NEURAL NETWORK ALGORITHMS FOR ONTOLOGY INFORMED INFORMATION EXTRACTION

by

Dongfang Xu

A Dissertation Submitted to the Faculty of the

SCHOOL OF INFORMATION

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2020

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by: Dongfang Xu, titled: Neural Network Algorithms for Ontology Informed Information Extraction

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Steven Bethard (Dec 21, 2020 22:19 MST) Date: 21 Dec 2020
Steven Bethard

hong cui (Dec 22, 2020 11:06 MST) Date: 22 Dec. 2020
Hong Cui

Mihai Surdeanu (Dec 22, 2020 20:05 MST) Date: 22 Dec. 2020
Mihai Surdeanu

Date: 23 Dec 2020
Timothy Miller

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Steven Bethard (Dec 24, 2020 15:07 MST) Date: 24 Dec 2020
Steven Bethard
Dissertation Committee Chair
School of Information

ACKNOWLEDGEMENTS

I imagined many dark moments in my PhD journey when I started in 2015: lonely, helpless, stressed, exhausted, etc. By the time I am standing here, I realize none of these truly happened. The path was not easy and obvious for me, but it was always full of joy and peace thanks to the help and support from many, many people.

First and foremost, my deepest thanks go to my advisor Steve, also a mentor and a role model to me. Since the first interaction with him, I was quickly impressed by his research work and how he explained it, straightforward, humorous, and insightful. In the past four and a half years working with him, I enjoyed countless moments when he described approach diagrams and research plans, when we brainstormed about neural network algorithms and paper ideas, and when we worked on the papers. I have learned so much from him, yet there are several other things I still eager to learn, like his enthusiasm, time management, and latex skills.

I would like to express my sincere gratitude to my co-advisor, Hong, for her encouragement, support, and guidance. Without her, I could not have started my PhD study. I learned a lot about ontologies and semantic web technologies from her, which are essential for my dissertation topic. She is always available for a random chat about my research and graduation plan.

It has been a great honor to have Mihai and Tim on my dissertation committee. They provided a variety of invaluable discussions and feedback on how to approach, organize, and present my work. Special thanks to Mihai for his instructions in NLP classes, and organizing the NLP reading group where I got to know lots of cool NLP folks. The work I have done in my PhD journey is very relevant to Tim's research, and I look forward to working with him in the near future.

I would like to express my gratitude to my marvelous friends, teammates, and collaborators: Dr. Egoitz Laparra, Dr. Vikas Yadav, and Dr. Farig Sadeque. Thanks for sharing most of my PhD journey: lots of suggestions and brainstorms, regular happy hours or online meetings with chit-chat and laughter, and a relaxing working environment. Thank the whole ischool NLP group, especially Yiyun, Masha, and Sarah, for their comments and feedback on my presentations and research ideas. Special thanks to Manoj, Zeyu, and Jiacheng for their contributions to my research work.

Thanks to all the professors who have equipped me with the knowledge that made this dissertation possible. Thanks to all the supervisors at ORNL and GE Healthcare for making my internships memorable. Outside of the professional area, I have been fortunate to be surrounded by many great friends from the iSchool Ghosts, soccer teams, universities, and high school.

Lastly, I thank my family for their unconditional love and support all the time. Without you, I couldn't be here. Video calls with my parents Yuanxiang and Guojin, and my sister

Zhenzhen are always the best way to recharge my internal batteries. My wife, Xingyi, I thank her for everything she has done for me. Her great personality has affected me since we met in 2010. Her emotional support arms me with strength and always reminds me to be a better man.

The PhD journey has been incredible, and thank you all for raising me up.

DEDICATION

To my parents and Xingyi.

This is your achievement as much as it is mine.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES – *Continued*

LIST OF TABLES – *Continued*

ABSTRACT

Ontology, as a formal and explicit specification of a shared conceptualization for a particular domain, is useful in information extraction. On the one hand, since information extraction is concerned with retrieving information for a particular domain, formally and explicitly specifying the concepts of that domain through an ontology defines the boundary of what information needs to be extracted. On the other hand, an ontology, typically consisting of classes (or concepts), attributes (or properties), and relationships (or relations among class members), contains the structured information that information extraction systems aim to extract. In this thesis, we are interested in how using an ontology can improve the information extraction process. We explore two research directions that both employ ontologies in the information extraction process, temporal normalization and biomedical concept normalization. In both research directions, we show that leveraging resources in ontologies helps to build high-performance information extraction systems, and presenting the extracted output using such ontologies makes the structured information concise and interchangeable.

