addition to word embeddings, we employ character-CNN based representations as commonly observed in recent neural NER models [55]. Character-based models capture morphological features and help generalize to out-of-vocabulary words. For the proposed model, such representations are composed by convolving over character embeddings of size π using a window of size 3, producing η feature maps; the feature maps are then max-pooled to produce η-length feature representations. Correspondingly, we denote $E^{\text{Char}} \in \mathbb{R}^{|\mathcal{V}^{\text{Char}}| \times \pi}$ as the embedding matrix given the character vocabulary $\mathcal{V}^{\text{Char}}$; the character-level embedding matrix $C^i \in \mathbb{R}^{\hat{n} \times \pi}$ for the word at position i is

$$C^i = \begin{pmatrix} E^{\text{Char}}{}_{[c_{i,1}]} \\ \vdots \\ E^{\text{Char}}{}_{[c_{i,\hat{n}}]} \end{pmatrix}$$

where $c_{i,j}$ for $1 \leq i \leq n, 1 \leq j \leq \hat{n}$, represents the j^{th} character index of the i^{th} word. The word-level representation $R^{\text{word}} \in \mathbb{R}^{n \times (\delta + \eta)}$ is a concatenation of character-based word embeddings and pretrained word embeddings along the feature dimension; formally,

$$R^{\text{Word}} = \begin{pmatrix} E^{\text{Word}}{}_{[w_1]} \parallel f^{3,\eta}_{\text{CNN}}(C^1) \\ \vdots \\ E^{\text{Word}}{}_{[w_n]} \parallel f^{3,\eta}_{\text{CNN}}(C^n) \end{pmatrix}.$$

Context-based Representation

We compose context-based representation by simply processing the word-level representation with a BiLSTM layer as is common practice; concretely, $R^{\text{Context}} = f^{\rho}_{\text{BLSTM}}(R^{\text{Word}})$ where ρ is a hyperparameter that determines the size of the context embeddings.

Graph-based Representation

In addition to the sequential nature of LSTMs, we propose an alternative and complementary graph-based approach for representing context using graph convolution (GC) networks. Typically composed on dependency parse trees, graph-based representations are useful for relation extraction as they capture long-distance relationships among words of a sentence as informed by the sentence's syntactic dependency structure. While graph convolutions are typically applied repeatedly, our initial cross-validation results indicate that single-layered GCs are sufficient and deep GCs typically resulted in performance degradation; moreover, Zhang et al. [31] report good performance with similarly shallow GC layers. Hence the following formulation describes a *single-layered* GC network, with an additional attention-based sigmoid gating mechanism, which we holistically refer to as a Graph Convolution with Attention-Gating (GCA) network. As mentioned in the beginning of Chapter 4, the GCA improves on conventional GCs with a sigmoid-gating mechanism derived via an alignment score function associated with *additive* attention [13]. The sigmoid "gate" determines whether or not (and to what extent) information is propagated based on a learned alignment function that conceives a "relevance" score between a *source* and *target* node (more later).

As a pre-processing step, we use a dependency parsing tool to generate the projective dependency tree for the input sentence. We represent the dependency tree as an $n \times n$ adjacency matrix A where $A_{i,j} = A_{j,i} = 1$ if there is a dependency relation between words at positions i and j. This matrix controls the flow of information between pairs of words corresponding to connected nodes in the dependency tree (ignoring dependency *type*); however, it is also important for the *existing* information of each node to carry over on each application of the GC. Hence, as with prior work [31], we use the modified version $\tilde{A} = A + I$ where I is the identity matrix to allow for self-loops in the GC network. The graph-based representation $R^{\text{Graph}} \in \mathbb{R}^{n \times \beta}$ is composed such that

$$R^{\text{Graph}}{}_{[i]} = \tanh\left(\sum_{j=1}^{n} \tilde{A}_{i,j} W^{\text{Graph}} R^{\text{Context}}{}_{[j]} + \mathbf{b}^{\text{Graph}}\right)$$

where $W^{\text{Graph}} \in \mathbb{R}^{\beta \times \rho}, \mathbf{b}^{\text{Graph}} \in \mathbb{R}^{\beta}$ are network parameters, $\tanh(\cdot)$ is the hyperbolic tangent activation function, and β is a hyperparameter that determines the hidden GC layer size. Thus, information propagated from source nodes $j = 1, \dots, n$ to target node i, based on the summation of intermediate representations, are unweighted and share equal importance.

As stated previously, we propose to extend the standard GC by adding an attention-based sigmoid gating mechanism to control the flow of information via the gating matrix $G \in \mathbb{R}^{n \times n}$. We define G such that

$$G_{i,j} = \sigma(\mathbf{v} \cdot \mathbf{a}^{i,j}) \qquad \text{for } i = 1, \dots, n, j = 1, \dots, n,$$

where $\mathbf{v} \in \mathbb{R}^{\alpha}$ is a network parameter and $\mathbf{a}^{i,j} \in \mathbb{R}^{\alpha}$ is the hidden attention layer composed as a function of the context representation at source node i and target node j; concretely,

$$\mathbf{a}^{i,j} = \tanh\left(W^{\text{Source}} R^{\text{Context}}{}_{[i]} + W^{\text{Target}} R^{\text{Context}}{}_{[j]} + \mathbf{b}^{\text{Attn}}\right),$$

where $W^{\text{Source}}, W^{\text{Target}} \in \mathbb{R}^{\alpha \times \rho}$ and $\mathbf{b}^{\text{Attn}} \in \mathbb{R}^{\alpha}$ are network parameters and α is a hyperparameter that determines hidden attention layer size. Intuitively, the network learns the relevance of node i to node j via the attention $\mathbf{a}^{i,j}$ and outputs a between 0 and 1 at gate $G_{i,j}$. Gate $G_{i,j}$ controls the flow of information from node i to j, where 0 indicates no information is passed and 1 indicates that all information is passed. To integrate the gating mechanism, we simply redefine $\tilde{A} = (A + I) \times G$. In the next two sections, we show how the intermediate representations are used for end-task prediction.

Sequence Labeling

The sequence labeling (SL) task for detecting precipitant drugs and their interaction type is handled by a bidirectional LSTM trained on a combination of the two types of losses: conditional random fields (CRF) and softmax cross entropy (SCE). Using CRFs results in choosing a globally optimal assignment of tags to the sequence, whereas a standard softmax at the output of each step may result in less globally consistent assignments (e.g., an L tag following an O tag) but better local or partial assignments. We begin by introducing a bidirectional LSTM layer that processes the various intermediate representations. The new representation, $R^{\text{SL}} \in \mathbb{R}^{n \times \gamma}$, is defined such that

$$R^{\text{SL}} = f^{\gamma}_{\text{BLSTM}} \begin{pmatrix} R^{\text{Word}}{}_{[1]} \parallel R^{\text{Context}}{}_{[1]} \parallel R^{\text{Graph}}{}_{[1]} \\ \vdots \\ R^{\text{Word}}{}_{[n]} \parallel R^{\text{Context}}{}_{[n]} \parallel R^{\text{Graph}}{}_{[n]} \end{pmatrix}$$

where γ is a hyperparameter that determines the hidden layer size. While R^{Graph} is based on R^{Context} and R^{Context} is based on R^{Word}, we observed that combining these intermediate representations (manifesting at varying depth in the architecture) resulted in improved sequence-labeling performance according to preliminary experiments and prior results from Tran et al. [66]. As with residual networks [78], they additionally provide a kind of shortcut or “skip-connection” over intermediate layers.

Given a set of n_{tag} possible tags, we compose an $n \times n_{\text{tag}}$ score matrix Y (where $Y_{i,t}$ represents the score of the t^{th} tag at position i) such that $Y_{[i]} = W^{\text{Out}} R^{\text{SL}}{}_{[i]} + \mathbf{b}^{\text{Out}}$ where $W^{Out} \in \mathbb{R}^{n_{\text{tag}} \times \gamma}, \mathbf{b}^{\text{Out}} \in \mathbb{R}^{n_{\text{tag}}}$ are network parameters. Given example x and the truth tag assignment as a matrix $\bar{Y}$ where rows are one-hot vectors over all possible tags, the SCE loss is

$$\ell_{\text{SCE}}(x, \tilde{A}, \bar{Y}; \theta) = -\sum_{i=1}^{n} \sum_{t=1}^{n_{\text{tags}}} \bar{Y}_{i,t} \log \left(\frac{\exp(Y_{i,t})}{\sum_{k=1}^{n_{\text{tags}}} \exp(Y_{i,k})} \right)$$

where $\bar{Y}_{i,t} \in \{0, 1\}$ indicates whether the tag t is assigned at position i and θ is the set of all network parameters. Next, we define the CRF loss as commonly used with LSTM based models for entity recognition. We learn a transition score matrix $M \in \mathbb{R}^{n_{\text{tag}} \times n_{\text{tag}}}$, inferred from the training data, such that $M_{i,j}$ is the transition score from tag i to tag j. Given an example x as a sequence of word indices $w_1, \ldots, w_n$ and candidate tag sequence $\bar{y}$ as a sequence of tag indices $s_1, \ldots, s_n$, the tag assignment score (t-score) is defined as

$$\text{t-score}\,(x, \tilde{A}, \bar{y}; \hat{\theta}) = \text{t-score}\,(w_1, \ldots, w_n, \tilde{A}, s_1, \ldots, s_n; \hat{\theta}) = \sum_{i=1}^{n} (Y_{i,s_i} + M_{s_{i-1},s_i})$$

where $\hat{\theta} = \theta \cup \{M\}$. Intuitively, this score summarizes the likelihood of observing a transition from tag s_{i-1} to tag s_i in addition to the likelihood of emitting tag s_i given the semantic context for $i = 1, \ldots, n$. Thus Y is treated as a matrix of emission scores for the CRF. For an example with input x and truth tag assignment $\bar{y}$, the loss is computed as the negative log-likelihood of the tag assignment as informed by the normalized tag assignment score, or

$$\ell_{\text{CRF}}(x, \tilde{A}, \bar{y}; \hat{\theta}) = -\log \frac{\exp(\text{t-score}(x, \tilde{A}, \bar{y}; \hat{\theta}))}{\sum_{y \in S} \exp(\text{t-score}(x, \tilde{A}, y; \hat{\theta}))}$$

where S is the set of all possible tag assignments. The final per-example loss for sequence labeling is simply a summation of the two losses: $\ell_{\text{SL}} = \ell_{\text{SCE}} + \ell_{\text{CRF}}$. During testing, we use the Viterbi algorithm [79], a dynamic programming approach, to decode and identify the globally optimal tag assignment.

Consequence prediction

Once precipitants (and corresponding interaction types) have been identified, we perform so called consequence prediction (CP) for all precipitant drugs identified as participating in PD or PK interactions. The classification task of CP takes as input the target sentence and two candidate entities that are referred to as the *subject* and *object* entities. Here the *subject* is always a precipitating drug; on the other hand, the *object* designation depends on the type of interaction (more later). First, we define the representation matrix for CP as $R^{\text{CP}} \in \mathbb{R}^{n \times (\rho + \beta)}$ where

$$R^{\text{CP}} = \begin{pmatrix} {R^{\text{Context}}}_{[1]} \parallel {R^{\text{Graph}}}_{[1]} \\ \vdots \\ {R^{\text{Context}}}_{[n]} \parallel {R^{\text{Graph}}}_{[n]} \end{pmatrix}.$$

We process the matrix via convolutions of windows sizes 3, 4, and 5 and concatenate the results to produce the final feature vector $\mathbf{g}^{\text{CP}}$. In addition to CNN features, we map entities to their graph based context features and append it to $\mathbf{g}^{\text{CP}}$, which has been previously shown to work well in a similar architecture [5]. Concretely, the final feature vector is

$$\mathbf{g}^{\text{CP}} = f^{3,\mu}_{\text{CNN}}(R^{\text{CP}}) \parallel f^{4,\mu}_{\text{CNN}}(R^{\text{CP}}) \parallel f^{5,\mu}_{\text{CNN}}(R^{\text{CP}}) \parallel R^{\text{CP}}[t_{\text{Sub}}] \parallel R^{\text{CP}}[t_{\text{Obj}}]$$

with $\mathbf{g}^{\text{CP}} \in \mathbb{R}^{3\mu + 2(\rho + \beta)}$ where μ, as a hyperparameter, is the number of CNN filters per convolution and t_{Sub} and t_{Obj} are the position index of the last word (typically the "head" word) of the subject and object respectively.

The actual entities determined to be the subject/object pair are based on the interaction type; for PD interactions, the *subject* is the precipitant drug and the *object* is some candidate *effect* mention. For PK interactions, however, the subject is the precipitant drug but the object is chosen to be the closest (based on character-offset) mention of the drug label with respect to the target precipitant drug. We found this appropriate based on manual review of the data, as the NCI code being assigned depends highly on whether the increase/decrease in functional measurements is with respect to the label drug or the precipitant drug. In case the label drug is not mentioned, a generic "null" vector is used to represent the object.

When performing sequence labeling, we pass in the entire dependency tree encoded as the matrix $\tilde{A}$. However, when performing consequence prediction and both entities are non-null, we pass in a *pruned* version of the entire tree that is tailored to the entity pair. We apply the same pruning strategy proposed by Zhang et al. [31], wherein for a pair of subject and object entities (corresponding to t_{Sub} and t_{Obj}), we keep only nodes either along or *within one hop* of the shortest dependency path. This prevents distant and irrelevant portions of the dependency tree from influencing the model while retaining important modifying and negating terms. Thus the notation $\tilde{A}_{\text{Sub}\leftrightarrow\text{Obj}}$ is used to denote the pruned version of $\tilde{A}$ as a function of the entity pair indicated by t_{Sub} and t_{Obj}.

To determine whether there is a PD interaction between a pair of entities, we employ a standard binary classification output layer. Concretely, for example sentence $\hat{x}$ and output $y \in \{0,1\}$, the probability of a PD interaction between the entity pair is $q = \text{sigmoid}(\mathbf{w}^{\text{PD}} \cdot \mathbf{g}^{\text{CP}} + b^{\text{PD}})$ where $\mathbf{w}^{\text{PD}} \in \mathbb{R}^{3\mu+2(\rho+\beta)}$ and $b^{\text{PD}} \in \mathbb{R}$ are network parameters. The associated binary cross entropy loss is

$$\ell_{\text{PD}}(x, \tilde{A}_{\text{Sub}\leftrightarrow\text{Obj}}, \hat{y}; \theta) = \hat{y} \log q + (1 - \hat{y}) \log(1 - q)$$

where $\hat{y} \in \{0,1\}$ indicates the ground truth. For PK interactions, we instead use a softmax function to produce a probability distribution, represented as vector $\mathbf{q} \in \mathbb{R}^{20}$, over the 20 labels corresponding to NCI Thesaurus codes. Concretely, the predicted probability of label j is $\mathbf{q}_j = \exp(\mathbf{y}_j^{\text{PK}}) / \exp(\sum_{k=1}^{20} \mathbf{y}_k^{\text{PK}})$ where $\mathbf{y}^{\text{PK}} = W^{\text{PK}} \mathbf{g}^{\text{CP}} + \mathbf{b}^{\text{PK}}$ and $W^{\text{PK}} \in \mathbb{R}^{20\times[3\mu+2(\rho+\beta)]}$ and $\mathbf{b}^{\text{PK}} \in \mathbb{R}^{20}$ are network parameters. Given a one-hot vector $\bar{\mathbf{y}} \in \mathbb{R}^{20}$ indicating the ground truth, the associated softmax cross entropy loss is

$$\ell_{\text{PK}}(x, \tilde{A}_{\text{Sub}\leftrightarrow\text{Obj}}, \bar{\mathbf{y}}; \theta) = \sum_{j=1}^{20} \bar{\mathbf{y}}_j \log \mathbf{q}_j \quad .$$

The loss for a batch of examples is simply the sum of its constituent example-based losses.

Table 4.3: Model configuration obtained through random search over 11-fold cross-validation of TR22 (training data).

Setting	**Value**	**Setting**	**Value**
Learning Rate	0.001	Context Embedding Size (ρ)	100
Dropout Rate	0.5	GC Hidden Size (β)	100
Character Embedding Size (π)	25	GC Attention Size (α)	25
Character Representation Size (η)	50	Sequence LSTM Hidden Size (γ)	200
Word Embedding Size (δ)	200	Outcome CNN Filter Count (μ)	50

Neural Network Configuration and Training Details

For each training iteration, we randomly sample 10 sentences from the training data. These are re-composed into three sets of task-specific examples $\mathcal{S}$, $\mathcal{D}$, and $\mathcal{K}$ corresponding to the tasks of sequence labeling, PD prediction, and PK prediction respectively. Unlike our prior work, in which the sub-tasks were trained in an interleaved fashion, we train on all three objectives jointly. Here, we dynamically switch between one of four training objective losses based on whether there are available training examples (in the batch and for the current iteration) for each task. The final training loss is then

$$\ell = \begin{cases} \sum_{x\in\mathcal{S}} \ell_{\text{SL}}(x) + \sum_{x\in\mathcal{D}} \ell_{\text{PD}}(x) + \sum_{x\in\mathcal{K}} \ell_{\text{PK}}(x) & \text{if } |\mathcal{D}| > 0 \text{ and } |\mathcal{K}| > 0, \\ \sum_{x\in\mathcal{S}} \ell_{\text{SL}}(x) + \sum_{x\in\mathcal{K}} \ell_{\text{PK}}(x) & \text{if } |\mathcal{D}| = 0 \text{ and } |\mathcal{K}| > 0, \\ \sum_{x\in\mathcal{S}} \ell_{\text{SL}}(x) + \sum_{x\in\mathcal{D}} \ell_{\text{PD}}(x) & \text{if } |\mathcal{D}| > 0 \text{ and } |\mathcal{K}| = 0, \\ \sum_{x\in\mathcal{S}} \ell_{\text{SL}}(x) & \text{otherwise.} \end{cases}$$

We train the network for a maximum of 10,000 iterations, check-pointing and evaluating every 100 iterations on a validation set of sentences from four held-out drug labels. Only the checkpoint that performed best on the validation set is kept for test time evaluation. The choice of hyperparameters is shown in Table 4.3; discrete numbered parameters corresponding to embedding or hidden size were chosen from {10, 25, 50, 100, 200, 400} based on random search and optimized by assessing 11-fold cross-validation performance on TR22. The learning and dropout rates are set to typical default values. We used Word2Vec embeddings pretrained on the corpus of PubMed abstracts [58]. All other variables are initialized using values drawn from a normal distribution with a mean of 0 and standard deviation of 0.1 and further tuned during training. Words were tokenized on both spaces and punctuation marks; punctuation tokens were kept as is common practice for NER type systems.

For dependency parsing, we use SyntaxNet[3] which implements the transition-based neural model by Andor et al. [80]. We trained the aforementioned parser, using default settings, on the GENIA corpus [81] and use it to obtain projective dependency parses for each example.

4.2.4 Transfer Learning with Network Pre-Training

An obstacle in solving this flavor of DDI extraction as a machine learning problem is the high potential for overfitting given the sparse nature of the output space, which is further intensified by the scarce availability of high quality training data. As quality training data is expensive and requires domain expertise, we propose to use a transfer learning approach where the model is pre-trained on external data as follows. First, we pre-train on the DDI2013 dataset, which contains strictly binary relation DDI annotations and no interaction consequence annotation. Hence, DDI2013 is only used to train the sequence labeling objective $\ell_{\text{SL}}(x)$. Next, we pre-train on NLM180, a collection of 180 drug labels annotated in a comparable format to TR22 but follows a different set of guidelines and lacks comprehensive interaction consequence annotation. Finally, we fine-tune for the target task by training on the official TR22 dataset.

Translating NLM180 and DD2013 to the TAC 2018 format is an imperfect process given structural (breadth and depth of annotations) and semantic (guidelines in addition to annotator experience and vision) differences. For example, differences in how entity boundaries are annotated, such as whether or not modifier terms should be kept as part of a named entity, may have a large impact on model performance. Hence, we expect the translated versions of NLM180 and DDI2013 to be very noisy as training examples for the target task. We describe the translation process for DDI2013 in Sections 4.2.4 and 4.2.4. We provide summary statistics about these datasets in Table 4.1.

NLM180 Mapping Scheme

In NLM180, there is no distinction between triggers and effects; moreover, PK effects are limited to *coarse*-grained (binary) labels corresponding to *increase* or *decrease* in function measurements. Hence, a direct mapping from NLM180 to the TR22 annotation scheme is impossible. As a compromise, NLM180 "triggers" were mapped to TR22 *triggers* in the case of unspecified and PK interactions. For PD interac-

---

[3] https://github.com/tensorflow/models/tree/master/research/syntaxnet

tions, we instead mapped NLM180 “triggers” to TR22 *effects*, which we believe to be appropriate based on our manual analysis of the data. Since we do not have both *trigger* and *effect* for every PD interaction, we opted to ignore trigger mentions altogether in the case of PD interactions to avoid introducing mixed signals. While trigger recognition has no bearing on relation extraction performance, this policy has the effect of reducing the recall upperbound on NER by about 25% based on early cross-validation results. To overcome the lack of fine-grained annotations for PK outcome in NLM180, we deploy the well-known bootstrapping approach [82] to incrementally annotate NLM180 PK outcomes using TR22 annotations as a starting point. To mitigate the problem of semantic drift, we re-annotated by hand iterative predictions that were not consistent with the original NLM180 coarse annotations (i.e., active learning [83]).

DDI2013 Mapping Scheme

The DDI2013 dataset contains annotations that are incomplete with respect to the target task; specifically, annotations are limited to typed binary relations between any two drug mentioned drugs in the sentence (and not necessary between a mentioned drug and the label drug) without outcome or consequence prediction. In DDI2013, there are four types of interactions: *mechanism*, *effect*, *advice* and *int*. The *mechanism* type indicates that a PK mechanism is being discussed; *effect* indicates that the consequence of a PD interaction is being discussed; *advice* indicates suggestions regarding the handling of the drugs; and *int* is an interaction without any specific additional information. We translate the annotation by first applying a filtering step on all interactions such that it conforms to the target task; namely, we filter such that only interactions involving the label drug is kept. The non-label drug entity is then annotated as a precipitant with an interaction tag based on the following mapping scheme. Entities involved in a *mechanism* relation with the drug label are treated as **KIN** precipitants; likewise, entities in *effect* and *advice* relations are treated as **DYN** precipitants and *int* relations are treated as **UNK** precipitants. As there is no consequence annotation, the mapped examples are used to train the sequence labeling objective but *not* the other objective.

4.2.5 Voting-based Ensembling

Our prior effort [66] showed that model ensembling resulted in optimal performance for this task. Hence, model ensembling remains a key component of the proposed model. Our ensembling method is based on ensembling over *ten* models each trained with randomly initialized weights and a random development split. Intuitively, models collectively “vote” on predicted annotations that are kept and annotations that are discarded. A unique annotation (entity or relation) has one vote for each time it appears in one of the *ten* model prediction sets. In terms of implementation, unique annotations are incrementally added (to the final prediction set) in order of descending vote count; subsequent annotations that conflict (i.e., overlap based on character offsets) with existing annotations are discarded. Hence, we loosely refer to this approach as “voting-based” ensembling.

4.2.6 Model Evaluation

We used the official evaluation metrics for NER and relation extraction based on the standard precision, recall, and F1 micro-averaged over exactly matched entity/relation annotations. We use the strictest matching criteria corresponding to the official “primary” metric (of the TAC DDI task), as opposed to the “relaxed” metric that ignores mention and interaction *type*. Concretely, the matching criteria for entity recognition considers entity bounds as well as the type of the entity. The matching criteria for relation extraction comprehensively considers precipitant drugs and, for each, the corresponding interaction type and interaction outcome. As relation extraction evaluation takes into account the bounds of constituent entity predictions, relation extraction performance is heavily reliant on entity recognition performance. On the other hand, we note that while NER evaluation considers *trigger* mentions, *triggers* are ignored when evaluating relation extraction performance. Two test sets of 57 and 66 drug labels, referred to as Test Set 1 and 2 respectively, with gold standard annotations are used for evaluation.

Table 4.4: Main results based on 95% confidence interval around mean precision, recall, and F1 based on evaluating N=100 ensembles for each model.

		Test 1 / Entity			Test 1 / Relation			Test 2 / Entity			Test 2 / Relation			Overall		
Method	**Training Data**	**P**	**R**	**F** (%)	**P**	**R**	**F** (%)	**P**	**R**	**F** (%)	**P**	**R**	**F** (%)	**P**	**R**	**F** (%)
BL	TR22	23.82	42.04	30.39	14.74	18.38	16.35	26.15	39.69	31.51	12.48	15.43	13.79	19.30 ± 0.12	28.88 ± 0.12	23.01 ± 0.06
GCA	TR22	32.87	32.35	32.59	22.70	13.95	17.27	38.82	31.31	34.65	19.26	11.63	14.49	28.41 ± 0.09	22.31 ± 0.16	24.75 ± 0.11
BL(1)	TR22 + NLM180	27.05	39.87	32.22	19.94	22.20	21.00	32.49	41.92	36.60	21.82	23.93	22.82	25.32 ± 0.09	31.98 ± 0.11	28.16 ± 0.06
GCA	TR22 + NLM180	38.30	31.20	34.38	27.97	15.14	19.63	44.13	31.18	36.53	31.79	15.76	21.06	35.55 ± 0.18	23.32 ± 0.20	27.90 ± 0.17
BL	TR22 + NLM180 + DDI2013	29.27	41.93	34.47	22.93	**25.42**	24.11	38.73	43.79	41.10	27.11	**27.32**	27.21	29.51 ± 0.10	34.61 ± 0.10	31.72 ± 0.06
GCA	TR22 + NLM180 + DDI2013	**41.58**	38.24	**39.83**	**31.84**	20.49	24.93	**47.54**	36.12	41.04	**32.07**	17.81	22.90	**38.26** ± 0.16	28.17 ± 0.12	32.18 ± 0.12
GC(2)	TR22 + NLM180 + DDI2013	38.85	36.30	37.52	29.82	18.59	22.88	43.74	34.88	38.80	31.14	16.40	21.48	35.89 ± 0.20	26.54 ± 0.20	30.17 ± 0.19
GCA + BL(3)	TR22 + NLM180 + DDI2013	35.22	**44.23**	39.20	27.58	24.77	**26.09**	45.50	**45.10**	**45.30**	31.69	24.89	**27.87**	35.00 ± 0.15	**34.75** ± 0.13	**34.61** ± 0.10

(1) Our original challenge submission using a BiLSTM-based approach and trained on only TR22 and NLM180.
(2) For reference, we include an evaluation of the standard GC without attention-gating.
(3) Our current best is a combination of GCA and BL by ensembling.

Next, we discuss the differences between these test sets. As shown in Table 4.1, Test Set 1 closely resembles TR22 with respect to the sections that are annotated. However, Test Set 1 is more sparse in the sense that there are more sentences per drug label (144 vs. 27), with a smaller proportion of those sentences having gold annotations (23% vs. 51%). Test Set 2 is unique in that it contains annotations from only two sections, namely DRUG INTERACTIONS and CLINICAL PHARMACOLOGY, the latter of which is not represented in TR22 (nor Test Set 1). Lastly, TR22, Test Set 1, and Test Set 2 all vary with respect to the distribution of interaction types, with TR22, Test Set 1, and Test Set 2 containing a higher proportion of PD, UN, and PK interactions respectively. Overall model performance is assessed using a single metric defined as the average of *entity recognition* and *relation extraction* performance across both test sets.

Table 4.5: Comparison of our method with comparable (based on training data) methods of teams in the top 5 trained on solely TR22 + NLM180.

	Test 1 / Entity			Test 1 / Relation			Test 2 / Entity			Test 2 / Relation		
Method	**P**	**R**	**F** (%)	**P**	**R**	**F** (%)	**P**	**R**	**F** (%)	**P**	**R**	**F** (%)
Dandala et al. [76]	41.94	23.19	29.87	25.24	16.10	19.66	44.61	29.31	35.38	22.99	16.83	19.43
Tran et al. [66]	29.50	37.45	33.00	22.08	21.13	21.59	36.68	40.02	38.28	22.53	21.13	23.55
BL + GCA (Ours)	**32.89**	**41.06**	**36.51**	**24.66**	**21.35**	**22.87**	**40.57**	**42.44**	**41.47**	**28.15**	**22.42**	**24.95**

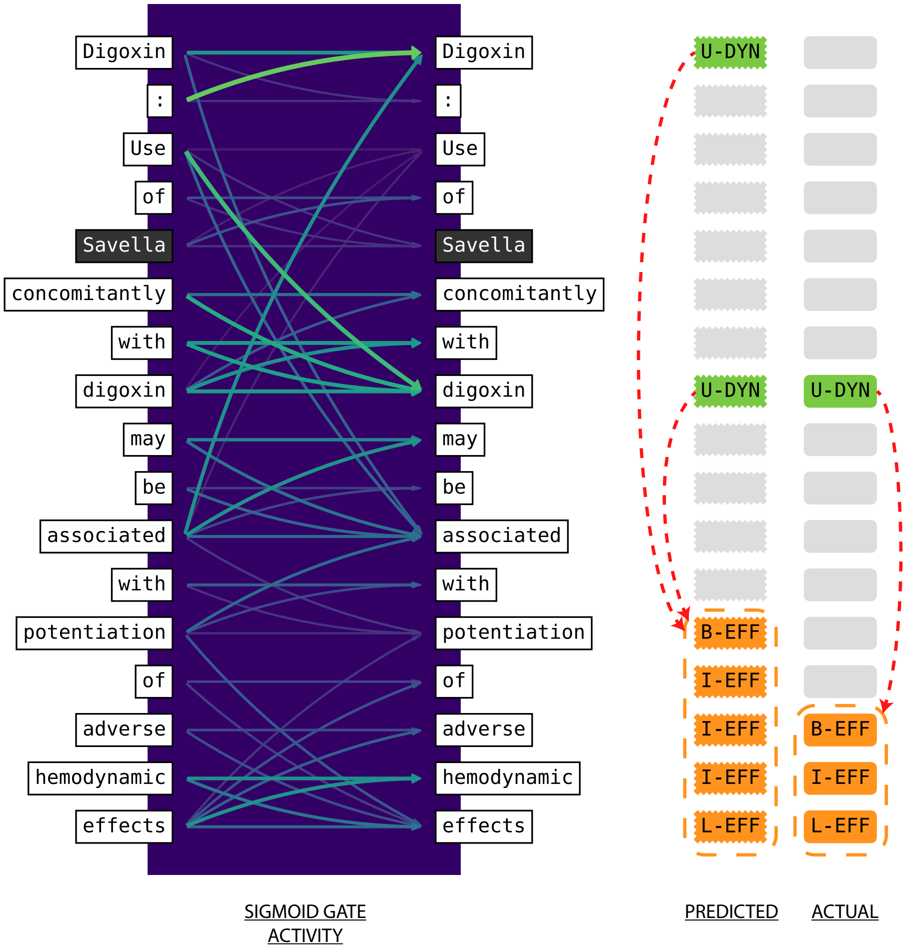
4.3 Results and Discussion

In order to assess model performance with confidence intervals and draw conclusions based on statistical significance, we perform a technique called *bootstrap ensembling* proposed by Kavuluru et al. [28]. That is, for each neural network (NN), we train a pool of 30 models each with a different set of randomly initialized weights and training-development set split. Performance of the NN is evaluated based on computing the 95% confidence interval around the mean F1 of $N = 100$ ensembles, where each ensemble is assembled from a set of ten models randomly sampled from the pool. This approach allows us to better assess average performance which is a nontrivial task given the high variance nature of models learned with limited training data. Our method for model ensembling (by "voting") is described in Section 4.2.5.

We present the main results of this study in Table 4.4 where we compare our prior efforts using strictly BiLSTMs (BL) and our current best results with graph convolutions (GCA). BL with TR22 and NLM180 as training data corresponds to our prior best at 28.16% overall F1, while GCA with TR22, NLM180, and DDI2013 as training data represents our current best at 32.18% overall F1 based on graph convolutions. Here, we observe a 4 point gain in overall F1 (statistically significant at 95% confidence level based on non-overlapping confidence intervals), with most gains owing to a substantial improvement in entity recognition performance. We note that GCA is more precision focused while BL is more recall focused; moreover, GCA tends to exhibit better performance on Test Set 1, while BL tends to exhibit better performance on Test Set 2. This hints that the two architectures are highly complementary and may work well in combination. Indeed, when combined via ensembling, we observe a major performance gain across almost all measures. Here, for each ensemble, we sample five models from each pool of models (GCA and BL) for a total of ten models to ensure that results remain comparable. The resulting hybrid model exhibits the best performance overall, improving over the prior best by two points and over the current best by six points in overall F1 at 34.61%. These differences are statistically significant at the 95% confidence level. Next, we highlight that a main benefit of the GCA model is that it operates well with very small amounts of training data, as evident by the almost 2 absolute point improvement over the BiLSTM model when trained solely on TR22. These gains tend to be less notable when we involve examples from NLM180 and DDI2013. Lastly, we note that GCA (graph convolution with attention-gating) performs better than the standard GC (graph convolution *without* attention-gating) by two absolute points in overall F1 with improvements that are consistent across all metrics. We present a comparison of our results with other works in Table 4.5. We omit results by Tang et al. [75] as they are not directly comparable to ours given the stark difference in available training data. When training on strictly TR22 and NLM180 (thus being comparable to most prior work), our model exhibits state-of-the-art performance across all metrics on either test sets.

We present Figures 4.3 and 4.4 to illustrate error cases to be discussed later in Section 4.4. In additional to actual and predicted annotations, these figures include a sigmoid gating activity visualization for edges in the dependency tree. The visualization serves two purposes. First, it confirms the intuition for this particular design and, second, provides a means to interpret model decisions. That is, we can observe the importance of each edge in the dependency tree as deemed by the network for a particular example. In Figure 4.3, for example, we can observe that for the tar-

Figure 4.3: An example sentence from the drug label for Savella along with the resulting prediction and ground truth labels. Red arrows indicate interaction outcome.

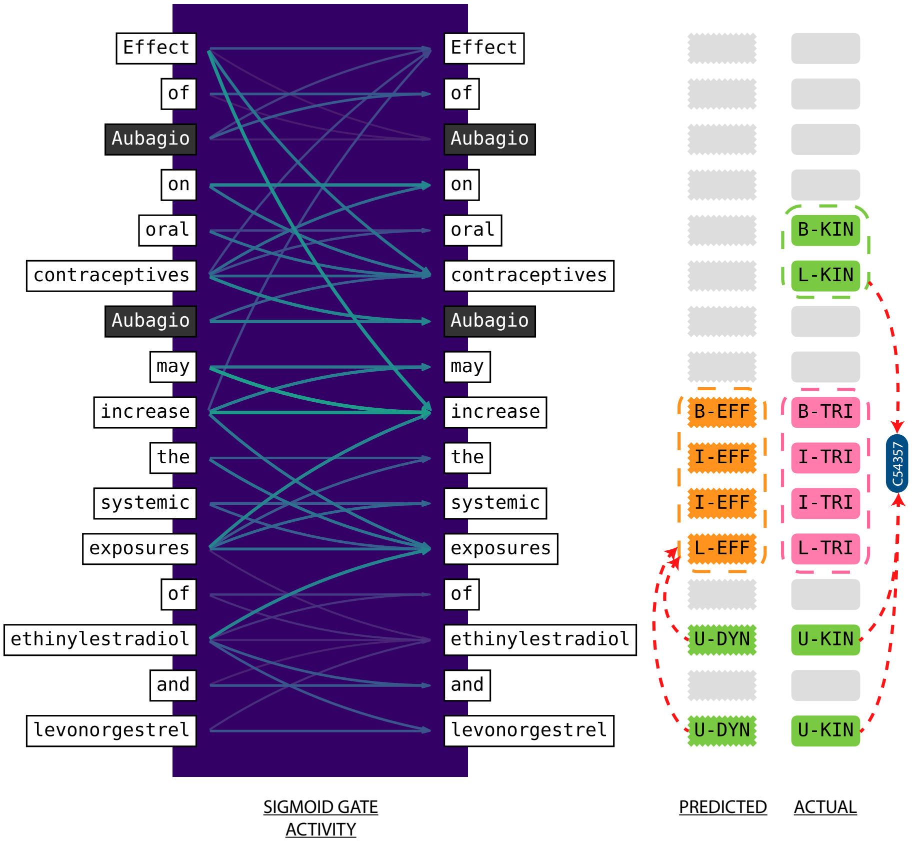


get word "digoxin" (which is a precipitant, the second occurrence in the sentence), the phrase "use", "concomitantly", and "with" show very high activity. Likewise, signal flow from "hemodynamic" to "effects" is strong, and vice versa. Less important words such as articles appear to receive less incoming activity overall, even through self-loops.

4.4 Error Analysis

In this section, we perform error analysis to identify challenging cases typically resulting in erroneous predictions by the model. One major source of difficulty for the model is boundary detection in cases of multi-word entities. Errors of this type are especially prominent in case of *effect* mentions which may manifest as potentially long noun phrases. Phrases with conjunctions or punctuation marks (or a combination of)

Figure 4.4: An example sentence from the drug label for Aubagio along with the resulting prediction and ground truth labels. Red arrows indicate interaction outcome, where C54357 is a PK label corresponding to the NCI Thesaurus code for "Increased Concomitant Drug Level."



may also present an obstacle for the model; for example, an *effect* expressed as "serious and/or life threatening reaction" may instead be predicted as simply "life threatening reaction." Figure 4.3 shows a general case of this error where the model recognizes "potentiation of adverse hemodynamic effects" as the effect while the ground truth identifies the effect as simply "adverse hemodynamic effects." This leads to both a false positive and a false negative for both the NER and the RE evaluation. We note that, given the potentially limitless ways an *effect* may be expressed, any disagreement among annotators (for cases beyond those addressed in annotator guidelines) during the initial annotation process will lead to inconsistent ground truth data and thus negatively affect downstream model performance. As an example, consider the following two sentences that appear in TR22: "Co-administration of SAMSCA with `potent CYP3A inducers` .." and "For patients chronically taking potent `inducers of CYP3A`, .." Here, one sentence is annotated such that *potent* is included as part of the precipitant expression, while another is annotated such that this modifier is excluded.

Mixed signals and noisy labels in general tend to be an issue especially when there is limited training data as deep learning models are prone to overfitting. When evaluating on purely effect mentions, we obtain a micro-F1 score of 66% (54% Precision, 87% Recall). However, the micro F1 is 87% when ignoring the starting boundary offset and 86% when ignoring the ending boundary offset during evaluation corresponding to roughly 20 absolute micro-F1 gain in performance. When applying the same looser evaluation criteria to triggers and precipitants, the gains are only ≈ 6% and ≈ 5% respectively. Thus there is immense potential for improving entity recognition of *effect* mentions if we can better handle boundary detection, possibly via rule-based methods or post-processing adjustments, with the added benefit of improving consequence prediction performance for PD interactions.

Precipitants interacting with the label drug being mentioned multiple times may also cause issues for the model. As an example, consider the sentence presented in Figure 4.3. Our model identifies both mentions of the precipitant "Digoxin" as being involved in an interaction with the drug Savella; however, the ground truth more specifically recognizes the second mention as the sole precipitant. This results in an additional false positive with respect to both NER and RE evaluation. Lastly, there are cases where the model will mistake a mention subtly referring to the label drug as a precipitant. This is a common occurrence in cases where the label drug is not referred to by name, but by a class of drugs. Typically, identifying a mention as a reference to the drug label beforehand will disqualify it from being predicted as a precipitant. While we do use a lexicon of drug names mapped to drug synonyms and drug classes to identify these indirect mentions, it is not exhaustive for all drugs. For example, within the label of the drug Lexapro, consider the sentence "Altered anticoagulant effects, including increased bleeding, have been reported when SSRIs and SNRIs are coadministered with warfarin." Here, the model recognized SSRI and SNRI as precipitants. This is incorrect, however, as Lexapro is an SSRI and these mentions are more than likely referring to Lexapro. Without this information, the model likely assumes that it is an implicit case where the label drug is not mentioned and therefore assume all drug mentions are precipitants. Hence, curating a more exhaustive lexicon for indirectly mentions of the label drug will improve overall performance.