CHAPTER 1

Introduction

1.1 Motivation

With the explosion of information in the form of news, blogs, social media posts, scientific articles, government documents, medical records, etc., evaluating and selecting relevant information becomes challenging than ever before. One consequence is information overload, occurring when decision-makers face a level of information that is greater than their information processing capacity. Research (Eppler and Mengis, 2004; Misra and Stokols, 2012; Roetzel, 2019) has found many impacts of information overload on people, e.g., a general lack of perspective, a greater tolerance of error, cognitive strain and stress, etc. However, a significant part of such information lies in unstructured form, i.e., free text documents, and is thus hard to search. Besides, the massive amounts of heterogeneous information sources make it a non-trivial task for human users to organize them manually. This results in a growing need for effective and efficient techniques for analyzing free-text data and discovering valuable and relevant knowledge from it in the form of structured information.

Information Extraction (IE), as one field of natural language processing (NLP), uses computational methods to identify relevant pieces of information in text and convert

it into a representation suitable for computer-based storage, processing, and retrieval (Wimalasuriya and Dou, 2010). It benefits many applications, for example, removing personally identifiable data from web news, finding drug-gene interactions from medical research articles, integrating product information from various websites, constructing knowledge graph from Wikipedia, automatically answering questions for elementary science exams, etc.

The task of IE is to identify a predefined set of concepts in a specific domain, ignoring other irrelevant information, where a domain consists of a corpus of texts and a specified information need (Piskorski and Yangarber, 2013). Consider, for example, the description of respiratory viral infections from one article below:

> **Contini et al. (2020):** 18 years ago, the world was astonished by the appearance of Severe Acute Respiratory Syndrome (SARS), supported by a zoonotic coronavirus, called SARS-CoV, from the Guangdong Province of southern China. After about 10 years, in 2012, another similar coronavirus triggered the Middle East Respiratory Syndrome (MERS-CoV) in Saudi Arabia. ... 8 years later, despite the MERS outbreak remaining in certain parts of the world, at the end of 2019, a new zoonotic coronavirus (SARS-CoV-2) and responsible of coronavirus Disease (COVID-19), arose from Wuhan, Hubei Province, China.

This text describes the epidemics of coronavirus pneumonia in 2002, 2012 and 2019. Assume we are interested in the epidemics of coronavirus pneumonia, their causative virus,

Epidemic	Causative virus	Year	Starting from
SARS (Severe acute respiratory syndrome)	SARS-CoV	2002	Guangdong
MERS (Middle East respiratory syndrome)	MERS-CoV	2012	Saudi Arabia
COVID-19 (Coronavirus disease 2019)	SARS-CoV-2	2019	Wuhan

Table 1.1: The processed and organized information from a scientific article on epidemics of coronavirus pneumonia.

timeline, and the starting place. As a human being, we have to finish reading the text line by line and then we could understand the key information about the epidemics (shown in table 1.1). We will know **COVID-19** or **coronavirus disease 2019** refer to the same epidemic, it was caused by the virus **SARS-CoV-2**, also called **novel coronavirus** or **2019-nCoV**, and it was first found in **Wuhan**. We also understand that given the publication year of Contini et al. (2020), **18 years ago** means **2002**. For a computer, quickly identifying certain proper nouns or noun phrases that denote the concepts of epidemics, viruses and time expressions is not an unsolvable problem, but due to the complexity and ambiguity of natural language, aggregating them into a structured form where their semantics are specified is a non-trivial task. There are many ways of expressing the same fact, which can be distributed across multiple words, sentences, or even documents.

In terms of understanding the information need and providing domain knowledge, we argue that ontologies are useful in the IE process. An ontology is defined as a formal and explicit specification of a shared conceptualization for a particular domain (Gruber, 1993; Studer, Benjamins, and Fensel, 1998). While as resources in computer and information sciences, ontologies have fewer constraints than the definition of formal ontologies: they come in various forms, ranging from lexicons, to dictionaries and thesauri, or even first-

order logical theories. Either as conceptual models or resources, we believe the goals of using ontologies remain the same. On the one hand, since IE is concerned with retrieving pre-defined concepts for a particular domain, formally and explicitly specifying the concepts of that domain through an ontology defines the boundary of what information needs to be extracted. On the other hand, an ontology, typically consisting of classes (or concepts), attributes (or properties), and relationships (or relations among class members), contains the structured information or background knowledge to guide the IE process. For instances, annotation by template filling, commonly used to train machine learning-based IE systems (Ciravegna, 2001), can be seen as the task of finding and marking all the attributes of a given ontological concept in a text. Meanwhile, the extracted information in IE is a useful source of knowledge to design and enrich ontologies. Such knowledge representation has the potential to assist the development of the Semantic Web. Nédellec (2005) summarize that "IE and ontologies are combined in a cyclic process: ontologies are used for interpreting the text at the right level for IE to be efficient, and IE extracts new knowledge from the text, to be integrated into ontologies".

In this thesis, we are particularly interested in how ontologies can improve the IE process. Existing work on this topic is also called ontology based information extraction, which has the following two key characteristics (Wimalasuriya and Dou, 2010):

- Guide the IE process using an ontology: leveraging the resources in an ontology such as classes, properties, instances, or relationships to create rules, extract features, or implement algorithms in the IE system.

- Present the IE output using an ontology: linking the extracted information to its semantic descriptions in an ontology.

The aims of this thesis are aligned with these two characteristics: 1) apply ontological resources to build high-performance IE systems; and 2) present the extracted output using an ontology to make information concise and interchangeable. However, as we mainly focus on the ontological resources such as classes, taxonomy, and semantic types, not including semantic relations as most ontology based information extraction systems do, we name our research topic as ontology informed information extraction (OIIE).

1.1.1 Research focus

In this thesis, we explore two research directions that both employ ontologies in the IE process:

Temporal Normalization A task that maps a temporal expression to either a specific point in time or to a duration.

Biomedical Concept Normalization A task that maps concept mentions, the in-text natural-language mentions of ontological concepts, to concept entries in an ontology or a knowledge base.

In both research directions, we implement neural network algorithms to map natural language inputs to output spaces with formally defined semantics. Specifically, for the time normalization, we use the Semantically Compositional Annotation of Time Expressions

(SCATE) scheme (Bethard and Parker, 2016) to present the extracted time expressions, where each time expression is annotated in terms of compositional time entity over intervals on the timeline; for the biomedical concept normalization, we use biomedical ontologies such as the Unified Medical Language System (UMLS) to present the extracted biomedical entities, where each biomedical entity is linked with its corresponding concept in an ontology. Both of these tasks show how ontologies can enrich natural language data with machine-readable annotations. The embedded semantics in annotations could enable a higher level of automation, such as integrating knowledge from heterogeneous data sources or reasoning over data. For instance, time normalization allows entities and events to be placed along a timeline, which is helpful for a machine to understand the temporal and causal relations among them.

1.2 Thesis outline

This thesis is organized as follows:

- In Chapter 2, we first present IE and its classic tasks. Next, we explain our research domain: ontologies in information and computer science areas, and ontologies in the OIIE. We then briefly discuss prior research of OIIE.

- In Chapter 3, we go into the details of our first research work in time normalization. We first compare two schemes for annotating normalized time expressions, ISO-TimeML and SCATE. We begin by describing an interval-based evaluation metric that can compare time normalizations annotated in both schemes, and con-

duct a quantitative analysis of their annotations on a subset of the TempEval 2013 corpus (UzZaman et al., 2013). We then introduce a neural parsing system for time normalization and describe its two components, a character-based multi-output neural network for time entity identification, and a rule-based system for time entity composition. We present the experimental results and discuss the performances of the neural time entity identifier and its variants.