Lastly, we describe a source of difficulty stemming from incorrectly classifying interaction types. Figure 4.4 presents an example sentence where our model mistakes PK for PD interactions and a *trigger* mention for an *effect* mention. As PD and PK interactions tend to frequently co-occur with *effect* and *trigger* mentions respectively,

Table 4.6: Confusion matrix for interaction type

		Predicted		
		PD	PK	UN
Actual	PD	788	37	68
	PK	57	353	147
	UN	170	10	599

predicted annotations tend to be polarized toward one pair (PD with *effect*) or the other (PK with *trigger*). Hence, differentiating between types of interactions for each recognized precipitant is another interesting class of error. Among all correctly recognized precipitants (based purely on boundary detection), we analyzed cases where one type of interaction, among PD, PK, and Unspecified (UN), is mistaken for another via the confusion matrix in Table 4.6. Clearly, many errors are due to cases where (1) we mistake *unspecified* precipitants for PD precipitants and (2) we mistake PK precipitants for *unspecified* precipitants. We conjecture that making precise implicit connections (not only whether there is evidence in the form of trigger words or phrases, but whether the evidence concerns the particular precipitant) is highly nontrivial. Likely, this aspect may be improved by inclusion of more high quality training data. Confusion between *trigger* and *effect* mentions is less concerning; among more than 1000 cases, there are six cases where we mistake *effect* for *trigger* and 20 cases where we mistake *trigger* for *effect*.

4.5 Conclusion

In this chapter, we proposed an end-to-end method for extracting drugs and their interactions from drug labels, including interaction outcome in the case of PK and PD interactions. The method involved composing various intermediate representations including sequential and graph based context, where the latter is produced using a novel attention-gated version of the graph convolution over dependency parse trees. The so called graph convolution with attention-gating (GCA), along with transfer learning via serial pre-training using other annotated DDI datasets including DDI2013, resulted in an improvement over our original TAC challenge entry by up to 6 absolute F1 points overall. Among comparable studies (based on training data composition), our method exhibits state-of-the-art performance across all metrics and test sets. Future work will focus on curating more quality training data and leveraging semi-supervised methods overcome the scarcity in training data.

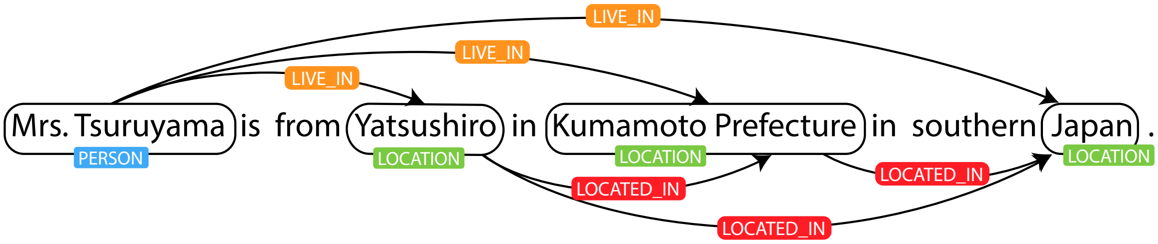
Chapter 5 Neural Metric Learning for Fast End-to-End Relation Extraction

Information extraction (IE) systems are fundamental to the automatic construction of knowledge bases and ontologies from unstructured text. While important, in and of themselves, these resulting resources can be harnessed to advance other important language understanding applications including knowledge discovery and question answering systems. Among IE tasks are named entity recognition (NER) and binary relation extraction (RE) which involve identifying named entities and relations among them, respectively, where the latter is typically a set of triplets identifying pairs of related entities and their relation types.

We present Figure 5.1 as an example of the NER and RE problem given the input sentence “Mrs. Tsuruyama is from Yatsushiro in Kumamoto Prefecture in southern Japan.” First, we extract as entities the spans “Mrs. Tsuruyama”, “Yatsushiro”, “Kumamoto Prefecture”, and “Japan” where “Mrs. Tsuruyama” is of type `PERSON` and the rest are of type `LOCATION`. Thus, NER consists of identifying both the bounds and type of entities mentioned in the sentence. Once entities are identified, the next step is to extract relation triplets of the form `(subject,predicate,object)`, if any, based on the context; for example, `(Mrs. Tsuruyama, LIVE_IN, Yatsushiro)` is a relation triple that may be extracted from the example sentence as output of an RE system. Given this, it is clear that $\mathcal{E}2\mathcal{E}$RE is a complex problem given the sparse nature of the output space; for a sentence of n length with k possible relation types, the output is a variable-length set of relations each drawn from kn^2 possible relation combinations.

NER and RE have been traditionally treated as independent problems to be solved separately and later combined in an ad-hoc manner as part of a pipeline system. End-to-end RE ($\mathcal{E}2\mathcal{E}$RE) is a relatively new research direction that seeks to model NER and RE jointly in a unified architecture. As these tasks are closely intertwined, joint models that simultaneously extract entities and their relations in a single framework have the capacity to exploit inter-task correlations and dependencies leading to potential performance gains. Moreover, joint approaches, like our method, are better equipped to handle datasets where entity annotations are non-exhaustive (that is, only entities involved in a relation are annotated), since standalone NER systems are not designed to handle incomplete annotations. Recent advancements in deep learning for $\mathcal{E}2\mathcal{E}$RE are broadly divided into two categories: (1). The first category

Figure 5.1: A simple relation extraction example.



involves applying deep learning to the table structure first introduced by Miwa and Sasaki [3], including Gupta et al. [84], Pawar et al. [43], and Zhang et al. [85] where $\mathcal{E}2\mathcal{E}$RE is reduced to some variant of the table-filling problem such that the (i, j)-th cell is assigned a label that represents the relation between tokens at positions i and j in the sentence. We further describe the table-filling problem in Section 5.2.1. Recent approaches based on the table structure operate on the idea that cell labels are dependent on features or predictions derived from preceding or adjacent cells; hence, the table is filled incrementally leading to potential efficiency issues. Also, these methods typically require an additional expensive decoding step, involving beam search, to obtain a globally optimal table-wide label assignment. (2). The second category includes models where NER and RE are modeled jointly with shared components or parameters without the table structure. Even state-of-the-art methods not utilizing the table structure rely on conditional random fields (CRFs) as an integral component of the NER subsystem where Viterbi algorithm is used to decode the best label assignment at test time [45, 48].

Our model utilizes the table formulation by embedding features along the third dimension. We overcome efficiency issues by utilizing a more efficient and effective approach for deep feature aggregation such that local metric, dependency, and position based features are simultaneously pooled — in a 3×3 cellular window — over many applications of the 2D convolution. Intuitively, preliminary decisions are made at earlier layers and corroborated at later layers. Final label assignments for both NER and RE are made simultaneously via a simple *softmax* layer. Thus, computationally, our model is expected to improve over earlier efforts without a costly decoding step. We validate our proposed method on the CoNLL04 dataset [2] and the ADE dataset [86], which correspond to the general English and the biomedical domain respectively, and show that our method improves over prior state-of-the-art in $\mathcal{E}2\mathcal{E}$RE. We also show that our approach leads to training and testing times that are seven to ten times faster, where the latter can be critical for time-sensitive end-user applications. Lastly, we perform extensive error analyses and show that our network is visually interpretable by examining the activity of hidden pooling layers (corresponding to intermediate decisions). To our knowledge, our study is the first to perform this type of visual analysis of a deep neural architecture for end-to-end relation extraction.

5.1 Background and Related Work

Li and Ji [4] proposed one of the first truly joint models wherein entities, including entity mention bounds, and their relations are predicted. Structured perceptrons [37], as a learning framework, are used to estimate feature weights while beam search is used to explore partial solutions to incrementally arrive at the most probable structure. Miwa and Sasaki [3] proposed the idea of using a table representation which simplifies the task into a table-filling problem such that NER and relation labels are assigned to cells of the table; the aim was to predict the most probable label assignment to the table, out of all possible assignments, using beam search. While the representation is in table form, beam search is performed sequentially, one cell-assignment per step. The table-filling problem for $\mathcal{E}2\mathcal{E}$RE has since been successfully transferred to the deep neural network setting [84, 43, 85].

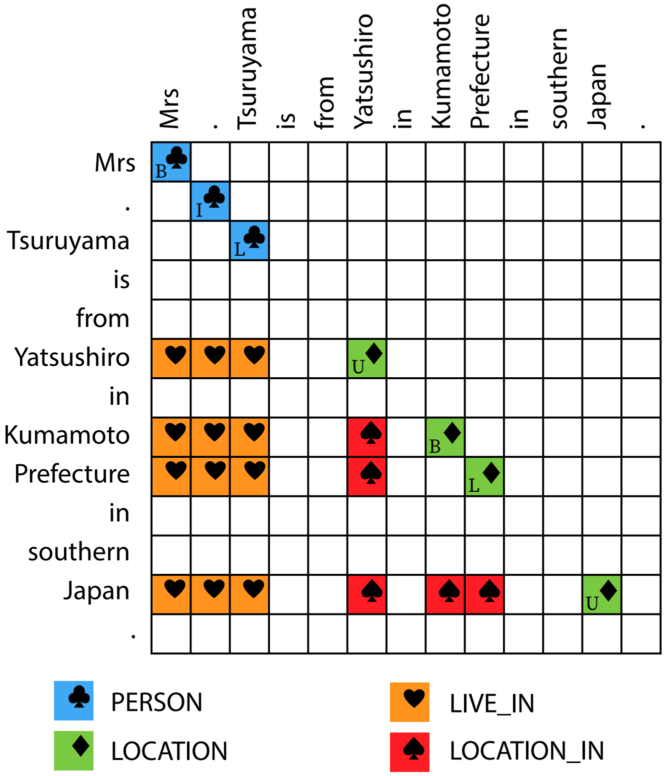
5.2 Methodology

We present our version of the table-filling problem, a novel neural network architecture to fill the table, and details of the training process. Here, Greek letter symbols are used to distinguish hyper-parameters from variables that are learned during training.

5.2.1 The Table-Filling Problem

Given a sentence of length n, we use an $n \times n$ table to represent a set of semantic relations such that the (i, j)-th cell represents the relationship (or non-relation) between tokens i and j. In practice, we assign a tag for each cell in the table such that entity tags are encoded along the diagonal while relation tags are encoded at non-diagonal cells. For entity recognition, we use the BILOU tagging scheme [77]. In the BILOU scheme, B, I, and L tags are used to indicate the beginning, inside, and last token of a multi-token entity respectively. The O tag indicates whether the token outside of an entity span, and U is used for unit-length entities.

Figure 5.2: Table representation for the example in Figure 5.1. BILOU-encoded entity tags are assigned along the diagonal and relation tags are assigned where entity spans intersect. Empty cells are implicitly assigned the O tag.

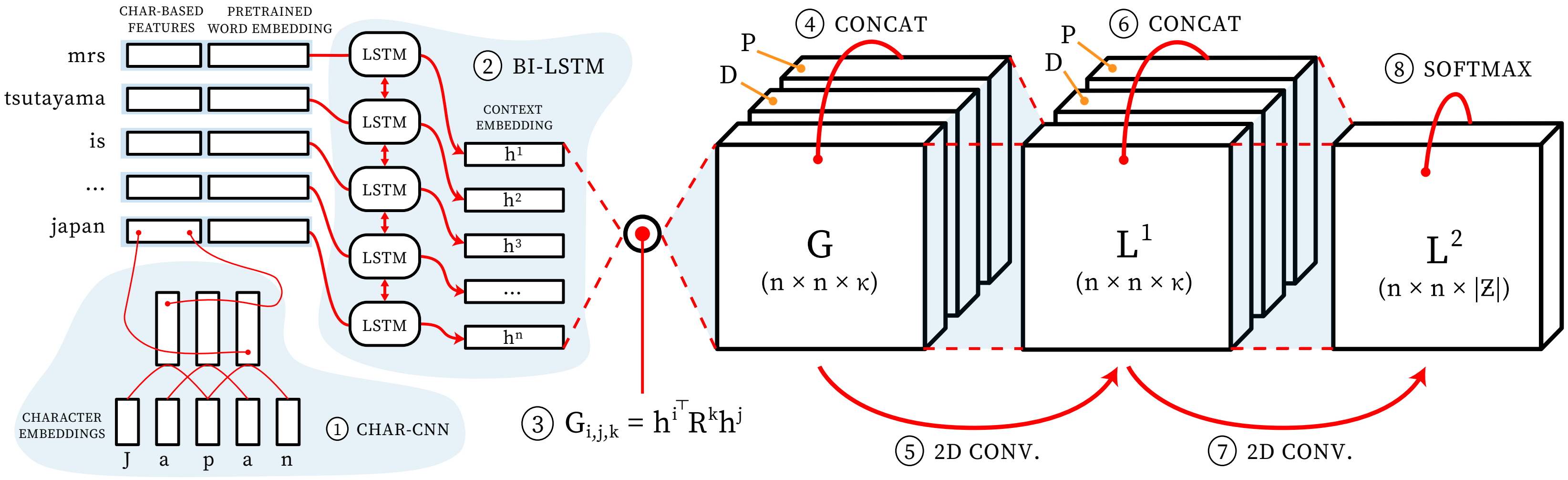


	Mrs	.	Tsuruyama	is	from	Yatsushiro	in	Kumamoto	Prefecture	in	southern	Japan	.
Mrs	B♣												
.		I♣											
Tsuruyama			L♣										
is													
from													
Yatsushiro	♥	♥	♥			U♦							
in													
Kumamoto	♥	♥	♥			♠		B♦					
Prefecture	♥	♥	♥			♠			L♦				
in													
southern													
Japan	♥	♥	♥			♠		♠	♠			U♦	
.													

♣ PERSON
♦ LOCATION
♥ LIVE_IN
♠ LOCATION_IN

In tabular form, entity and relation tags are drawn from a unified list $\mathcal{Z}$ serving as the label space; that is, each cell in the table is assigned exactly one tag from $\mathcal{Z}$. For simplicity, the O tag is also used to indicate a *null* relation when occurring outside of a diagonal. As each entity type requires a BILOU variant, a problem with n_{ent} entity types and n_{rel} relation types has $|\mathcal{Z}| = 4n_{\text{ent}} + n_{\text{rel}} + 1$ where the last term accounts for the O tag. Our conception of the table-filling problem differs from Miwa and Sasaki [3] in that we utilize the entire table as opposed to only the lower triangle; this allows us to model directed relations without the need for additional inverse-relation tags. Moreover, we assign relation tags to cells where entity spans intersect instead of where head words intersect; thus encoded relations manifest as rectangular blocks in the proposed table representation. We present a visualization of our table representation in Figure 5.2. At test time, entities are first extracted, and relations are subsequently extracted by averaging the output probability estimates of the blocks where entities intersect. We describe the exact procedure for extracting relations from these blocks at test-time in Section 5.2.3.

Figure 5.3: Overview of the network architecture for $\lambda = 2$. For simplicity, we ignore punctuation tokens.



5.2.2 Our Model: Relation-Metric Network

We propose a novel neural architecture, which we call the relation-metric network, combining the ideas of metric learning and convolutional neural networks (CNNs) for table filling. The schematic of the network is shown in Figure 5.3, whose components will be detailed in this section.

Context Embeddings Layer

In addition to word embeddings, we employ character-CNN based representations as commonly observed in recent neural NER models [55] and $\mathcal{E}2\mathcal{E}$RE models [5]. Character-based features can capture morphological features and help generalize to out-of-vocabulary words. For the proposed model, such representations are composed by convolving over character embeddings of size π using a window of size 3, producing η feature maps; the feature maps are then max-pooled to produce η-length feature representations. As our approach is standard, we refer readers to Chiu and Nichols [55] for full details. This portion of the network is illustrated in step ① of Figure 5.3.

Suppose the input is a sentence of length n represented by a sequence of word indices $w_1, \ldots, w_n$ into the vocabulary $\mathcal{V}^{\text{Word}}$. Each word is mapped to an embedding vector via embedding matrices $E^{\text{Word}} \in \mathbb{R}^{|\mathcal{V}^{\text{Word}}| \times \delta}$ such that δ is a hyperparameter that determines the size of word embeddings. Next, let $C_{[i]}$ be the character-based representation for the i^{th} word. An input sentence is represented by matrix S wherein rows are words mapped to their corresponding embedding vectors; or concretely,

$$S = \begin{pmatrix} {E^{\text{Word}}}_{[w_1]} \parallel C_{[1]} \\ \vdots \\ {E^{\text{Word}}}_{[w_n]} \parallel C_{[n]} \end{pmatrix}$$

where $\parallel$ is the vector concatenation operator and ${E^{\text{Word}}}_{[i]}$ is the i^{th} row of E^{Word}.

Next, we compose *context* embedding vectors (CVs) as $H = f^{\rho}_{\text{BLSTM}}(S)$ where f^{ρ}_{BLSTM} is a BiLSTM composition previously defined in Section 2.3. This concludes step ② of Figure 5.3.

Relation-Metric Learning

Our goal is to design a network such that any two CVs can be compared via some "relatedness" measure; that is, we wish to learn a relatedness measure (as a parameterized function) that is able to capture correlative features indicating semantic relationships. A common approach in metric learning to parameterize a relatedness function is to model it in bilinear form. Here, for input vectors $\mathbf{x}, \mathbf{z} \in \mathbb{R}^m$, a similarity function in bilinear form is formally defined as

$$s_R(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top R \mathbf{z} \tag{5.1}$$

where $R \in \mathbb{R}^{m \times m}$ is a parameter of the relatedness function, dubbed a *relation-metric embedding* matrix, that is learned during the training process.

In machine learning research, Eq. 5.1 is also associated with a type of attention mechanism commonly referred to as "multiplicative" attention [87]. However, we apply Eq. 5.1 with the classical goal of learning a variety of metric-based features. Our aim is to compute s_R for all pairs of CVs in the sentence. Concretely, we can compute a "relational-metric table" $G \in \mathbb{R}^{n \times n}$ over all pairs of CVs in the sentence such that $G_{i,j} = \mathbf{h}^{i\top} R \mathbf{h}^j$. In fact, we can learn a collection of κ similarity functions corresponding to κ relation metric tables; for our purposes, this is analogous to learning a diverse set of convolution filters in the context of CNNs. Thus we have the 3-dimensional tensor

$$G_{i,j,k} = \mathbf{h}^{i\top} R^k \mathbf{h}^j, \qquad \text{for } k = 1, \dots, \kappa, \tag{5.2}$$

with $G \in \mathbb{R}^{n \times n \times \kappa}$ where the first and second dimension correspond to word position indices while the third dimension embeds metric-based features. This constitutes step ③ of Figure 5.3. We show how G is consumed by the rest of the network in Section 5.2.2. However, as a prerequisite, we first describe how dependency parse and relative position information is prepared in Section 5.2.2 and Section 5.2.2 respectively and define the 2D convolution in Section 5.2.2.

Dependency Embeddings Table

Let $\mathcal{V}^{\text{dep}}$ be the vocabulary of syntactic dependency tags (e.g., `nsubj, dobj`). For an input sentence, let $\mathcal{T} = \{(a_1, b_1, z_1), \dots, (a_{\hat{d}}, b_{\hat{d}}, z_{\hat{d}})\}$ be the set of dependency relations where z_i are mappings to tags in $\mathcal{V}^{\text{dep}}$ that express the dependency-based relations between pairs of words at positions $a_i, b_i \in \{1, \dots, n\}$, respectively. We define the dependency embedding matrix as $F^{\text{dep}} \in \mathbb{R}^{|\mathcal{V}^{\text{dep}}| \times \beta}$, where each unique dependency tag is a β-dimensional embedding. We compose the dependency representation tensor D for $\mathcal{T}$ as

$$D_{i,j,k} = \begin{cases} F^{\text{dep}}_{t,k} & \text{if } (i,j,t) \in \mathcal{T} \text{ or } (j,i,t) \in \mathcal{T}, \\ \phi_k & \text{otherwise,} \end{cases}$$

for $k = 1, \dots, \beta$, where ϕ is a trainable embedding vector representing the *null* dependency relation. As shown in the above equation for $D_{i,j,k}$, we embed the dependency parse tree simply as an undirected graph.

Position Embeddings Table

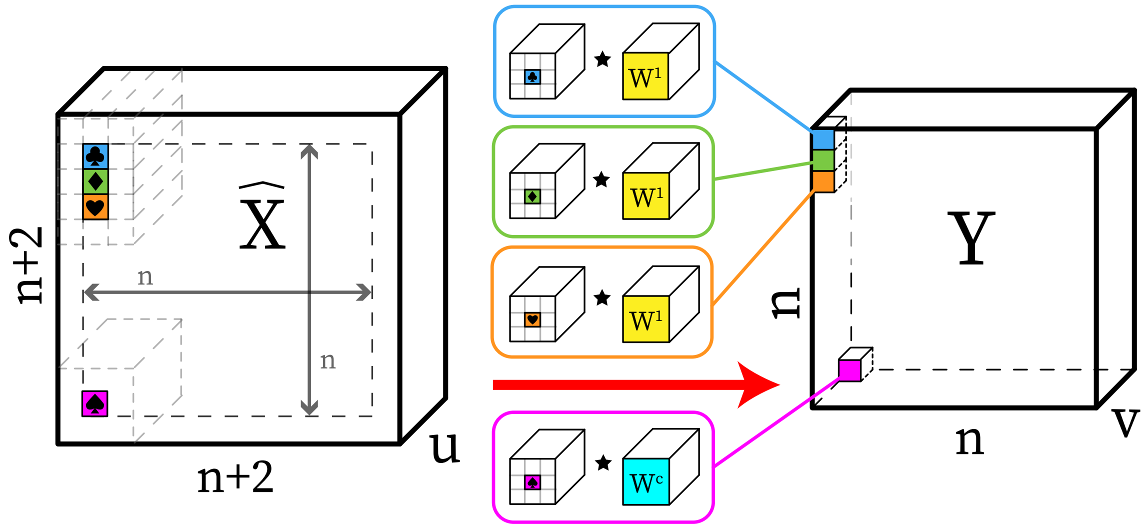
First proposed by Zeng et al. [88], so called *position vectors* have been shown to be effective in neural models for relation classification. Position vectors are designed to encode the relative offset between a word and the two candidate entities (for RE) as fixed-length embeddings. We bring this idea to the tabular setting by proposing a *position* embeddings table P, which is composed the same way as the dependencies table; however, instead of dependency tags, we simply encode the distance between two candidate CVs as discrete labels mapped to fixed-length embeddings (of size γ, a hyperparameter). It is straightforward to see there will be $2(n_{\max} - 1) + 1$ distinct position offset labels where $n_{\max}$ is the maximum length of a sentence in the training data. Specifically, given a position vocabulary $\mathcal{V}^{\text{dist}}$, associated position embedding matrix $F^{\text{dist}} \in \mathbb{R}^{|\mathcal{V}^{\text{dist}}| \times \gamma}$, the position embeddings tensor is $P_{i,j,k} = F^{\text{dist}}_{(i-j),k}$ for $k = 1, \ldots, \gamma$. As an implementation detail, we set $\mathcal{V}^{\text{dist}}$ to $\{-n_{\max}, \ldots, n_{\max}\}$ where $n_{\max}$ is the maximum sentence length over all training examples. Both dependency and position embedding tensors are concatenated to the metric tensor (Eq. (5.2)) along the 3rd dimension prior to every convolution operation. Hence they are shown in steps ④ and ⑥ of Figure 5.3 for the network with two convolutional layers.

2D Convolution Operation

Unlike the standard 2D convolution typically used in NLP tasks, which takes 2D input, our 2D convolution operates on 3D input commonly seen in computer vision tasks where colored image data has height, width, and an additional dimension for color channel. The goal of the 2D convolution is to pool information within a 3×3 window along the first two dimensions such that metric features and dependency/positional information of adjacent cells are pooled locally over several layers. However, it is necessary to perform a *padded* convolution to ensure that dimensions corresponding to word positions are not altered by the convolution. We denote this padding transformation using the *hat* accent. That is, for some tensor input $X \in \mathbb{R}^{n \times n \times m}$, the padded version is $\widehat{X} \in \mathbb{R}^{(n+2) \times (n+2) \times m}$ and the zero-padding exists at the beginning and at the end of the first and second dimensions. Next, we define the 2D convolution operation via the $\star$ operator which corresponds to an element-wise product of two tensors followed by summation over the products; formally, for two input tensors A and B, $A \star B = \sum_i \sum_j \sum_k A_{i,j,k} B_{i,j,k}$.

Now our 2D convolution step is a tensor map $f_v(X) : \mathbb{R}^{n \times n \times u} \to \mathbb{R}^{n \times n \times v}$ with v

Figure 5.4: 2D convolution on 3D input with padding



filters of size $3 \times 3 \times u$, defined as

$$f_v(X)_{i,j,k} = W^k \star \widehat{X}_{[i:i+2][j:j+2][1:u]} + \mathbf{b}_k \tag{5.3}$$

for $i = 1, \ldots, n, j = 1, \ldots, n, k = 1, \ldots, v$, where $W^k \in \mathbb{R}^{3\times3\times u}$ for $k = 1, \ldots, v$, and $\mathbf{b} \in \mathbb{R}^v$ are filter and bias variables respectively, and $\widehat{X}_{[i:i+2][j:j+2][1:u]}$ is a $3 \times 3 \times u$ window of $\widehat{X}$ from i to $i+2$ along the first dimension, j to $j+2$ along the second dimension, and 1 to u along the final dimension. We show how $f_v(X)$ is used to repeatedly pool contextual information in Section 5.2.2. Instead of a 3×3 window, the convolution operation can be over any $t \times t$ window for some odd $t \geq 3$ where large t values lead to larger parameter spaces and multiplication operations. The 2D convolution is illustrated in Figure 5.4 and manifests in steps ⑤ and ⑦ of Figure 5.3.

Pooling Mechanism

Central to our architecture is the iterative *pooling* mechanism designed so that preliminary decisions are made in early iterations and further corroborated in subsequent iterations. It also facilitates the propagation of local metric and dependency/positional features to neighboring cells. Let $\mathcal{Z}$ be the set of tags for the target task. We denote hyper-parameters κ and λ as the number of channels and the number of CNN layers respectively, where κ is same hyperparameter previously defined to represent the size of metric-based features. The pooling layers are defined recursively with base case

$L^1 = \text{relu}(f_\kappa(G \parallel D \parallel P))$ and

$$L^i = \begin{cases} \text{relu}(f_\kappa(L^{i-1} \parallel D \parallel P)) & 1 < i < \lambda, \\ f_{|\mathcal{Z}|}(L^{i-1} \parallel D \parallel P) & i = \lambda, \end{cases}$$

where f is the convolution function from Eq. (5.3), G is the tensor from Eq. (5.2), and $\parallel$ is the tensor concatenation operator along the third dimension, and $\text{relu}(x) = \max(0, x)$ is the linear rectifier activation function. Here, κ and λ determine the breadth and depth of the architecture. A higher λ corresponds to a larger receptive field when making final predictions. For example at $\lambda = 2$, the decision at some cell is informed by its immediate neighbors with a receptive field of 3×3. However, at $\lambda = 3$, decisions are informed by all adjacent neighbors in a 5×5 window. The last layer, L^λ, is the output layer immediately prior to application of the *softmax* function. Given the architecture in Figure 5.3 with two convolutional layers, the convolve-and-pool operation is applied twice, indicated as steps ⑤ and ⑦ in the figure.

Softmax Output Layer

Given L^λ, we apply the *softmax* function along the third dimension to obtain a categorical distribution tensor $Q \in \mathbb{R}^{n \times n \times |\mathcal{Z}|}$over output tags $\mathcal{Z}$ for each word position pair such that $Q_{i,j,k} = \exp(L^\lambda_{i,j,k}) / (\sum_{l=1}^{|\mathcal{Z}|} \exp(L^\lambda_{i,j,l}))$, where $Q_{i,j,k}$ is the probability estimate of the pair of words at position i and j being assigned the kth tag. This constitutes the final step ⑧ of the network (Figure 5.3). Suppose $Y \in \mathbb{R}^{n \times n \times |\mathcal{Z}|}$ represents the corresponding one-hot encoded ground truth along the third dimension such that $Y_{i,j,k} \in \{0, 1\}$. Then the example-based loss ℓ is obtained by summing the categorical cross-entropy loss over each cell in the table, normalized by the number of words in the sentence; that is,

$$\ell(Y, Q; \theta) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{|\mathcal{Z}|} Y_{i,j,k} \log(Q_{i,j,k}), \tag{5.4}$$

where θ is the network parameter set. During training, the loss ℓ is computed per example and averaged along the mini-batch dimension.

5.2.3 Decoding the Output

While we learn concrete tags during training, the process for extracting predictions is slightly more nuanced. Entity spans are straightforwardly extracted by decoding BILOU tags along the diagonal. However, RE is based on “ensembling” the cellular outputs of the table where entity spans intersect. For entities a and b represented by their starting and ending offsets, $(a_{\mathrm{S}}, a_{\mathrm{E}})$ and $(b_{\mathrm{S}}, b_{\mathrm{E}})$, the relation between them is the label computed as $\operatorname{argmax}_{1 \le k \le |\mathcal{Z}|} \sum_{i=a_{\mathrm{S}}}^{a_{\mathrm{E}}} \sum_{j=b_{\mathrm{S}}}^{b_{\mathrm{E}}} Q_{i,j,k}$, which indexes a tag in the label space $\mathcal{Z}$.

5.3 Experimental Setup

In this section, we describe the established evaluation method, the datasets used for training and testing, and the configuration of our model. We note that the computing hardware is controlled across experiments given we report training and testing run times. Specifically, we used the Amazon AWS EC2 `p2.xlarge` instance which supports the NVIDIA Tesla K80 GPU with 12 GB memory.

5.3.1 Evaluation Metrics

We use the well-known F1 measure (along with precision and recall) to evaluate NER and RE subtasks as in prior work. For NER, a predicted entity is treated as a *true positive* if it is exactly matched to an entity in the groundtruth based on both character offsets and entity type. For RE, a predicted relation is treated as a *true positive* if it is exactly matched to a relation in the ground truth based on subject/object entities and relation type. As relation extraction performance directly subsumes NER performance, we focus purely on relation extraction performance as the primary evaluation metric of this study.

5.3.2 Datasets

CoNLL04 We use the dataset originally released by Roth and Yih [2] with 1441 examples consisting of news articles from outlets such as WSJ and AP. The dataset has four entity types including *Person*, *Location*, Organization, and *Other* and five relation types including *Live_In*, *Located_In*, *OrgBased_In*, *Work_For*, and *Kill*. We report results based on training/testing on the same train-test split as established by Gupta et al. [84], Adel and Schütze [89], Bekoulis et al. [45, 48], which consists of 910 training, 243 development, and 288 testing instances.

ADE We also validate our method on the Adverse Drug Events (ADE) dataset from Gurulingappa et al. [86] for extracting drug-related adverse effects from medical text. Here, the only entity types are *Drug* and *Disease* and the relation extraction task is strictly binary (i.e., Yes/No w.r.t the ADE relation). The examples come from 1644 PubMed abstracts and are divided in two partitions: the first partition of 6821 sentences contain at least one drug/disease pair while the second partition of 16695 sentences contain no drug/disease pairs. As with prior work [90, 5, 45, 48], we only use examples from the first partition from which 120 relations with nested entity annotations (such as "lithium intoxication" where *lithium* and *lithium intoxication* are the drug/disease pair) are removed. Since sentences are duplicated for each pair of drug/disease mention in the original dataset, when collapsed on *unique* sentences, the final dataset used in our experiments constitutes 4271 sentences in total. Given there are no official train-test splits, we report results based on 10-fold cross-validation, where results are based on averaging performance across the ten folds, as in prior work.

5.3.3 Model Configuration

Table 5.1: Model configuration as tuned on the CoNLL04 development set.

Setting	**Value**	**Setting**	**Value**
Optimization Method	RMSProp	Character Embedding Size (π)	25
Learning Rate	0.005	Character Representation Size (η)	50
Dropout Rate	0.5	Position Embedding Size (γ)	25
Num. Epochs	100	Dependency Embedding Size (β)	10
Num. Channels (κ)	15	Word Embedding Size (δ)	200
Num. Layers (λ)	8	Context Embedding Size (ρ)	200

We tuned our model on the CoNLL04 development set; the corresponding configuration of our model (including hyperparameter values) used in our main experiments is shown in Table 5.1. For the ADE dataset, we used Word2Vec embeddings pretrained on the corpus of PubMed abstracts [58]. For the CoNLL04 dataset, we used GloVe embeddings pretrained on Wikipedia and Gigaword [91]. All other variables are initialized using values drawn from a normal distribution with a mean of 0 and standard deviation of 0.1 and further tuned during training. Words were tokenized on both spaces and punctuations; punctuation tokens were kept as is common practice for NER systems. For part-of-speech and dependency parsing, we use the well-known tool spaCy[1]. For both datasets, we used projective dependency parses produced from the default pretrained English models. We found that using models pretrained on biomedical text (namely, the GENIA [81] corpus) did not improve performance on the ADE dataset.

---

[1] https://spacy.io/

Early experiments showed that applying exponential decay to the learning rate in conjunction with batch normalization [92] is essential for stable/effective learning for this particular architecture. We apply exponential decay to the learning rate such that it is roughly halved every 10 epochs; concretely, $r_k = r_b{}^{\frac{k}{10}}$ where r_b is the base learning rate and r_k is the rate at the kth epoch. We apply dropout [93] on h_i for $i = 1, \ldots, n$ as regularization at the earlier layers. However, dropout had a detrimental impact when applied to later layers. We instead apply batch normalization as a form of regularization on representations G and L^i for $i = 1, \ldots, \lambda - 1$. We optimize the objective loss using RMSProp [61] with a relatively high initial learning rate of 0.005 given exponential decay is used.

5.4 Results and Discussion

Table 5.2: Results comparing to other methods on the CoNLL04 dataset. We report 95% confidence intervals around the mean F1 over 30 runs for models in the last two rows. Our model was tuned on the CoNLL04 development set corresponding to the configuration from Table 5.1.

	Entity Recognition			Relation Extraction			Avg. Epoch	Avg.
Model	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	Train Time	Test Time ⁎
Table Representation[3]	81.20	80.20	80.70	76.00	50.90	61.00	-	-
Multihead [45]	83.75	84.06	83.90	63.75	60.43	62.04	-	-
Multihead with **AT** [48]	-	-	83.61	-	-	61.95	-	-
Replicating Multihead with **AT** [48]†	84.36	85.80	**85.07** ± 0.26	65.81	57.59	61.38 ± 0.50	614 sec	34 sec
Relation-Metric (Ours)†	84.46	84.67	84.57 ± 0.29	67.97	58.18	**62.68** ± 0.46	**101 sec**	**4.5 sec**

† These results are directly comparable given the same train-test splits, pretrained word embeddings, and computing hardware.

⁎ Average test time is per test set of 288 examples; dependency parsing accounts for approximately 0.5 second of our reported test time.

We report our main results in Tables 5.2 and 5.3 for the CoNLL04 and ADE datasets respectively. As a baseline, we replicate the prior best models [48] for both datasets based on publicly available source code[2]. Unlike prior work, which reports performance based on a single run, we report the 95% confidence interval around the mean F1 based on 30 runs with differing seed values for the CoNLL04 dataset. For the ADE dataset, we instead report the mean performance over 10-fold cross-validation so that results are comparable to established work. These experiments were performed

---

[2] https://github.com/bekou/multihead_joint_entity_relation_extraction

Table 5.3: Results comparing to other methods on the ADE dataset. We report the mean performance over 10-fold cross-validation for models in the last two rows. Our model was tuned on the CoNLL04 development set corresponding to the configuration from Table 5.1.

	Entity Recognition			Relation Extraction			Avg. Epoch	Avg.
Model	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	Train Time	Test Time ∗
Neural Joint Model [90]	79.50	79.60	79.50	64.00	62.90	63.40	-	-
Neural Joint Model [5]	82.70	86.70	84.60	67.50	75.80	71.40	-	-
Multihead [45]	84.72	88.16	86.40	72.10	77.24	74.58	-	-
Multihead with **AT** [48]	-	-	86.73	-	-	75.52	-	-
Replicating Multihead with **AT** [48]†	85.76	88.17	86.95	74.43	78.45	76.36	1567 sec	40 sec
Relation-Metric (Ours)†	86.16	88.08	**87.11**	77.36	77.25	**77.29**	**134 sec**	**4.5 sec**

† These results are directly comparable given the same fixed 10-fold splits, pretrained word embeddings, and computing hardware.

∗ Average test time is per test set of 427 examples; dependency parsing accounts for approximately 0.5 second of our reported test time.

using the same splits, pretrained embeddings, and computing hardware; hence, results are directly comparable.

We make the following observations based on our results from Table 5.2. Both our model and the model from Bekoulis et al. [48] tend to skew heavily towards precision. However, our method improves on both precision and recall, and by over 1% F1 on relation extraction where improvements are statistically significant ($p < 0.05$) based on the two-tailed Student's t-test. We note that our model performs slightly worse when evaluated *purely* on NER. We contend this is a worthwhile trade-off given our model is tuned purely on relation extraction and the relation extraction metric, being end-to-end, indirectly accounts for NER performance. Based on Table 5.3, when tested on the ADE dataset, our method improves over prior best results by approximately 1% F1 for RE on average. While the prior best skews toward recall in this case, our method exhibits better balance of precision and recall. Based on run time results, we contend that our method is more computationally efficient given training and testing times are nearly seven times lower on the CoNLL04 and ten times lower on the ADE set when compared to prior efforts. We note that dependency parsing accounts approximately one-half second of our testing time. While training time may not be crucial in most settings, we argue that fast and efficient predictions are important for many end-user applications.

As an auxiliary experiment, we tested the potential for integrating adversarial training (AT) with our model; however, there were no performance gains even with extensive tuning. On the CoNLL04 dataset, our method with AT performs at 62.26% F1, compared to 62.68% without AT. On the ADE dataset, our method performs at 76.83% F1 with AT, compared to 77.29% without AT. Given this, we have elected not to include AT evaluations as part of our main results.

Comparison with More Prior Efforts Gupta et al. [84], Adel and Schütze [89], and Zhang et al. [85] also experimented with the CoNLL04 dataset; however, Gupta et al. [84] evaluate on a more relaxed evaluation metric for matching entity bounds while Adel and Schütze [89] assume entity bounds are known at test time thus treating the NER aspect as a simpler entity *classification* problem. Of the three studies, results from Zhang et al. [85] are most comparable given they consider entity bounds in their evaluations; however, their results are based on a *random* 80%–20% split of the train and test set. As we use established splits based on prior work, the two results are not directly comparable.

5.4.1 Ablation Analysis

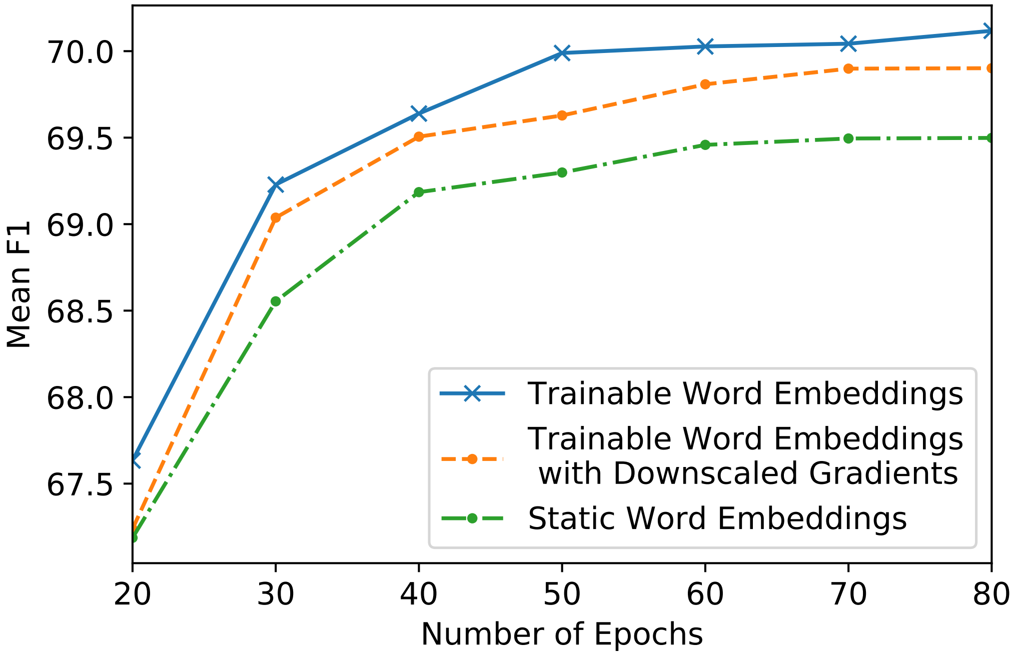
We report ablation analysis results in Table 5.4 using our best model as the baseline. We note that the model hyperparameters were tuned on the CoNLL04 development set. Character and dependency based features all had a notable impact on performance for either dataset. On the hand, while position embeddings had a positive effect on the ADE dataset, performance gains were negligible when testing on CoNLL04. For the CoNLL04 dataset, we find that character based features had little effect on precision while improving recall substantially.