- In Chapter 4, we present our advances of using pre-trained contextualized character embeddings in the time entity identification task. We first discuss our motivations and how we derive and apply the pre-trained contextualized character embeddings. We then present the experimental results on two datasets from SemEval 2018 Task 6 (Laparra et al., 2018), and conduct an in-depth analysis of the neural models to understand better where improvements come from and what they depend on. Finally, we discuss the limitations of our work on time normalization and future research directions.

- In Chapter 5, we explore biomedical concept normalization- why we are interested in it, what its challenges are, and how we leverage resources in ontologies to address the task. We first give a high-level overview of the biomedical concept normalization task and some notable progress in the past. Next, we introduce a generate-and-rank framework with semantic type regularization for biomedical concept normalization. We then present the experimental results on four datasets, with three in the domain of

social media posts and one in the domain of clinical notes. In particular, we present our submitted system in the 2019 N2C2 Shared Task Track 3 Concept Normalization (the third-highest performance), and explain how UMLS improves sieve-based generation and BERT-based ranking. Finally, we discuss current limitations and summarize the work.

- Chapter 6 discusses another approach for concept normalization, where we view biomedical concept normalization as an information retrieval task. We first briefly review the literature using vector space models for concept normalization. We then introduce a vector-space model for concept normalization, where mentions and concepts are encoded via transformer networks that are trained via a triplet objective with online hard triplet mining. We also explore a variety of strategies for searching with the trained vector-space model. Next, we present our experimental results on five datasets, with three in the domain of scientific articles and two in the domain of clinical notes. Finally, we discuss its current limitations and future work.

- In Chapter 7, we summarize the works that have been presented in this thesis, and conclude in the end.

1.3 Contributions

This thesis's research shows how we leverage resources in ontologies to implement neural network algorithms and present the extracted output. We first summarize the contributions

of our work on **time normalization** (Laparra, Xu, and Bethard, 2018; Laparra et al., 2018; Xu, Laparra, and Bethard, 2019):

- We compare two different schemes for annotating normalized time expressions, SCATE and TimeML annotations on the same corpus, and find that the SCATE is a better way of modeling and interpreting time expressions. Specifically, SCATE covers a wider variety of time expressions and is more suitable for machine-learning approaches.

- We propose the first machine-learning models trained on the SCATE corpus of time normalizations. Following the taxonomy of the SCATE schema, we deploy a multi-output neural network architecture, and show it outperforms single-output models that do not consider the structural information of time entities. We also showed that our approach outperforms the state-of-the-art HeidelTime (Strötgen, Zell, and Gertz, 2013). We further improved our approach by applying the pre-trained contextual character embeddings into our multi-output neural models.

In terms of **biomedical concept normalization** (Xu, Zhang, and Bethard, 2020; Xu et al., 2020), we make the following contributions:

- We propose a concept normalization framework consisting of a candidate generator and a list-wise classifier. This two-step architecture has the advantage of dealing with an ontology with millions of concepts, and predicting all concepts in an ontology even they are not covered in the training data.

- We leverage the resources such as synonyms or semantic type from an ontology in our proposed framework. In particular, our candidate generator in the N2C2 shared task, a Lucene-based sieve normalization system, combines dictionary look-up over training data, UMLS preferred terms and synonyms. Our BERT-based candidate ranker incorporates semantic types through a regularizer, which encourages the model to consider the semantic type information of the candidate concepts.

- We further explore a simpler and more efficient approach for biomedical concept normalization. Specifically, we propose a triplet network with online hard triplet mining for training a vector-space model for concept normalization, and explore a variety of strategies for matching mentions to concepts using the vector-space model, with the most successful being a simple sieve-based approach that checks domain-specific synonyms before domain-independent ones. Our framework also produces models trained on only an ontology – no domain-specific training data – that can incorporate domain-specific concept synonyms at search time without retraining. These models achieve competitive performances against the state-of-the-art.

CHAPTER 2

Background

This chapter aims to discuss the foundations of ontology informed information extraction (OIIE) and its past research efforts. To better understand what ontological resources can contribute to the development of information extraction (IE), and how using ontologies can help present the extracted information, it is essential to discuss IE and ontologies separately. Thus, in this chapter, we first describe IE and its sub-tasks in Section 2.1. We then explain what ontologies are from the broad perspective of computer and information science and what ontologies are from the perspective of NLP in Section 2.2. Finally, in Section 2.3, we describe prior research on OIIE, and how they apply ontologies in the IE process.