Unsurprisingly, pretrained word embeddings had the greatest impact on performance in terms of both precision and recall. Early experiments showed that, unlike

Table 5.4: Ablation studies for relation extraction over the CoNLL04 and ADE dataset; each row after the first indicates removal of a particular feature/component.

	CoNLL04 (Relation)			ADE (Relation)		
Model	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)
Full model	67.97	58.18	**62.68**	77.36	77.25	**77.29**
– Character-based Input	67.30	52.69	59.09	76.73	76.44	76.58
– Dependency Embeddings	66.56	57.69	61.78	75.79	77.16	76.45
– Position Embeddings	68.57	57.34	62.43	75.94	76.62	76.27
– Pretrained Word Embeddings	62.33	46.09	52.96	72.50	71.41	71.91

Figure 5.5: Mean F1-score (over 10 runs) on CoNLL04 development set with respect to number of training epochs for various embedding training strategies.



models from prior work that used static word embeddings [5, 48], our model benefits from trainable word embeddings as shown in Figure 5.5. Here, trainable word embeddings with *downscaled* gradients refer to reducing the gradient of word embeddings by a factor of 10 at each training step.

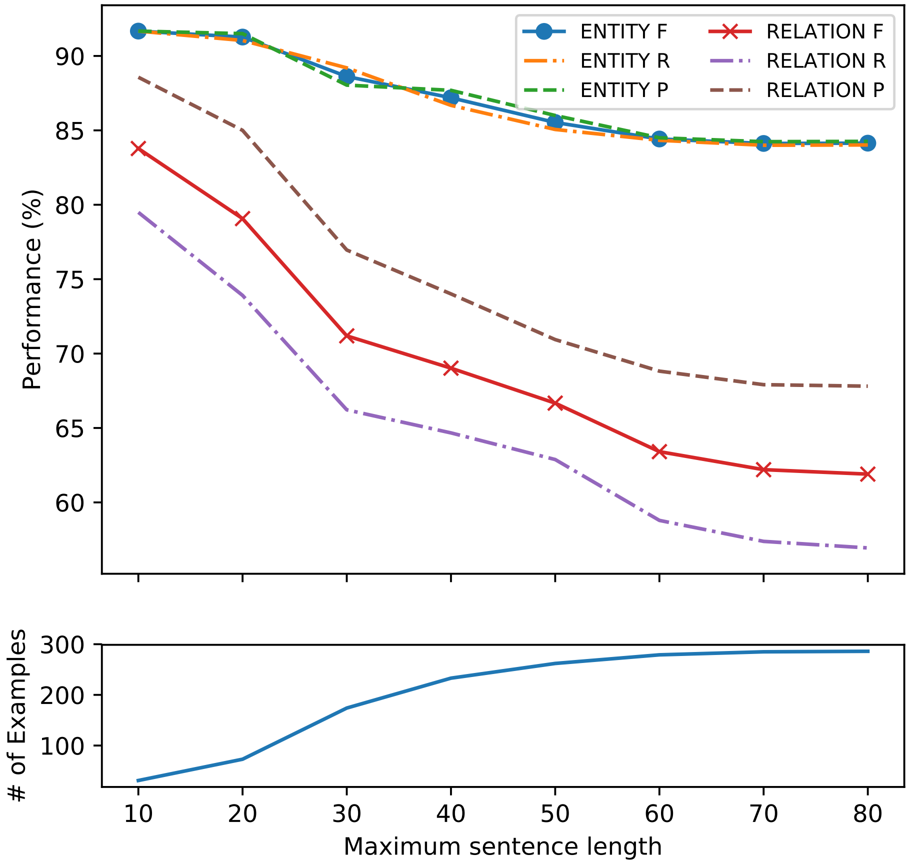
5.4.2 Error Analyses

In this section, we first perform a class based analysis where performance variations for different classes of examples are examined. Then, a more in-depth error analysis is performed for interesting example cases. The class based analyses entail partitioning examples by length, entity distance, and relation type and are covered in Section 5.4.2. The more in-depth example based analysis is discussed in Section 5.4.2.

Class based analyses

Long sentences are a natural source of difficulty for relation extraction models given the potential for long-term dependencies. In this section, we perform straightforward analysis by conducting experiments to assess model performance with respect to increasing sentence length. For this experiment, we train a single model using 80% of the dataset with 20% held out for testing. For some sentence length limit $\hat{k}$, we

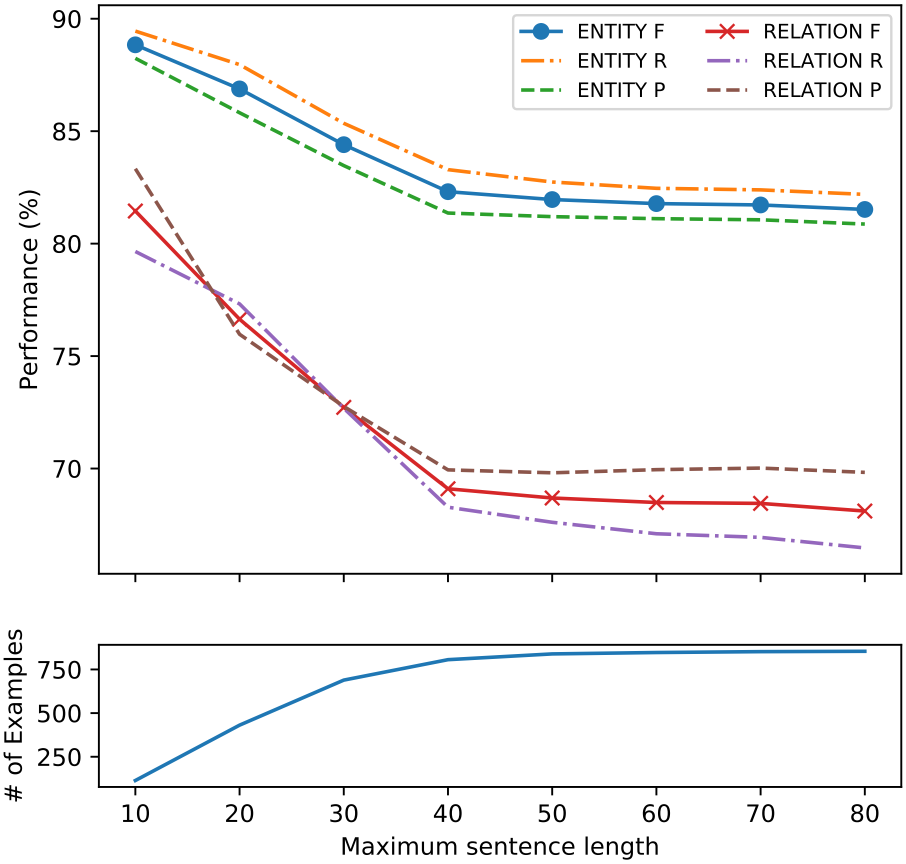
Figure 5.6: Entity and relation extraction performance with respective to change in maximum sentence length for CoNLL04.



evaluate on a subset of the overall test set that includes only examples with a sentence length that is less than or equal to $\hat{k}$.

Results from these experiments are plotted in Figures 5.6 and 5.7, for the CoNLL04 and ADE datasets respectively, such that $\hat{k}$ is varied along the horizontal x-axis. The top graph displays performance, while the bottom graph plots the number of examples with sentence length less than or equal to $\hat{k}$ that are used for evaluation. As shown, performances for both NER and RE tend to decline as longer sentences are added to the evaluation set. Unsurprisingly, relation extraction is more susceptible to long sentences compared to entity recognition. While there is a decline in both relation extraction precision and recall, we note that recall drops at a faster rate with respect to maximum sentence length and this phenomenon is apparent for both datasets.

Figure 5.7: Entity and relation extraction performance with respect to change in maximum sentence length for ADE.



In addition to length-based analysis, we also conducted experiments to study the variation in relation extraction performance with respect to the distance between subject and object entities as shown in Table 5.5. We measure distance by computing the absolute character offset between the last character of the first occurring entity and first character of the second occurring entity, which is henceforth simply referred to as "entity distance." Our results show that, at least on the CoNLL04 dataset, notable performance differences occur at the boundary cases; i.e., very short range relations (0-20 entity distance) tend to be easier and very long range relations (80-100 entity distance) tend to be harder (mostly due to changes in recall). For the ADE dataset, performance is similar across all partitions of entity distances. This is surprising, as sentence length appears to have a more notable impact on relation extraction performance than entity distance for this particular architecture.

Table 5.6 shows variance in performance when examined by relation type. Here, we see that performance depends heavily on the type of relation being extracted; our model exhibits much higher accuracy on the *Kill* relation at 80% F1, with *Located_In* and *Work_For* being the most difficult with performance below 60% F1. These results further corroborate our analysis based on Table 5.5 that entity distance does not correlate with example difficulty given that the *Kill* relation, being the easiest relation to extract, occurs with the highest average entity distance.

Example based analysis

A common source of difficulty that occurs is ambiguity with respect to expression of the *Live_In* and *Work_In* relation types. For example, consider the sentence “After buying the shawl for $1,600, Darryl Breniser of Blue Ball, said the approximately 2-by-5 foot shawl was worth the money.” The ground truth relation is (Darryl Breniser, *Live_In*, Blue Ball) which indicates that “Blue Ball” is in fact a location. However, it is difficult to assess whether “Blue Ball” is a location or company based on the context alone and without broader geographical knowledge (even for humans). Our model predicted (Darryl Breniser, *Work_For*, Blue Ball) in this case. We observe a similar pattern in the following case: “Santa Monica artist Tom Van Sant said Monday after the 23-foot-tall statue was found crushed and broken in pieces.”; here, we see the same phenomenon where our model mistakes (Tom Van Sant, *Live_In*, Santa Monica) for (Tom Van Sant, *Work_For*, Santa Monica). Finally, we present the most interesting example of this type of ambiguity in the sentence: “‘Temperatures didn’t get too low, but the wind chill was bad’, said Bingham County Sheriff’s Lt. Bill Gordon.” Here, the ground truth indicates that the only relation to be extracted is (Bill Gordon,

Table 5.5: Relation extraction performance partitioned based on “Entity Distance”, which is defined as the *number of characters* separating the subject and object entities (i.e., absolute character offset).

	CoNLL04 (Relation)				ADE (Relation)			
Entity Distance	# of Examples	**P** (%)	**R** (%)	**F** (%)	# of Examples	**P** (%)	**R** (%)	**F** (%)
0 — 20	207	83.7	43.80	57.51	447	88.50	42.02	56.98
20 — 40	51	59.09	24.07	34.21	265	77.17	35.51	48.64
40 — 60	43	80.00	18.60	30.19	181	78.72	37.00	50.34
60 — 80	22	100.00	25.93	41.18	125	82.35	29.58	43.52
80 — 100	13	100.00	15.38	26.67	91	85.00	34.00	48.57

Table 5.6: Relation extraction performance on the CoNLL04 dataset partitioned based on relation type.

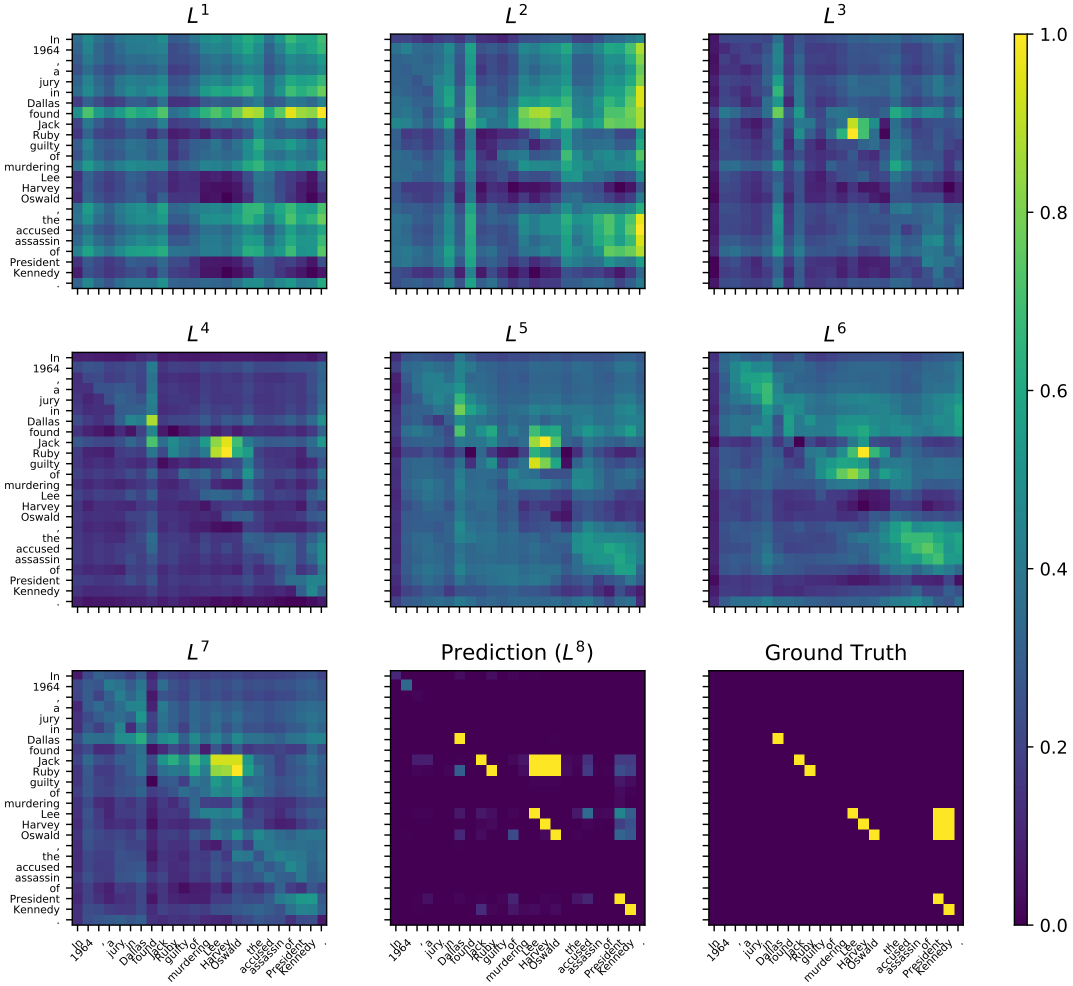
			CoNLL04 (Relation)		
Relation Type	# of Examples	Avg. Entity Distance	**P** (%)	**R** (%)	**F** (%)
Kill	46	47	81.25	82.98	82.11
Live_In	82	37	71.76	61.00	65.95
Located_In	58	28	80.77	44.68	57.53
Work_For	65	24	60.56	56.58	58.50
OrgBased_In	70	29	91.38	50.48	65.03

Live_In, Bingham County); however, our model extracts (Bill Gordon, *Work_For*, Bingham County Sheriff), which is also technically a valid relation. Such cases present ambiguities that are also difficult for human annotators; here, imbuing the NER component with external knowledge or learning based on a broader level of context may alleviate these types of errors.

Inconsistencies in the way entities are annotated can also cause issues when it comes to demarcating names that are accompanied with honorifics or titles. For example, some ground truth annotations will include the title, such as "President Park Chung-hee" or "Sen. Bob Dole", and other cases will leave out the title, such as "Kennedy" instead of "President Kennedy." These truth annotations are inconsistent and present a source of difficulty for the model during training and testing. For example, "Navy spokeswoman Lt. Nettie Johnson was unable to say immediately whether the aircraft had experienced problems from faulty check and drain valves." Here, our model extracted (Lt. Nettie Johnson, *Work_For*, Navy), while the groundtruth is (Nettie Johnson, *Work_For*, Navy) — while both are technically correctly, the extremely precise nature of the evaluation metric causes this prediction to be considered a false positive.

We also see such issues with annotation at the relation extraction stage; for example, consider the sentence "In 1964, a jury in Dallas found Jack Ruby guilty of murdering Lee Harvey Oswald, the accused assassin of President Kennedy." Figure 5.8 shows the internal activity of the network as it attempts to extract entities and relations from this particular example. Here, the ground truth annotation includes (Lee Harvey Oswald, *Kills*, President Kennedy), which our model fails to recognize; we instead obtain the prediction (Jack Ruby, *Kills*, Lee Harvey Oswald) which is a valid relation missed by the ground truth. In fact, it can be argued that the latter relation is a stronger manifested of the "Kill" relation based on the linguistic context

Figure 5.8: Visualization of activity of pooling layers at various depths (L^i for $i = 1, \ldots, \lambda$), as tabular heatmaps, for a network with a depth of $\lambda = 8$ given the following input sentence: “In 1964, a jury in Dallas found Jack Ruby guilty of murdering Lee Harvey Oswald, the accused assassin of President Kennedy.” Here, we measure activity by sum-pooling the activations along the channel dimension of each hidden representation. For the prediction activity, we simply max-pool probabilities along the relation dimension thus ignoring the exact *type* of entity or relation.



as evidenced by the trigger phrase “found [..] guilty of murdering”. We note that our model is able to detect (Lee Harvey Oswald, *Kills*, President Kennedy) as shown in the center-bottom heatmap of Figure 5.8; however, signals were not strong enough to warrant a concrete extraction of the relation.

In the ADE dataset, we mostly observe issues with entity recognition where boundaries of noun phrases are not properly recognized. Modifier phrases are sometimes not predicted as part of the named entity, for example: “protracted neuromuscular block" instead extracted as "neuromuscular block", and "generalized mite infestation" instead extracted as simply "mite infestation." The nature of the data results in especially long named entities that are often entire noun or verb phrases which can be difficult to delimit. For example, consider the following case: "DISCUSSION: Central nervous system (CNS) toxicity has been described with ifosfamide, with most cases reported in the pediatric population." Here, instead of extracting (Central nervous system (CNS) toxicity, ifosfamide) as the relation pair, our model predicts (Central nervous system, ifosfamide) and (CNS, ifosfamide). Essentially, long entity phrases are often not recognized in their entirety, and broken down into segments where each segment is independently involved in a relation. In this particular case, this error in prediction lead to one false negative and two false positives. This phenomenon occurs frequently with coordinated noun phrases which present a nontrivial challenge. For example, "<u>Growth and adrenal suppression</u> in asthmatic children treated with high-dose fluticasone propionate." is annotated with "Growth and adrenal suppression" as a singular entity, while our model falsely recognizes it as two entities "Growth" and "adrenal suppression." We see similar outcomes for the sentence: "<u>Generalized maculopapular and papular purpuric eruptions</u> are perhaps the most common thionamide-induced reactions." Such cases occur frequently which we suspect are a major source of hampered precision given the increased number of false positives for each predictive mistake.

5.5 Conclusion

In this study, we introduced a novel neural architecture that combines the ideas of metric learning and convolutional neural networks to tackle the highly challenging problem of end-to-end relation extraction. Our method is able to simultaneously and efficiently recognize entity boundaries, the type of each entity, and the relationships among them. It achieves this by learning intermediate table representations by pooling local metric, dependency, and position information via repeated application of the 2D convolution. For end-to-end relation extraction, this approach improves over the state-of-the-art across two datasets from different domains with statistically significant results based on examining average performance of repeated runs. Moreover, the proposed architecture operates at substantially reduced training and testing times with testing times that are seven to ten times faster, the latter important for many user-end applications. We also perform extensive error analysis and show that our network can be visually analyzed by observing the hidden pooling activity leading to preliminary or intermediate decisions. Currently, the architecture is designed for extracting relations involving two entities and occur within sentence bounds; handling n-ary relations and exploring document-level extraction involving cross-sentence relations will be the focus of future work.

Chapter 6 End-to-End Extraction of Cross-Sentence N-ary Relations

Most studies on relation extraction are focused on intra-sentence binary relation extraction. However, real-world problems are typically more complex, and may involve several key challenges not adequately addressed by existing work. One of which is the *end-to-end* aspect, which we have discussed and addressed in the previous chapters of this dissertation. However, other interesting challenges include the extraction of *cross sentence* relations and the extraction of *N-ary* relations. The ability to extract relations of variable arity across multiple sentences is highly prized in some specialized domains, including the biomedical domain, and with problems related to precision medicine. Given genetic variants may impact targeted treatment outcomes, knowledge about relationships among drugs, genes, and mutations is crucial for personalizing patient care. Such data, when compiled and distilled, will lead to better clinical decisions; however, cutting-edge knowledge of this form is still latent in biomedical literature. Being able to extract *cross sentence N-ary* relations is important for the following reasons.

- Cross-sentence relation extraction is important, especially in the biomedical domain given documents tend to be especially long. Long-range relations that occur across sentences appear frequently and tend to be difficult to identify and extract as they are usually expressed in an implicit manner. Based on a preliminary analysis of the JAX-CKB dataset [94], with gold standard annotations among drug-gene-mutations, limiting efforts to extracting intra-sentence relations will impair recall by up to 32 absolute percentage points.

- N-ary relation extraction refers to problems where relations may involve a variable number of participating entities, as opposed to binary relation extraction, where the number of entities is fixed to two. Complex biomedical relationships exist, for example, wherein a drug is asserted as being effective for treating a disease arising from a particular genetic mutation in some gene. As an example, consider the following sentence: "The FDA-approved RAF inhibitor *vemurafenib* and *dabrafenib* have elicited responses and extended survival of patients with *BRAF V600E* melanomas." Here, the sentence expresses a relationship wherein the drugs *vemurafenib* and *dabrafenib* are used in treatment of a type of cancer caused by the mutation *V600E* occurring in the BRAF gene. Such relationships are complex and cannot be addressed by existing binary approaches. Based on our analysis of the JAX-CKB dataset, limiting efforts to only predicting binary relations, as is typical, may impair recall by up to 18 absolute percentage points.

Designing an architecture able to simultaneously handle cross-sentence relations and N-ary relations is a nontrivial task given scalability issues. In case of N-ary relations, the variability of N depends on the nature of the problem being studied. To illustrate the complexity of the problem, we use set notation to describe the class of arity for a particular problem; for example, we classify a problem using the set $\mathcal{K}$ such that, as an example, a problem with 2-ary and 3-ary relations has $\mathcal{K} = \{2,3\}$. Henceforth, we refer to such specific cases of the N-ary problem as $\mathcal{K}$-ary. Suppose $\mathcal{E}$ represents the set of entities for a hypothetical example. For a problem with $\mathcal{K} = \{2,3\}$, that is, $\{2,3\}$-ary relations, we must consider up to $|\mathcal{E}|^2 + |\mathcal{E}|^3$ candidate selections. Thus, in general, a problem with $\mathcal{K}$-ary relations must consider up to $\sum_{x \in \mathcal{K}} |\mathcal{E}|^x$ comparisons. The complexity of an $\mathcal{K}$-ary problem is defined by the maximum arity, or $\max(\mathcal{K})$, such that we observe a time complexity of $O(|\mathcal{E}|^{\max(\mathcal{K})})$, which is polynomial with respect to the number of entities and exponential with respect to the maximum arity of candidate relations. Given the number of entities tend to scale with input length (naturally, a paragraph will contain more entities than a sentence), longer sentences tend to exacerbate the number of comparisons needed.

Recent works moving beyond existing standards have typically focused on two of three aspects; for example, end-to-end and cross-sentence [95, 96] or cross-sentence and N-ary [97, 98, 99, 100]. To our knowledge, no studies have explored all three aspects in a single model. In this study, we propose the first end-to-end approach for extracting inter-sentence N-ary relations. We overcome issues of scale by appropriating the efficient neural-metric learning architecture introduced in Chapter 5 as a base for learning lower-arity representations. The representations of lower-arity relations are used to construct high-arity representations and subsequent high-arity relations. We demonstrate the effectiveness of our approach by validating our method on a dataset, called JAX-CKB [94], of full-text PubMed documents annotated with gold standard relations among drugs, genes, and genetic mutations. For this particular problem, we wish to extract these biomedical entities and the $\mathcal{K}$-ary relations among them, where $K = \{2,3\}$, treating entire paragraphs as units of discourse. Besides architectural innovations, we highlight the following contributions. We demonstrate the extent to which cross-sentence N-ary relation extraction improves over baseline models that are restricted to intra-sentence binary relation extraction. We show that learning to extract higher-arity relations has a side effect of improving the quality of binary relations extracted. Lastly, we introduce several variants of the architecture with different levels of restrictiveness when considering triple candidates, and show that ensembling them leads to improved overall results on all accounts.

6.1 Related Work

In this section, we discuss recent related work primarily dealing with *cross-sentence* and *N-ary* relations. For a review of work in *end-to-end* relation extraction, we refer readers to Section 2.2.

Cross-sentence relation extraction. Recent studies dealing primarily with *cross-sentence* relation extraction typically deal with problems in the biomedical domain such as chemical-disease [101] and protein-protein [102] interaction extraction from biomedical literature. For some problem domains, such as those involving chemical-protein interactions [103], there are few enough cross-sentence relations (< 1%) that they can be effectively ignored without adversely affecting recall [104]. Otherwise, cross-sentence relations are typically considered by simply changing the scope of each unit example from sentence-level to either paragraph-, abstract-, or document-level [95, 96]. To limit the number of candidates, some approaches involving a filtering step wherein only candidates within a certain distance — for example, within three sentences — are considered [96]. Other notable approaches include applying distant supervision to cross-sentence relation extraction [105, 106, 107], graph LSTMs [97, 98], or focusing on scalability by use of multi-scale, entity-centric representations [100]. Focusing on chemical-disease relations, Verga et al. [44] proposed a method for simultaneously predicting relations between all pairs of mentions; aggregating over mention pairs allowed for multi-instance learning for extracting document-level relations between entity pairs.

N-ary relation extraction. N-ary relation extraction has been explored in a variety of ways including rule-based methods on shortest dependency path [108], pattern discovery based on domain ontology [109], and linguistic theory involving Frame Semantics [110]. Machine learning based approaches include methods based on distant supervision [106], LSTMs over graph structures [97, 98], and learning high-arity relations based on lower arity representations [100, 111].

End-to-end, cross-sentence, and N-ary relation extraction. Existing works that focus on *end-to-end* and *cross sentence* relation extraction are mostly limited to studies on protein-protein interactions [102]. On this front, Tran and Kavuluru [95] devised an end-to-end pipeline method for cross-sentence binary relation extraction, considering all protein pairs mentioned within the abstract as potential interaction candidates. Zhou et al. [96] instead considered only protein pairs mentioned within a window of three sentences as potential candidates. The other set of studies focused primarily on *cross sentence* and *N-ary* relation extraction with deep neural network approaches. Approaches in this category included the use of graph-structured LSTMs [97, 98], hybrid LSTM-CNN network [99], distant supervision [106], and multi-scale entity-centric representation learning [100]. To our knowledge, no studies have approached the problem of relation extraction in an end-to-end, cross-sentence, and N-ary fashion.

6.2 Methodology

Our approach is based on the premise that an N-ary relation can be decomposed and represented as a set of constituent binary sub-relations. Intuitively, we can analyze the semantics of a ternary relation between entities `a`, `b`, and `c`, denoted by tuple `(a, b, c)`, by exploring binary sub-relations between entities represented by pairs `(a,b)`, `(a,b)`, and `(a,c)`. Inspired by prior work on N-ary relation extraction [100, 111], the goal is to learn ternary (or N-ary) relations by representing them as a composition of representations corresponding to constituent binary pairs. Thus, we approach the problem of end-to-end extraction of binary and ternary relations as follows. To extract entities and binary relations, we follow the steps as established in Section 5.2.2 of the previous chapter. To recap, we extract entities by decoding softmax signals along the diagonal and we extract binary relations by averaging the signals at intersections of the output. In Section 6.2.1, we describe how the architecture is extended to accommodate ternary relations unique to the target problem and dataset.

6.2.1 Neural Network Architecture

The neural-metric architecture readily and naturally represents a semantic relationship between pairs of words at positions i and j through the representation $\mathbf{g}^{i,j}$. For an example, we denote $\mathcal{E}$ as the set of all gold entities for a particular example. Concretely, each entity e is represented as a pair of indices indicating the starting and ending offsets, denoted as e^{end} and e^{start} respectively, such that $e = (e^{\text{start}}, e^{\text{end}})$. For a set of entities, we can derive a list of candidate relations by observing all entity combinations matching a relevant type template (more later). Let $\mathcal{R}_{(\mathcal{E})}$ represent the set of all candidate 2, 3-ary relations between entities in $\mathcal{E}$ expressed in x, where $\mathcal{R}^2_{(\mathcal{E})} \subset \mathcal{R}_{(\mathcal{E})}$ represents the set of candidate binary relations and $\mathcal{R}^3_{(\mathcal{E})} \subset \mathcal{R}_{(\mathcal{E})}$ represents the set of candidate ternary relations. We compute $\mathcal{R}^2_{(\mathcal{E})}$ by including all pairs of entities in $\mathcal{E}$ adhering to one of the following type templates: `(Drug,Gene)`, `(Gene,Variant)`, and `(Drug,Variant)`. Likewise, we compute $\mathcal{R}^3_{(\mathcal{E})}$ by including all triples of entities in $\mathcal{E}$ adhering to the type template `(Drug, Gene, Variant)`.

The proposed method builds on the neural metric learning architecture discussed in the previous chapter. We again denote the relation-metric representation first defined in Equation 5.2 as $G \in \mathbb{R}^{n \times n \times \kappa}$ representing an $n \times n$ sentence wherein the third dimension embeds metric-based features. We use $\mathbf{g}^{i,j}$ to denote the embedding vector of length κ for position i, j, such that $\mathbf{g}^{i,j}_k = G_{i,j,k}$ for $k = 1 \dots \kappa$. For a candidate ternary relation $(a, b, c) \in \mathcal{R}^3_{(\mathcal{E})}$, the probability of there being a relationship between (a, b, c) is expressed as

$$\bar{p} = \sigma(\, W^{\text{DrugGene}} f(a,b) + W^{\text{GeneVariant}} f(b,c) + W^{\text{DrugVariant}} f(a,c) + \mathbf{b}^{\text{DGV}}\,)\,,$$

where $f(r, s)$ is a vector representation of the relationship between entities r and s, defined as

$$f(r,s) = \mathbf{g}^{r^{\text{start}}, s^{\text{start}}} \parallel \mathbf{g}^{r^{\text{end}}, s^{\text{end}}},$$

where $\parallel$ is the vector concatenation operator; W^{DrugGene}, $W^{\text{GeneVariant}}$, $W^{\text{DrugVariant}}$, and $\mathbf{b}^{\text{DGV}}$ are network parameters; and σ is the sigmoid function. The binary cross entropy loss for the candidate triple is computed as

$$\bar{\ell}(\bar{y}, \bar{p} | \theta) = -\left(\bar{y} \log(\bar{p}) + (1 - \bar{y}) \log(1 - \bar{p})\right)$$

where $\bar{y} \in \{0, 1\}$ is the groundtruth and θ is the set of all parameters.

6.2.2 Ternary Relation Extraction

Let $\mathcal{E}'$ represent the set of entities *predicted* for a particular test example. For conciseness, we use $\mathcal{R}^2_{(\mathcal{E}')}$ to denote the set of *candidate* binary relations and $\mathcal{R}^{2\star}_{(\mathcal{E}')}$ to denote the set of *predicted* binary relations such that $\mathcal{R}^{2\star}_{(\mathcal{E}')} \subset \mathcal{R}^2_{(\mathcal{E}')}$. During extraction time, we first extract the set of predicted entities $\mathcal{E}'$ according to procedures in Section 5.2.3. Based on those same procedures, we can extract a set of predicted binary relations $\mathcal{R}^{2\star}_{(\mathcal{E}')}$ directly without considering $\mathcal{R}^2_{(\mathcal{E}')}$. However, $\mathcal{R}^{2\star}_{(\mathcal{E}')}$ can be used as a filtering mechanism for pre-emptively eliminating unlikely ternary candidates from $\mathcal{R}^3_{(\mathcal{E}')}$, thus resulting in fewer comparisons and more focused triple predictions. We propose three natural approaches for filtering:

- For **none**, no filtering is applied and we consider all candidates in $\mathcal{R}^3_{(\mathcal{E}')}$.
- For **relaxed**, we only consider a candidate triple if *at least one* of its constituent binary subrelations exist in the set of predicted binary relations. Concretely, we only consider a triple $(a, b, c) \in \mathcal{R}^3_{(\mathcal{E}')}$ if

$$(a, b) \in \mathcal{R}^{2\star}_{(\mathcal{E}')} \vee (b, c) \in \mathcal{R}^{2\star}_{(\mathcal{E}')} \vee (a, c) \in \mathcal{R}^{2\star}_{(\mathcal{E}')} .$$

- For **strict**, we only consider a candidate triple if *all* of its constituent binary subrelations exist in the set of predicted binary relations. Concretely, we only consider a triple $(a, b, c) \in \mathcal{R}^3_{(\mathcal{E}')}$ if

$$(a, b) \in \mathcal{R}^{2\star}_{(\mathcal{E}')} \wedge (b, c) \in \mathcal{R}^{2\star}_{(\mathcal{E}')} \wedge (a, c) \in \mathcal{R}^{2\star}_{(\mathcal{E}')} .$$

After applying one of the three aforementioned filtering step on $\mathcal{R}^3_{(\mathcal{E}')}$, we process the remaining candidates using the neural network model to obtain $\mathcal{R}^{3\star}_{(\mathcal{E}')}$, with $\mathcal{R}^{2\star}_{(\mathcal{E}')} \cup \mathcal{R}^{3\star}_{(\mathcal{E}')}$ being the final predicted set of {2,3}-ary relations.

6.2.3 Training Procedure

We train the core objective loss ℓ, defined in Equation 5.4 of Section 5.2.2, and triple prediction objective loss $\bar{\ell}$, defined in Equation 6.2.1, in an interleaved fashion. That is, for each training iteration, we train a set of examples (that is, a minibatch) on the original task entity and binary relation extraction task (corresponding to loss ℓ). Then, we transform the set of examples in the minibatch to a set of triple candidate level examples, which are then used to train objective $\bar{\ell}$. We filter triple candidates used for training based on the chosen filtering mode; given this, with *strict* filtering, there is a strong bias toward predicting as positive any candidate meeting the initial filter criteria; in this sense, *strict*-based predictions are effectively based on constituent binary predictions.

6.2.4 Model Ensembling

Models that are complementary tend to benefit from ensembling. Thus, in addition to the experiments with the proposed model, we also experiment with a simple ensembling approach based on majority voting. We discuss our motivation for ensembling later in Section 6.4; herein, we simply describe our approach. For entity recognition, each unique entity (entity name and entity type) and relation (participating entities and relation type) is included in the final ensemble prediction set if it is predicted in the majority of models of the ensemble. For example, in a 3-model ensemble, if the unique entity pair (e.g., the drug cisplatin) is predicted in at least two of the three models, it is included in the final prediction set. Likewise, a unique relation (e.g., a drug-gene relation involving the drug cisplatin and the gene metallothionein) indicating participating entities and the relation type is included in the final prediction set only if it is predicted by at least two of the three models of the ensemble.

6.3 Experimental Setup

In this section, we describe materials relevant to the experiments conducted in this study, including the dataset used to validate our method, the method for evaluation, and the configuration of the neural network model.

6.3.1 JAX-CKB Dataset

We train and evaluate our model on data from the JAX Clinical Knowledge-base (JAX-CKB) [94] consisting of gold standard drug-gene-mutation relations manually-curated by the Jackson Laboratory (JAX). The final dataset used in our experiments contains 342 PubMed full-text documents partitioned into 240, 51, and 52 documents based on a random 70-15-15 split for training, development, and testing respectively. While we use JAX-CKB annotations, preprocessing and entity-linkage of the input is based on data made publicly available by Jia et al. [100]. Characteristics of the dataset are presented in Table 6.1. According to our analysis, each document is, on average, comprised of about 30 paragraphs and each paragraph is, on average, comprised of about 6 sentences.

6.3.2 Evaluation Method

As JAX-CKB contains annotations at the document level (i.e., unique genes regardless of offset) based on normalized entities, we evaluate our method in a similar manner to reflect the expected real-world use case. That is, we train and extract relations at the *mention-level* as is typical of end-to-end elation extraction systems; before evaluating, we collapse the mention-level relations by normalizing them to unique drug/gene/mutations and truncating positional information. As normalization is not a goal of this study, we assume perfect normalization if there is an exact match of a drug/gene/mutation mention being extracted based on a pre-computed mapping of mentions to unique names. Mentions extracted but cannot be mapped on exact matching are not normalized and thus evaluated as false positives for both entity and

Table 6.1: Characteristics of the JAX-CKB dataset

	Training	Development	Testing
Num. Documents	240	51	52
Num. Paragraphs	7,414	1,545	1,605
Num. Sentences	42,675	9,050	9,547
Num. Entities	89,883	18,151	18,643
↳Num. Entities (`Drug`)	24,296	5,165	4,286
↳Num. Entities (`Gene`)	55,178	11,037	12,257
↳Num. Entities (`Variant`)	10,409	1,949	2,100
Num. Binary Relations	125,788	26,858	16,384
↳Num. Binary Relations (`Drug-Gene`)	62,304	14,334	7,103
↳Num. Binary Relations (`Gene-Variant`)	41,172	7,449	7,178
↳Num. Binary Relations (`Drug-Variant`)	22,312	5,075	2,103
Num. Ternary Relations	162,422	40,972	14,031

relation extraction. We evaluate using the popular F1 metric, where the F1 score is micro-averaged across class types to account for distributional differences between classes.

6.3.3 Model Configuration

We used the exact settings as described in Section 5.1 as a starting point given their success in our prior work and preliminary experiments. We instead focused on optimize architecture-specific settings, such as number of layers λ and number of channels κ. Based on a grid search set over $\{3, 6, 9, 12\}$ for λ and $\{15, 30, 45, 60\}$ for κ, evaluating on the development set, we found that $\lambda = 3$ and $\kappa = 45$ was optimal for this particular problem. These settings suggest that, in contrast to the previous study in Chapter 5, a "shallow but wide" architecture is more suitable at least for some problems. Given the biomedical nature of the data, we used Word2Vec embeddings pretrained on the corpus of PubMed abstracts. We again used spaCy to produce projective dependency parses at the sentence level. Since we focus on paragraph-level examples, we produce paragraph-level dependency graphs by joining the dependency parses of constituent sentences. We accomplish this by simply connecting the root node of a sentence to the root node of every other sentence in the same paragraph.

6.4 Results and Discussion

Table 6.2: Results on the test set for various models when learning and predicting at the sentence level, and when also evaluating at the sentence level. The "N-ary Relations" column are results from evaluating on a test set containing both binary and ternary relations, while the "Only Binary" and "Only Ternary" columns evaluates on binary relations and ternary relations exclusively.

	Entities			N-ary Relations			Only Binary Relations			Only Ternary Relations		
Model	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	**P**(%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)
Binary	93.19	89.35	91.23	48.12	60.95	53.78	48.12	69.18	56.76	-	-	-
N-ary with *no* filtering	94.30	88.64	**91.39**	51.35	61.16	**55.83**	52.55	62.93	**57.27**	42.08	48.02	44.85
N-ary with *relaxed* filtering	94.09	84.68	89.14	47.38	66.87	55.46	48.24	68.80	56.71	40.43	52.54	**45.70**
N-ary with *strict* filtering	94.15	87.97	90.96	50.74	59.81	54.90	52.23	62.47	56.89	38.17	40.11	39.12

We begin by presenting initial results of models evaluating on sentence-level relations from the test set in Table 6.2. Specifically, these are results on the test set for various models when learning and predicting at the sentence level, and when also evaluating at the sentence level. Models are thus not penalized for failing to identify inter-sentence relations. Each row corresponds to a different model or model variant while each column correspond to a different evaluation criteria as follows. The "N-ary Relations" column are results from evaluating on a test set containing both binary and ternary relations, while the "Only Binary" and "Only Ternary" columns evaluates on binary relations and ternary relations exclusively. As an aside, we note that entity recognition performance is much higher than results from prior chapters – this is owed to the fact that entity recognition is simply an easier problem here, in that the vast majority of entities span a single token.