2.1 Information extraction

The IE task is to automatically extract structured information from a collection of unstructured texts, and organize such information with representations that meet users' information needs. As IE is often an early stage in the pipeline for various high-level NLP tasks like text summarization and question answering, the NLP community has put decades of effort into IE.

The structured information that IE is interested in can be categorized into three types:

named entities, events, and relations. Two competitions strongly influence the scopes of such structured information, the Message Understanding Conferences (MUCs) (Sundheim, 1995; Marsh and Perzanowski, 1998) and Automatic Content Extraction (ACE) Program (Doddington et al., 2004). In the following, we describe each of them:

Named entity is typically a noun phrase consisting of more than one token in the unstructured text. The determination of named entity is task- specific, from a common entity such as person, location, organization, to a domain-specific entity like gene, protein or, medication. Named entities are commonly extended to include things that are not entities per se, including dates and times (Pustejovsky et al., 2003b; Pustejovsky et al., 2003a), and even numerical expressions like clinical attributes (Li et al., 2013).

Event is an expression denoting an event or a state that can be assigned to a particular time point or a time interval. Typically, most event mentions correspond to verbs, and most verbs introduce events. Noun phrases can also introduce events. For instance, in the following text, "*[**Event** Citing] high fuel prices, United Airlines [**EVENT** said] Friday it has [**EVENT** increased] fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately [**EVENT** matched] [**EVENT** the move], spokesman Tim Wagner [**EVENT** said]*". Tokens in the curly brackets are events, also called event triggers.

Relationship is defined over two or more entities related in a predefined way. Examples are "is an employee of" relationship between a person and an organization, "is acquired by"

relationship between pairs of companies, “a location of outbreak” relationship between a location and a disease. Unlike entities or events where a sequence of words are annotated, relationships are not annotations over the words. Instead, they express the associations between two separate text snippets of the entities, and the annotations of them require background knowledge or context information.

To extract these elements, various sub-tasks involved in IE include:

Named Entity Extraction (NER) is a task that identifies the named entities in the text. It is typically the first step in most IE tasks.

Coreference Resolution requires the identification of multiple co-referring mentions of the same entity in the text. In addition to the named entities, coreference resolution also finds the pronouns and nominal phrases that refer to the entities.

Named Entity Linking (NEL) links textual mentions of entities with their corresponding knowledge base (KB) or ontology identifiers. NEL helps to disambiguate the named entities and bridge the gap between the unstructured text and KB/ontology.

Relation Extraction (RE) finds the semantic relations among pre-identified entities.

Event Extraction (EE) identifies textual mentions of events, and their event type and arguments such as modality and aspect. Usually, event extraction involves the extraction of several entities and relationships between them, for instance, who did what to whom, when, and where.

Event coreference (EC) is a task to find all the event mentions in a text referring to the same event.

Temporal Information Extraction aims to answer when the events in a text happened, i.e., extract temporal expressions from the text. It typically contains two subtasks, identifying temporal expressions and normalizing them into a standard format.

Temporal Relation Extraction is a special task of relation extraction, where the goal is to detect the temporal orderings of events and time expressions by fitting the events into a complete timeline.

Template Filling is a task to find text that triggers a particular template and then fill the slots of that template with fillers extracted from the text. These slot-fillers consist of text segments extracted from the text, such as events, named entities, and their properties. Several other IE tasks, such as coreference resolution and relations extraction, are also involved in template filling.

2.2 Ontology

Ontology[1], also known as metaphysics, is the study of what exists and what is the nature of reality. The occurrence of ontologies[2] in the computer and information science field can be traced back to the early development of artificial intelligence (AI), when AI researchers

[1]Philosophers use the term “Ontology" with an upper-case O, as it is the Philosophia Prima concerned with metaphysics.(Smith and Welty, 2001).