We note that the precision of entity recognition tends to increase while recall tends to decrease when moving from binary to N-ary modeling. For relation extraction, the N-ary model exhibits an improvement over the binary mode by up to two absolute F1 points regardless of filtering mode (53.78% vs. 55.83% F1). In general, all models tend to exhibit predictions that tend to skew toward higher recall and lower precision. This is likely an artifact of training on mention-level annotations derived from document-level annotations.

Among N-ary model variants, we mainly observe varying differences in terms of precision-recall trade-off. With no filtering, we observe the best balance of precision and recall, while with "relaxed" filtering, we observe a much greater bias toward recall than precision. Interestingly, "strict" filtering has a trade-off that is somewhere in between the other two filtering variants. Among model variants, N-ary with *no* filtering tends to exhibit the best performance on N-ary (55.83% F1) and exclusively binary (57.27% F1) relations specifically. The "relaxed" model exhibits the best performance on ternary relations (45.70% F1), while the "strict" model exhibits the worst performance on ternary relations (39.12% F1). We note that this model's improvement over the binary version when evaluating on *only* binary relations (owing to improved precision) is an unexpected outcome, and may indicate that learning to additionally identify ternary relations has the side-effect also improving the quality of binary relations extracted.

Table 6.3: Our main results on the test set for various models and various levels of discourse when evaluating on both intra- and inter-sentence relations (at the paragraph level). Results from Table 6.2, in which models learn and predict at the sentence level, are included after adjusting for inter-sentence relations — by penalizing recall based on missed inter-sentence relations — so that all displayed results are *directly comparable*.

		Entity Recognition			N-ary Relations			Only Binary Relations			Only Ternary Relations		
Model	Scope	**P** (%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)	**P**(%)	**R** (%)	**F** (%)	**P** (%)	**R** (%)	**F** (%)
Binary	Sentence	93.19	89.35	91.23	48.14	41.72	44.70	48.14	50.39	49.24	-	-	-
N-ary with *no* filtering	Sentence	94.30	88.64	**91.39**	51.38	41.86	46.13	52.58	45.83	48.98	42.08	22.73	29.51
N-ary with *relaxed* filtering	Sentence	94.09	84.68	89.14	47.40	45.72	46.55	48.26	50.06	49.14	40.43	24.87	30.79
N-ary with *strict* filtering	Sentence	94.15	87.97	90.96	50.74	40.94	45.32	52.23	45.50	48.63	38.17	18.98	25.36
N-ary with ensembling⋆	Sentence	89.53	92.01	90.75	45.05	53.40	48.87	45.83	58.06	51.23	39.06	31.02	34.58
Binary	Paragraph	90.98	90.84	90.91	42.60	61.59	50.37	42.60	74.39	54.18	-	-	-
N-ary with *no* filtering	Paragraph	92.41	84.84	88.46	46.32	64.54	53.93	48.32	67.00	56.15	36.96	52.67	43.44
N-ary with *relaxed* filtering	Paragraph	91.43	90.15	90.78	43.31	67.11	52.64	46.31	69.11	55.46	31.48	57.49	40.68
N-ary with *strict* filtering	Paragraph	92.74	90.07	**91.39**	45.28	66.24	53.79	47.53	69.06	56.31	34.87	52.67	41.96
N-ary with ensembling⋆	Paragraph	93.85	88.96	91.34	47.77	66.10	**55.46**	50.31	68.50	**58.01**	36.62	54.55	**43.82**

⋆ Ensembling refers to an ensemble of three models from each variant: no filtering, relaxed filtering, and strict filtering.

While Table 6.2 focused on evaluating at the sentence level, we present our paragraph-level evaluations in Table 6.3. Results from Table 6.2 are included after adjusting to account for inter-sentence relations — where differences mainly manifest as penalties to recall — so that all results are directly comparable. As expected, recall drops up to 20 absolute percentage points (from sixties to forties) when we penalize the model for not identifying intersentence relations, which account for a significant portion of overall relations. When we train and test on examples the paragraph-level examples instead of sentence-level examples, precision tends to suffer (up to five percentage points). This is expected as input length presents a well-known source of difficulty, as previously demonstrated in Section 5.4.2. However, this is counterbalanced by a significant boost in recall, of up to 25 absolute percentage points, which results in an overall improvement to F1 across all models.

Once again, we observe a trend of improved precision on binary relation extraction performance between binary (lower) and N-ary models (higher), which reinforces our early intuition that learning higher arity relations impacts binary relation performance. At the paragraph level, the model with *strict* filtering exhibits the best performance across the board with an overall relation extraction F1 score of 53.79%. Our intuition based on preliminary results is that these variants are highly complementary, and we expect that ensembling may result in further improved performance. Thus, we included an additional evaluation based on an ensemble of these three models based on the method described in Section 6.2.4. As expected, the ensemble outperforms all other models on relation extraction, at both the sentence and the paragraph level with an F1 score of 55.46% on relation extraction at the paragraph level. Overall, results from Table 6.3 indicate that we can improve recall by at least 20 absolute percentage points (41.72% vs. 64.54%), while maintaining similar levels of precision (48.14% vs. 46.32%), by additionally accounting for N-ary and, more importantly, cross-sentence relations.

Type-based analysis. We present type-based evaluations of the model, corresponding to the last row in Table 6.3, for entity recognition in Table 6.4. The model exhibits extremely high precision and recall, each above 97%, in identifying genetic mutations. Overall, precision is relatively high (above 90%) across entity type with differences most notable in terms of recall. Genes are identified with a recall of 90%, while drugs are identified with an even lower recall of 80%. Thus, genes and drugs are areas of weakness of entity recognition for this model, and warrant further exploration in future work.

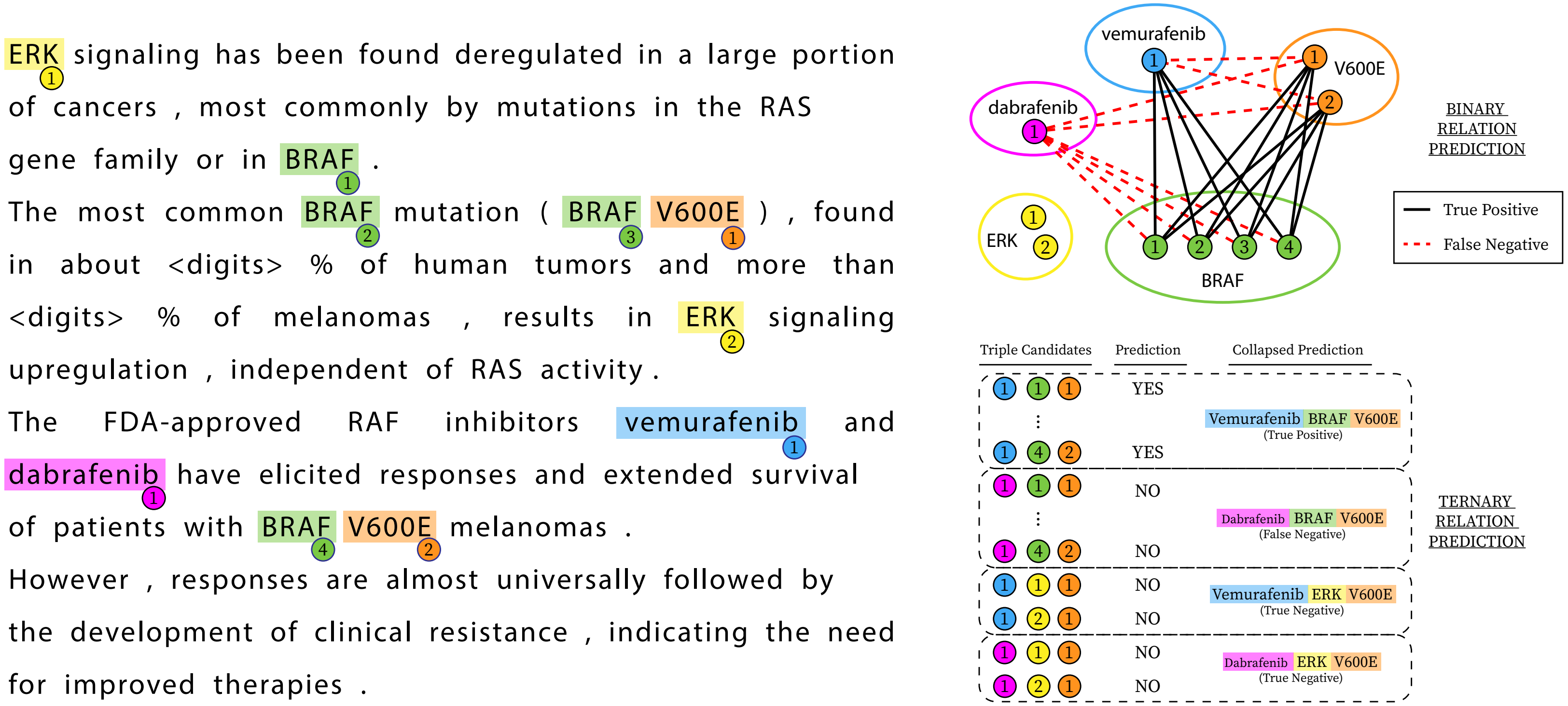
Table 6.4: Entity recognition results for the N-ary with ensembling model based on entity type.

Relation Type	**P** (%)	**R** (%)	**F** (%)
Drug	95.40	80.82	87.51
Gene	92.69	90.28	91.47
Mutation	**97.76**	**97.86**	**97.81**
Overall	93.85	88.96	91.34

Table 6.5: Relation extraction results for the N-ary with ensembling model based on relation type.

Relation Type	**P** (%)	**R** (%)	**F** (%)
Drug-Gene	50.20	65.48	56.83
Gene-Mutation	**54.44**	**77.98**	**64.12**
Drug-Mutation	42.94	58.38	49.48
Drug-Gene-Mutation	36.62	54.55	43.82
Overall	47.77	66.10	55.46

Figure 6.1: Error analysis on an example paragraph, with recall-related errors, appearing in the article with PMID 27523909. The entire input with model-annotated entities appear on the left side, where mentions of the same concept share the same color. On the top right, we present binary predictions and highlight missed binary relations. On the bottom right, we show predictions of ternary candidates and the final output evaluated based on ground truth information.



ERK ① signaling has been found deregulated in a large portion of cancers , most commonly by mutations in the RAS gene family or in BRAF ① .
The most common BRAF ② mutation (BRAF ③ V600E ①) , found in about <digits> % of human tumors and more than <digits> % of melanomas , results in ERK ② signaling upregulation , independent of RAS activity .
The FDA-approved RAF inhibitors vemurafenib ① and dabrafenib ① have elicited responses and extended survival of patients with BRAF ④ V600E ② melanomas .
However , responses are almost universally followed by the development of clinical resistance , indicating the need for improved therapies .

Triple Candidates	Prediction	Collapsed Prediction
① ① ①	YES	Vemurafenib BRAF V600E (True Positive)
⋮		
① ④ ②	YES	
① ① ①	NO	Dabrafenib BRAF V600E (False Negative)
⋮		
① ④ ②	NO	
① ① ①	NO	Vemurafenib ERK V600E (True Negative)
① ② ①	NO	
① ① ①	NO	Dabrafenib ERK V600E (True Negative)
① ② ①	NO	

We present similar evaluations for relation extraction, in Table 6.5, partitioned by relation type. Gene-mutation relations tend to be extracted at a higher accuracy than other types of relations, which aligns with our intuition given genes and genetic mutations are closely related concepts. Conversely, drug-mutation relation extraction performance is relatively worse, owing to both reduced precision and recall. Drug-gene-mutation relations expectedly exhibit the worst performance of all relation types, with the least precision overall.

Error analysis. For a more indepth analysis of potential issues, we present a false negative case in Figure 6.1. The example input is from a paragraph, of four sentences, appearing in the PubMed article identified by PMID 27523909. First, we note that the model exhibits perfect accuracy on entity recognition for this particular test example. This particular example is an interesting example given the third sentence strongly expresses a treatment relationship between the drugs *Vemurafenib* and *Dabrafenib* and the mutation *V600E* of *BRAF*. From analyzing the binary predictions, we notice two thematic issues. First, despite a clear therapeutic assertion involving the two drugs, only *Vemurafenib* is recognized as being involved in any binary or ternary relation. While surprising, this aligns with earlier observations in Section 5.4.2 that coordinating conjunctions, such as ‘and’ and ‘or’, are potential sources of confusion for this particular architecture. One simple way to alleviate this problem is to employ a rule-based syntactic preprocessing method that deconstructs such sentences into a set of multiple simpler sentences conveying the exact same idea. As an example, for the discussed sentence from Figure 6.1, the third sentence would be replaced with the following blurb.

> The FDA-approved RAF inhibitor *vemurafenib* have elicited responses and extended survival of patients with BRAF V600E melanomas. The FDA-approved RAF inhibitor *dabrafenib* have elicited responses and extended survival of patients with BRAF V600E melanomas.

Given the model is designed to detect intersentence relations, we suspect such an approach would resolve simple coordinated cases without any adverse effects. In terms of binary relations, we further observe that, despite detecting *vemurafenib* as being involved in a ternary relation with *BRAF* and *V600E*, the model has issues recognizing the simpler binary relationship between *vemurafenib* and *V600E*. This incongruity aligns with our type-based analysis, from Table 6.5, and suggests that optimizations specifically targeting these particular types of relations, potentially as rule-based postprocessing steps, will remedy at least some of the recall-related errors. As a caveat related to Figure 6.1, there may be quality assurance issues related to how data is annotated, and it is likely that the gene ERK is involved in a relation with other entities appearing in the article (as hinted here or elsewhere in the article). However, ground truth annotations indicate that ERK is not explicitly conveyed as being involved with other entiites. For consistency, we assume correctness of the ground truth when analyzing the presented example.

6.5 Conclusion

In this study, we proposed an approach for extracting inter-sentence N-ary relations corresponding to interactions between drugs, genes, and mutation, from the biomedical literature. We demonstrated that our approach improved over traditional intra-sentence binary relation extraction substantially through recall-focused improvements. While the focus of this study is on drug-gene-mutation relations of 2,3-arity, our method can be easily extended to handle any level of arity. In this study, we limited the scope of cross-sentence relations to discourse units at the paragraph-level; while much less frequent, it is possible for relations to be express across paragraphs. Thus, future work will focus on exploring ways of feasibly tackling document-level relation extraction.

Chapter 7 Conclusion

In this dissertation, we introduced several advanced deep learning approaches to tackling the problem of end-to-end relation extraction. We showed, via increasingly sophisticated neural network architectures, that modeling entity and relation extraction in a joint or coordinated manner, results in highly competitive models that are able to attain state-of-the-art results in many relation extraction problems including those in specialized domains. Overall, our work contributes to the advancement of modern information extraction by offering end-to-end solutions that are highly competitive in terms of efficiency and accuracy.

7.1 Contributions

We highlight the contributions of this dissertation as follows.

- Chapter 3 focused on extracting protein-protein interactions, that are additionally affected by a mutation, from biomedical literature. In this study, we showed that we can greatly improve recall by additionally consulting external knowledge sources. And, we showed that our end-to-end approach improves over prior works that focus solely on optimizing relation classification and rely exclusively on existing tools for entity recognition. As of this writing, our model remains state-of-the-art for this problem and dataset.
- Chapter 4 focused on extracting drug-drug interactions from drug labels. We show that our proposed variant of the graph convolutional network (GCN), with a novel attention-gating mechanism (holistically, GCA), improves over the standard GCN. We further showed that the GCA, in conjunction with pre-training on external DDI data, improves substantially over prior BiLSTM-based approaches. Lastly, we showed that GCA and BiLSTM predictions are complementary, and demonstrate that ensembling GCA and BiLSTM models can result in further performance gains. Our approach is currently state-of-the-art among models trained and tested on solely on the official FDA and NLM-supplied dataset.
- Chapter 5 focused on devising an approach capable of jointly extracting both entities and relations without the need for local classifiers. We validated our method on two datasets in both the general English and biomedical domain and

showed that this approach improves over the prior best in terms of raw relation extraction performance. We also experimentally showed that, in lieu of "global optimization", our approach is able to extraction relations with extraction times that are up to an order of magnitude faster than the prior best. Finally, we presented a visualization of intermediate layer activities that confirm our initial intuition for this particular architecture.

- Chapter 6 presented an architecture for relation extraction that, in addition to being *end-to-end*, is able to handle relations that are N-arity and expressed over multiple sentences. We also showed that by additionally accounting for cross-sentence and higher-arity relations, we improved over baseline approaches drastically owing to substantial gains in recall. Our analysis showed that learning to extract higher-arity relations has a side effect of improving the quality of binary relations extracted. And lastly, we demonstrated that ensembling the output of model variants that train and test on different filtering modes (corresponding to different levels of restrictiveness for early elimination of triple candidates), that appear to complement eachother, can lead to nontrivial gains in overall model performance.

7.2 Limitations and Future Work

A universal problem in this domain is the lack of quality datasets for evaluation. Guidelines for annotating datasets are designed around the problem they are intended to solve. There are few gold standard datasets, and the few available datasets tend to be highly specialized and vary widely in terms of problem specification. While some datasets are produced for the sole purpose of methodological evaluation, and thus adhere to established standards, many datasets in RE are often produced on an as-needed basis to tackle real-world problems. Thus each problem-dataset pair has its own set of nuances and idiosyncrasies. Even within the extremely narrow problem of extracting drug-drug interactions, as discussed in Chapter 4, there are several comparable datasets derived from varying annotation schemes and guidelines with different levels of annotation granularity.

The lack of portability is a hindrance in attempting to compare methods designed for different flavors of the same problem. While our methodological advances via neural-metric learning is highly generalizable, given the strong adherence to standard problem specifications, other works in this dissertation are less so. For example, our work on protein-protein interaction requires an additional entity normalization step occurring in between NER and RE. In both of our work on extracting protein-protein interactions and drug-gene-mutation interactions, gold annotations are only available at the *document* level — as opposed to mention-level — thus requiring document-level evaluations. And in our work on drug-drug interactions, there is an additional nontrivial subtask involving the prediction of the outcome of interactions. While we believe there are common elements among these specialized relation extraction tasks that are readily generalizable, it is impossible to draw direct comparisons among them. Thus, a limitation of this thesis is a lack of analyses of the methodologies as a collective.

Given end-to-end relation extraction is a relatively new problem domain, one impactful avenue for future research is to establish a universal framework for crafting portable problem specifications and annotation guidelines. Once established, work can begin to map existing datasets to the new standard. This would benefit the field in several ways. First, moving forward, methods designed for one problem domain can be quickly and easily evaluated on other problem domains to assess for generalizability. Second, annotators trained on one annotator guideline would quickly adapt to other, thus easing human effort on the part of linguists and domain experts. And lastly, with a shared standard for end-to-end relation extraction data sets, we expect performance to improve across the board (“a rising tide lifts all boats”), given the newly expanded scope of available training data and the well-known effectiveness of modern transfer learning and domain-adaptation techniques.

Copyright© Tung Tran, 2020.

Abbreviations

AE — Adverse Event.
ADE — Adverse Drug Event.
API — Application Programming Interface.
ASCII — American Standard Code for Information Interchange.
AWP — All Word Pairs.
AT — Adversarial Training.
BIES — Beginning, Inside, Ending, Single Tagging Scheme.
BILOU — Beginning, Inside, Last, Outside, Unit Tagging Scheme.
BL — Bidirectional LSTM.
CF — Convolutional Filter.
CNN — Convolutional Neural Networks.
CP — Consequence Prediction.
CRF — Conditional Random Fields.
CYK — Cocke-Younger-Kasami algorithm.
DDI — Drug-Drug Interaction.
DYN — Pharmacodynamic Interaction Tag.
EN — Entity Normalizaiton.
EFF — Effect Mention Tag.
FDA — U.S. Food and Drug Administration.
FN — False Negative.
FP — False Positive.
GCN — Graph Convolutional Networks.
GC — Graph Convolution.
GCA — Graph Convolution with Attention-Gating.
GM — Gene Mention.
GN — Gene Normalization.
IE — Information Extraction.
IOB — Inside, Outside, Beginning Tagging Scheme.
IOBES — Inside, Outside, Beginning, Ending, Single Tagging Scheme.
JAX-CKB — Jackson Laboratory Clinical Knowledgebase.
KIN — Pharmacokinetic Interaction Tag.
LP — Linear Programming.
LSTM — Long Short-Term Memory.

NCBI — U.S. National Center for Biotechnology Information.
NCI — U.S. National Cancer Institute.
NER — Named Entity Recognition.
NIST — U.S. National Institute for Standards and Technology.
NLM — U.S. National Library of Medicine.
NLP — Natural Language Processing.
NN — Neural Networks.
PD — Pharmacodynamic Interaction.
PK — Pharmacokinetic Interaction.
PMID — PubMed Identifier.
PPI — Protein-Protein Interactions.
RE — Relation Extraction.
RNN — Recurrent Neural Network.
RC — Relation Classfication.
SCE — Softmax Cross Entropy.
SGD — Stochastic Gradient Descent.
SL — Sequence Labelling.
SPL — Structured Product Labelling.
SVM — Support Vector Machines.
TAC — Text Analysis Conference.
TP — True Positive.
TRI — Trigger Mention Tag.
TREC — Text REtrieval Conference.
UMLS — Unified Medical Language System.
UN — Unspecified Interaction.

Bibliography

[1] Tung Tran and Ramakanth Kavuluru. Predicting mental conditions based on "history of present illness" in psychiatric notes with deep neural networks. *Journal of Biomedical Informatics*, pages S138–S148, 2017.

[2] Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8, 2004.

[3] Makoto Miwa and Yutaka Sasaki. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, 2014.

[4] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 402–412, 2014.

[5] Fei Li, Meishan Zhang, Guohong Fu, and Donghong Ji. A neural joint model for entity and relation extraction from biomedical text. *BMC bioinformatics*, 18(1):198, 2017.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D14-1181`.

[8] Anthony Rios and Ramakanth Kavuluru. Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 258–267. ACM, 2015.

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[10] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[12] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709, 2013.

[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of 3th International Conference on Learning Representations (ICLR)*, 2015.

[14] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*, 28:1310–1318, 2013.

[15] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471, 2000.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012. ISBN 978-3-642-24796-5. doi: 10.1007/978-3-642-24797-2.

[18] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[19] Razvan C Bunescu and Raymond J Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731. Association for Computational Linguistics, 2005.

[20] Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics*, volume 1, pages 697–704. Association for Computational Linguistics, 2008.

[21] Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 9(11):S2, 2008.

[22] Katrin Fundel, Robert Küffner, and Ralf Zimmer. RelEx - relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[23] Jiexun Li, Zhu Zhang, Xin Li, and Hsinchun Chen. Kernel-based learning for biomedical relation extraction. *Journal of the Association for Information Science and Technology*, 59(5):756–769, 2008.

[24] Arzucan Özgür, Thuy Vu, Güneş Erkan, and Dragomir R Radev. Identifying gene-disease associations using centrality on a literature mined gene-interaction network. *Bioinformatics*, 24(13):i277–i285, 2008.

[25] Bryan Rink, Sanda Harabagiu, and Kirk Roberts. Automatic extraction of relations between medical concepts in clinical texts. *Journal of the American Medical Informatics Association*, 18(5):594–600, 2011.

[26] Oana Frunza, Diana Inkpen, and Thomas Tran. A machine learning approach for identifying disease-treatment relations in short texts. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):801–814, 2011.

[27] Shengyu Liu, Kai Chen, Qingcai Chen, and Buzhou Tang. Dependency-based convolutional neural network for drug-drug interaction extraction. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1074–1080. IEEE, 2016.

[28] Ramakanth Kavuluru, Anthony Rios, and Tung Tran. Extracting drug-drug interactions with word and character-level recurrent neural networks. In *Fifth IEEE International Conference on Healthcare Informatics (ICHI)*, pages 5–12. IEEE, 2017.

[29] Desh Raj, SUNIL SAHU, and Ashish Anand. Learning local and global contexts using a convolutional recurrent network model for relation classification in biomedical text. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 311–321, 2017.

[30] Yuan Luo, Yu Cheng, Özlem Uzuner, Peter Szolovits, and Justin Starren. Segment convolutional neural networks (Seg-CNNs) for classifying relations in clinical notes. *Journal of the American Medical Informatics Association*, 25(1): 93–98, 2017.

[31] Yuhao Zhang, Peng Qi, and Christopher D Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[32] Yejin Choi, Eric Breck, and Claire Cardie. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, 2006.

[33] Rohit J Kate and Raymond J Mooney. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL 2010)*, pages 203–212, 2010.

[34] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., 2008. ISBN 0131873210.

[35] Xiaofeng Yu and Wai Lam. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1399–1407, 2010.

[36] Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 1–6, 2013.

[37] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8, 2002.

[38] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1105–1116, 2016.

[39] Arzoo Katiyar and Claire Cardie. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 917–928, 2017.

[40] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[41] Suncong Zheng, Yuexing Hao, Dongyuan Lu, Hongyun Bao, Jiaming Xu, Hongwei Hao, and Bo Xu. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing*, 257:59–66, 2017.

[42] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1227–1236, 2017.

[43] Sachin Pawar, Pushpak Bhattacharyya, and Girish Palshikar. End-to-end relation extraction using neural networks and markov logic networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 818–827, 2017.

[44] Patrick Verga, Emma Strubell, and Andrew McCallum. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. *arXiv preprint arXiv:1802.10569*, 2018.

[45] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45, 2018.

[46] Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, 2017.

[47] Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 506–514, 2018.

[48] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2830–2836, 2018.

[49] Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC bioinformatics*, 6(1):S1, 2005.

[50] Tung Tran and Ramakanth Kavuluru. Exploring a deep learning pipeline for the BioCreative VI precision medicine task. In *BioCreative VI Workshop*, pages 107–110, 2017.

[51] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. GNormPlus: an integrative approach for tagging genes, gene families, and protein domains. *BioMed research international*, 2015, 2015.

[52] Alexander A Morgan, Zhiyong Lu, Xinglong Wang, Aaron M Cohen, Juliane Fluck, Patrick Ruch, Anna Divoli, Katrin Fundel, Robert Leaman, Jörg Hakenberg, et al. Overview of BioCreative II gene normalization. *Genome biology*, 9(2):S3, 2008.

[53] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic acids research*, 41(W1):W518–W522, 2013.

[54] Qingyu Chen, Nagesh C. Panyam, Aparna Elangovan, Melissa Davis, and Karin Verspoor. Document triage and relation extraction for protein-protein interactions affected by mutations. In *BioCreative VI Workshop*, pages 102–105, 2017.

[55] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016.

[56] Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999.

[57] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[58] Sampo Pyysalo, Filip Ginter, Hans Moen, Tapio Salakoski, and Sophia Ananiadou. Distributional semantics resources for biomedical text processing. In *Proceedings of 5th International Symposium on Languages in Biology and Medicine*, pages 39–44, 2013.

[59] Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning*, pages 265–272. Omnipress, 2011.

[60] Donna Maglott, Jim Ostell, Kim D Pruitt, and Tatiana Tatusova. Entrez Gene: gene-centered information at NCBI. *Nucleic acids research*, 39(suppl_1):D52–D57, 2010.

[61] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[62] Daniel R Levinson. Adverse events in hospitals: national incidence among medicare beneficiaries. *Department of Health and Human Services Office of the Inspector General*, 2010.

[63] Linda T Kohn, Janet M Corrigan, and Molla S Donaldson. *To err is human: building a safer health system*, volume 6. National Academies Press, 2000.

[64] Dina Demner-Fushman, Kin Wah Fung, Phong Do, Richard D. Boyce, and Travis Goodwin. Overview of the tac 2018 drug-drug interaction extraction from drug labels track. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, pages 1–10, 2018.

[65] María Herrero-Zazo, Isabel Segura-Bedmar, Paloma Martínez, and Thierry Declerck. The ddi corpus: An annotated corpus with pharmacological substances

and drug–drug interactions. *Journal of biomedical informatics*, 46(5):914–920, 2013.

[66] Tung Tran, Ramakanth Kavuluru, and Halil Kilicoglu. A multi-task learning framework for extracting drugs and their interactions from drug labels. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, pages 1–11, 2018.

[67] Yuan Luo, Özlem Uzuner, and Peter Szolovits. Bridging semantics and syntax with graph algorithmsóÀẺstate-of-the-art of extracting biomedical relations. *Briefings in bioinformatics*, 18(1):160–178, 2016.

[68] Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. Drug-drug interaction extraction via convolutional neural networks. *Computational and mathematical methods in medicine*, 2016, 2016.

[69] Zhehuan Zhao, Zhihao Yang, Ling Luo, Hongfei Lin, and Jian Wang. Drug drug interaction extraction from biomedical literature using syntax convolutional neural network. *Bioinformatics*, 32(22):3444–3453, 2016.

[70] Víctor Suárez-Paniagua, Isabel Segura-Bedmar, and Paloma Martínez. Exploring convolutional neural networks for drug–drug interaction extraction. *Database*, 2017, 2017.

[71] Sunil Kumar Sahu and Ashish Anand. Drug-drug interaction extraction from biomedical texts using long short-term memory network. *Journal of biomedical informatics*, 86:15–24, 2018.

[72] Sangrak Lim, Kyubum Lee, and Jaewoo Kang. Drug drug interaction extraction from the literature using a recursive neural network. *PloS one*, 13(1):e0190926, 2018.

[73] Masaki Asada, Makoto Miwa, and Yutaka Sasaki. Enhancing drug-drug interaction extraction from texts by molecular structure information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–685, 2018.

[74] Xia Sun, Ke Dong, Long Ma, Richard Sutcliffe, Feijuan He, Sushing Chen, and Jun Feng. Drug-drug interaction extraction via recurrent hybrid convolutional neural networks with an improved focal loss. *Entropy*, 21(1):37, 2019.

[75] Siliang Tang, Qi Zhang, Tianpeng Zheng, Mengdi Zhou, Zhan Chen, Lixing Shen, Xiang Ren, Yueting Zhuang, Shiliang Pu, and Fei Wu Wu. Two step joint model for drug drug interaction extraction. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, 2018.

[76] Bharath Dandala, Diwakar Mahajan, and Ananya Poddar. IBM Research system at TAC 2018: Deep learning architectures for drug-drug interaction extraction from structured product labels. In *Proceedings of the 2018 Text Analysis Conference (TAC 2018)*, 2018.

[77] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, 2009.

[78] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[79] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13 (2):260–269, 1967.

[80] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*, 2016.

[81] J-D Kim, Tomoko Ohta, Yuka Tateisi, and JunóÀẼichi Tsujii. Genia corpusóÀẺa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19 (suppl_1):i180–i182, 2003.

[82] Rosie Jones, Andrew McCallum, Kamal Nigam, and Ellen Riloff. Bootstrapping for text learning tasks. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, volume 1, 1999.

[83] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[84] Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceed-*

ings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2537–2547, 2016.

[85] Meishan Zhang, Yue Zhang, and Guohong Fu. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, 2017.

[86] Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of biomedical informatics*, 45(5):885–892, 2012.

[87] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, 2015.

[88] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING 2014)*, pages 2335–2344, 2014.

[89] Heike Adel and Hinrich Schütze. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1723–1729, 2017.

[90] Fei Li, Yue Zhang, Meishan Zhang, and Donghong Ji. Joint models for extracting adverse drug events from biomedical text. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, volume 2016, pages 2838–2844, 2016.

[91] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[92] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd*

International Conference on Machine Learning (ICML 2015), pages 448–456, 2015.

[93] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[94] Sara E Patterson, Rangjiao Liu, Cara M Statz, Daniel Durkin, Anuradha Lakshminarayana, and Susan M Mockus. The clinical trial landscape in oncology and connectivity of somatic mutational profiles to targeted therapies. *Human genomics*, 10(1):4, 2016.

[95] Tung Tran and Ramakanth Kavuluru. An end-to-end deep learning architecture for extracting protein-protein interactions affected by genetic mutations. *Journal of Biological Databases and Curation (Database)*, 2018:1–13, 2018.

[96] Huiwei Zhou, Zhuang Liu, Shixian Ning, Yunlong Yang, Chengkun Lang, Yingyu Lin, and Kun Ma. Leveraging prior knowledge for protein–protein interaction extraction with memory network. *Database*, 2018, 2018.

[97] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115, 2017.

[98] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. N-ary relation extraction using graph-state lstm. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2226–2235, 2018.

[99] Angrosh Mandya, Danushka Bollegala, Frans Coenen, and Katie Atkinson. Combining long short term memory and convolutional neural network for cross-sentence n-ary relation extraction. *arXiv preprint arXiv:1811.00845*, 2018.

[100] Robin Jia, Cliff Wong, and Hoifung Poon. Document-level n-ary relation extraction with multiscale representation learning. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1–13, 2019.

[101] Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. Overview of the biocreative v chemical disease relation (cdr) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, pages 154–166. Sevilla Spain, 2015.

[102] Rezarta Islamaj Dogan, Sun Kim, Andrew Chatr-aryamontri, Chih-Hsuan Wei, Donald C. Comeau, and Zhiyong Lu. Overview of the biocreative vi precision medicine track: Mining protein interactions and mutations for precision medicine. In *BioCreative VI Workshop*, pages 83–87, 2017.

[103] Martin Krallinger, Obdulia Rabal, Saber A. Akhondi, MartÕắn PÕẳrez PÕẳrez, JesÕể s Santa-marÕắa, PÕẳrez Gael RodrÕắguez, Georgios Tsatsaronis, Ander Intxaurrondo, JosÕẳ Antonio LÕẽpez, Umesh Nandal, Erin Van Buel, Akileshwari Chan-drasekhar, Marleen Rodenburg, Astrid Laegreid, Marius Doornenbal, Julen Oyarzabal, Analia LourenÕấo, and Alfonso Valencia. Overview of the biocreative vi chemical-protein interaction track. In *BioCreative VI Workshop*, pages 141–146, 2017.

[104] Yifan Peng, Anthony Rios, Ramakanth Kavuluru, and Zhiyong Lu. Extracting chemical–protein relations with ensembles of svm and deep learning models. *Database*, 2018, 2018.

[105] Chris Quirk and Hoifung Poon. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182, 2017.

[106] Hai Wang and Hoifung Poon. Deep probabilistic logic: A unifying framework for indirect supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, page 1891óÀẼ1902, 2018.

[107] Jinghang Gu, Fuqing Sun, Longhua Qian, and Guodong Zhou. Chemical-induced disease relation extraction via attention-based distant supervision. *BMC bioinformatics*, 20(1):403, 2019.

[108] Gitansh Khirbat, Jianzhong Qi, and Rui Zhang. N-ary biographical relation extraction using shortest path dependencies. In *Proceedings of the Australasian Language Technology Association Workshop 2016*, pages 74–83, 2016.

[109] Soumia Lilia Berrahou, Patrice Buche, Juliette Dibie, and Mathieu Roche. Xart: Discovery of correlated arguments of n-ary relations in text. *Expert Systems with Applications*, 73:115–124, 2017.

[110] Marco Fossati, Emilio Dorigatti, and Claudio Giuliano. N-ary relation extraction for simultaneous t-box and a-box knowledge base augmentation. *Semantic Web*, 9(4):413–439, 2018.

[111] Kosuke Akimoto, Takuya Hiraoka, Kunihiko Sadamasa, and Mathias Niepert. Cross-sentence n-ary relation extraction using lower-arity universal schemas. 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)*, pages 6226–6232, 2019.

Vita

Name

Tung Tran

Education

2014–2018, *M.S. in Computer Science*, University of Kentucky, Lexington, KY, USA
2010–2014, *B.S. in Computer Science*, University of Kentucky, Lexington, KY, USA

Experience

2016–Present, *Graduate Research Assistant*, University of Kentucky
2017, *Summer Research Intern*, U.S. National Institutes of Health (CGSB/NLM/NIH)
2014–2016, *Teaching Assistant*, University of Kentucky
2013–2014, *Software Engineering Intern*, PatentRank
2011–2012, *Undergraduate Research Assistant*, University of Kentucky

Awards

2019 — Departmental Fellowship, 2019–2020 Academic Year, CS, UKY
2018 — Ranked **2nd out of 8 teams** in the TAC 2018 task on DDI extraction
2018 — Biomedical Informatics Training Program Appointee, NLM, NIH
2017 — Ranked **2nd out of 6 teams** in the BioCreative VI task on PPI extraction
2016 — Graduate School Travel Grant, UKY
2015 — Departmental Nominee for the Microsoft PhD Fellowship Program, CS, UKY
2014 — Graduated Cum Laude, B.S. in Computer Science, UKY
2010-2014 — Dean's List for Three Semesters, UKY
2010-2014 — Kentucky Educational Excellence Scholarship (KEES)
2010 — Academic Competitiveness Grant
2010 — Catalyst Scholarship

Publications

1. **T. Tran** and R. Kavuluru. Social Media Surveillance for Perceived Therapeutic Effects of Cannabidiol (CBD) Products. *International Journal of Drug Policy*, 2020.

2. **T. Tran** and R. Kavuluru. Distant Supervision for Treatment Relation Ex-

traction by Leveraging MeSH Subheadings. *Artificial Intelligence in Medicine*, 2019.

3. H. Kilicoglu, Z. Peng, S. Tafreshi, **T. Tran**, G. Rosemblat, and J. Schneider. Confirm or Refute?: A Comparative Study on Citation Sentiment Classification in Clinical Research Publications. *Journal of Biomedical Informatics*, 2019.
4. R. DoỎỈan, S. Kim, A. Chatr-aryamontri, C. Wei, D. Comeau, and **22** others, including **T. Tran**. Overview of the BioCreative VI Precision Medicine Track: mining protein interactions and mutations for precision medicine. *Database: Journal of Biological Databases and Curation*, 2019.
5. **T. Tran**, R. Kavuluru, and H. Kilicoglu. A Multi-Task Learning Framework for Extracting Drugs and Their Interactions from Drug Labels. In Proceedings of *The Eleventh Text Analysis Conference (TAC)*, 2018.
6. A. Sarker, M. Belousov, J. Friedrichs, K. Hakala, S. Kiritchenko, F. Mehryary, S. Han, **T. Tran**, and **8** others. Data and systems for medication-related text classification and concept normalization from Twitter: Insights from the Social Media Mining for Health (SMM4H) 2017 shared task. *Journal of the American Medical Informatics Association*, 2018.
7. **T. Tran** and R. Kavuluru. An End-to-End Deep Learning Architecture for Extracting Protein-Protein Interactions Affected by Genetic Mutations. *Database: Journal of Biological Databases and Curation*, 2018.
8. A. Rios, **T. Tran**, and R. Kavuluru. Predicting Psychological Health from Childhood Essays with Convolutional Neural Networks for the CLPsych 2018 Shared Task (Team UKNLP). In Proceedings of *The Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic (CLPsych)*, 2018.
9. **T. Tran** and R. Kavuluru. Supervised Approaches to Assign Cooperative Patent Classification (CPC) Codes to Patents. In Proceedings of *The Fifth International Conference on Mining Intelligence and Knowledge Exploration (MIKE)*, 2017.
10. **T. Tran** and R. Kavuluru. Exploring a Deep Learning Pipeline for the BioCreative VI Precision Medicine Task. In Proceedings of *The BioCreative VI Workshop*, 2017.

11. S. Han, **T. Tran**, A. Rios, and R. Kavuluru. Team UKNLP: Detecting ADRs, Classifying Medication In-take Messages, and Normalizing ADR Mentions on Twitter. In Proceedings of *The 2nd Social Media Mining for Health Applications Workshop and Shared Task at AMIA (SMM4H)*, 2017.
12. R. Kavuluru, A. Rios, and **T. Tran**. Extracting drug-drug interactions with word and character-level recurrent neural networks. In Proceedings of *The Fifth IEEE International Conference on Healthcare Informatics, Workshop on Healthcare Knowledge Discovery and Management (ICHI)*, 2017.
13. **T. Tran** and R. Kavuluru. Predicting Mental Conditions Based on "History of Present Illness" in Psychiatric Notes with Deep Neural Networks. *Journal of Biomedical Informatics*, 2017.