[2]The term “ontologies" or "an ontology" (plural or singular, small o) is used in computer/information science research.

recognized that knowledge engineering is the key to building large and robust AI systems. They argued that they could create new ontologies as computational models that enable certain kinds of **automated reasoning** (McCarthy, 1980). Ontologies here are artifacts designed to model knowledge about some domain, real or imagined, and such knowledge modeling is in a machine-interpretable form. What these ontologies have in common with the Ontology in philosophy is the representation of entities, ideas, events, and their properties and relations.

In addition to knowledge engineering, Smith and Welty (2001) explained two other starting points of ontologies in computer and information science, conceptual modeling and domain modeling, where the former is concerned with the **interoperability** of databases as the ad hoc and inconsistent modeling leads to many practical problems of database integration, and the latter is concerned about the **reusability** of programs as large and complex programs result in more difficulties in maintaining and putting them to new uses.

Although the use of ontologies was already widespread since the 1980s, the first credible attempt at defining the term happened in 1993: "**a specification of a representational vocabulary for a shared domain of discourse — definitions of classes, relations, functions, and other objects - is called an ontology**" (Gruber, 1993). There are two essential points in an ontology from this definition, vocabulary and the corresponding definitions of terms. The representational vocabulary includes terms for classes, properties, and relationships, while the definitions of these terms could be informal text or specified by a formal language like predicate logic. The advantage of formal definitions for terms

is that these definitions allow a machine to perform much deeper reasoning, while the disadvantage is that these definitions are much more challenging to construct. Although one argument against this definition is overly broad, it allows ontologies with various forms, ranging from lexicons to glossaries to taxonomies to thesauri, or even first-order logical theories (Jurisica, Mylopoulos, and Yu, 2004). For instance, an ontology can be as simple as a product catalog where each product type has a unique code (e.g. the item number); a catalog is, in a sense, the ontology of the things a company sells. We describe a few forms of ontologies (knowledge organization systems) as follows:

Lexicon is a standardized dictionary of terms that represent critical concepts for a domain. There are no synonyms or related terms identified in a lexicon.

Glossary contains a list of terms in a particular domain of knowledge and provides natural language descriptions for terms. It imposes some structure on the text (indexing by terms).

Taxonomy provides a simple hierarchical arrangement among terms (concepts) in one domain, where terms (concepts) are associated with parent-child relationships.

Thesaurus is a kind of dictionary that represents all the concepts for a specific domain in a consistent manner and labels each concept with a preferred term. It contains preferred terms, variant terms, broader and narrower terms, and related terms.

There are two components in common among these different forms, **(1)** concepts (classes) and their aliases in a domain, and **(2)** relationships between these concepts, from simple

hierarchical relationships (subclass relationships), to vague relationships such as is-related-to, or predefined relationships like is-made-of, is-part-of, etc. In any of these forms, ontologies are useful because they encourage the standardization and interconnections of terms that can represent knowledge about a domain. However, all these forms neglect another important component of ontology, **(3)** constraints on what can be expressed using an ontology, e.g., domain and range restrictions of the concept property, cardinality constraints of the relationships, etc. Such constraints help define formal semantics, which enforces the meaning of the expressed knowledge for the ontologies and support automated reasoning. In this thesis, we consider one as an ontology if it contains **(1) and (2)**, or if it contains **(1), (2), and (3)**.

2.3 Ontology informed information extraction

Inspired by previous work of ontology based information extraction (Wimalasuriya and Dou, 2010), we define an **OIIE** system as "a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies".

In terms of guiding the IE process, ontologies help understand the information need and provide the domain knowledge. On the one hand, since IE is concerned with retrieving pre-defined concepts for a particular domain, formally and explicitly specifying the concepts of that domain through an ontology defines the boundary of what information needs to be extracted. On the other hand, ontology, typically consisting of classes (or concepts),

attributes (or properties), and relationships (or relations among class members), contains the structured information or background knowledge that can be used to create rules, extract features, or implement algorithms in IE system. To discuss how ontologies guide the IE process, we describe two kinds of OIIE systems:

Rule-based systems use the components in ontologies to create heuristic rules. There are two different ways to utilize ontological resources (Wimalasuriya and Dou, 2010): 1) gazetteer lists, where the words/phrases to be extracted are provided to the system in the form of a list (Rindflesch et al., 1999; Buitelaar et al., 2006; Saggion et al., 2007), e.g., gazetteer lists containing all the city names used to identify the city of the US; and 2) linguistic rules represented by regular expressions, where the target information appearing in the documents have certain semantic patterns (Appelt et al., 1993; Cunningham, 2002; Valenzuela-Escárcega, Hahn-Powell, and Surdeanu, 2016), e.g., the expression "<NP> (is the capital | city | town of) <NP>", where "<NP>" denotes noun phrases and "(is the capital | city | town of)" are the semantic relationships in an ontology, and might capture the names of geographical locations (represented by the noun phrase) in a set of documents. Several IE systems (Riloff, Jones, et al., 1999; Ono et al., 2001; Thelen and Riloff, 2002) combine both ways of using ontological resources. They use bootstrapping techniques to alternately select the extraction patterns and bootstrap its extractions into the semantic lexicon, which are then used to identify new patterns.

Learning-based systems leverage ontological resources in machine learning or deep learning models. The most straightforward way of using ontologies is to create gazetteer

features, where binary variables check whether terms are present in a gazetteer (Smith and Osborne, 2006; Surdeanu et al., 2011; Xu et al., 2018). Such gazetteer features may not provide discriminative information to the classifiers as terms in a gazetteer are treated as identical. Thus several systems (Chieu, Ng, and Lee, 2003; Xu et al., 2012) further use semantic type features where terms that appear in gazetteer are assigned with a semantic class. With the development of deep learning techniques, recent systems leverage ontological resources in neural networks, e.g., via a multi-task learning framework with an auxiliary task to predict whether the input terms appear in a gazetteer (Niu et al., 2019), a hierarchical loss to inject hypernym/is-a relation (Murty et al., 2018), learning to rank approaches with synonyms of concepts as input (Xu, Zhang, and Bethard, 2020; Sung et al., 2020), etc.

While in terms of organizing the IE outputs, ontologies provide frameworks or techniques to model the extracted information, from a simple entity-quality spreadsheet that stores entities and their attribute values (Cui et al., 2016), to a particular knowledge representation technique, e.g., resource description format (RDF) that notates a semantic triple (subject, predicate, and object) (Gerber et al., 2013), or even to an incomplete ontology that instantiates concepts, taxonomic relations, non-taxonomic relations, and axioms (Jiang and Tan, 2010). Based on how close the organized outputs are to the ontologies, we describe three other tasks from the intersection of Semantic Web and IE (Martinez-Rodriguez, Hogan, and Lopez-Arevalo, 2020):

Semantic Annotation aims to annotate documents with the components in ontologies

such as entities, classes, instances, or relationships. It consists of finding mappings between text chunks of a document and the instances or individuals in ontologies. For example, semantic annotation of bio-systematics literature aims to extract biological structures (organs or parts of an organ) and their physical characteristics (shape or color) from morphological descriptions (Cui, 2012; Cui et al., 2016). Such extracted structures and characteristics could be the concepts from the Phenotype Quality Ontology or FNA Categorical Glossary. Prior surveys on semantic annotation can be found at Uren et al. (2006), Liao and Zhao (2019), and Jovanović and Bagheri (2017).

Knowledge Extraction aims to extract information from an unstructured (or semi-structured) corpus and organize them with knowledge representation techniques such as OWL (ontology web language). Knowledge extraction can be seen as a general IE task but with a stronger focus on modeling the extracted outputs. For example, due to the heterogeneity and ambiguity of information, AIDA (Yosef et al., 2011) maps mentions of ambiguous names onto canonical entities (e.g., individual people or places) registered in the YAGO2 knowledge base. Comprehensive comparisons of a few knowledge extraction tools can be found at Unbehauen et al. (2012) and Gangemi (2013).

Ontology Learning refers to extracting ontological components such as concepts and taxonomic relations from a corpus of domain-specific text and building an ontology from them. It helps automate the costly process of ontology building as the manual acquisition of ontologies requires comprehensive knowledge of a domain, and the manually acquired

ontologies could be incomplete and biased (Hazman, El-Beltagy, and Rafea, 2011). One ontology learning system, CRCTOL (Jiang and Tan, 2010), adopts a full-text parsing technique and employs a combination of statistical and lexico-syntactic methods to extracts key concepts, disambiguates words in the key concepts, and extracts relations between the key concepts. Prior surveys on ontology learning can be found at Hazman, El-Beltagy, and Rafea (2011) and Wong, Liu, and Bennamoun (2012).

CHAPTER 3

Temporal Normalization

In this chapter, we will cover our temporal normalization system, an ontology informed information extraction system that maps time expressions in a text to semantic annotations. Our system has two primary components - time entity identifier and linker. Before diving into the details of our implementations, we give a brief introduction of temporal normalization in Section 3.1. In particular, we describe two scheme for annotating time expressions - ISO-TimeML and semantically compositional annotation of time expressions (SCATE) scheme (Bethard and Parker, 2016). To verify the SCATE schema is able to represent a wider variety of time expressions than TimeML, we adopt an interval-based evaluation metric that can compare time normalizations annotated in both two scheme, and conduct a quantitative analysis of their annotations in the same corpora.

In Section 3.2, we present a character-based multi-output neural network architecture for time entity identification, and a rule-based system for time entity composition. For the time entity identification, we present three different neural network architectures based on the taxonomy of the SCATE schema, and input features to these neural networks. To make the neural networks robust to different formats of document creation time, we automatically generate an extra 800 isolated training examples for a wide variety of such expression formats.

Next, we provide more implementation details of our system, and present the empirical results in terms of input features, neural architectures and evaluation metrics in Section 3.3. We further discuss 1) what errors our current systems are making and 2) why character-based models are better than the word-based models in terms of performances.

3.1 Background and motivation

Temporal normalization is a crucial step for many IE tasks as it allows entities and events to be placed along a timeline. However, unlike signal processing, where the timestamp often naturally comes with the data, timestamps rarely exist in natural language text. Temporal normalization aims to extract timestamps by translating natural language expressions of time to computer-readable forms. For instance, the expression *three days ago* could be normalized to the formal representation *2017-08-28* in the ISO-8601 standard. Since the first shared tasks on time normalization (Verhagen et al., 2007), interest in the problem and the variety of applications have been growing. For example, Lin et al. (2015) use normalized timestamps from electronic medical records to contribute to patient monitoring and detect potential causes of disease. Vossen et al. (2016) identify multi-lingual occurrences of the same events in the news by, among other steps, normalizing time-expressions in different languages with the same ISO standard. Fischer and Strötgen (2015) extract and normalize time-expressions from a large corpus of German fiction as the starting point of a deep study on trends and patterns of the use of dates in literature.

In natural language text, time expressions come into different forms ranging from

formatted timestamps such as *05-01-2015 13:00:00* to calendar dates such as *May 1st* to times of day such as *1 o'clock in the afternoon*. It can be vague timeline such as *once upon a time* or repeated time interval such as *Saturdays since March 6*. A key consideration for time normalization systems is what formal representation the time expressions should be normalized to.

ISO-TimeML (Pustejovsky et al., 2003b; Pustejovsky et al., 2010) is the most popular scheme for annotating time expressions. It annotates time expressions as phrases, and assigns an ISO 8601 normalization (e.g., 1990-08-15T13:37 or PT24H) as the *VALUE* attribute of the normalized form. ISO-TimeML is used in several corpora, including the TimeBank (Pustejovsky et al., 2003a), WikiWars (Mazur and Dale, 2010), TimeN (Llorens et al., 2012), and the TempEval shared tasks (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013).

However, the ISO-TimeML schema has a few drawbacks. First, times that align to more than a single calendar unit (day, week, month, etc.), such as *Saturdays since March 6* (where multiple Saturdays are involved), cannot be described in the ISO 8601 format since they do not correspond to any prefix of YYYY-MM-DDTHH:MM:SS. Second, each time expression receives a single *VALUE*, regardless of the word span, the compositional semantics of the expression are not represented. For example, in the expressions *since last week* and *since March 6*, the semantics of *since* are identical – find the interval between the anchor time (*last week* or *March 6*) and now. But ISO-TimeML would have to annotate these two phrases independently, with no way to indicate the shared portion of their

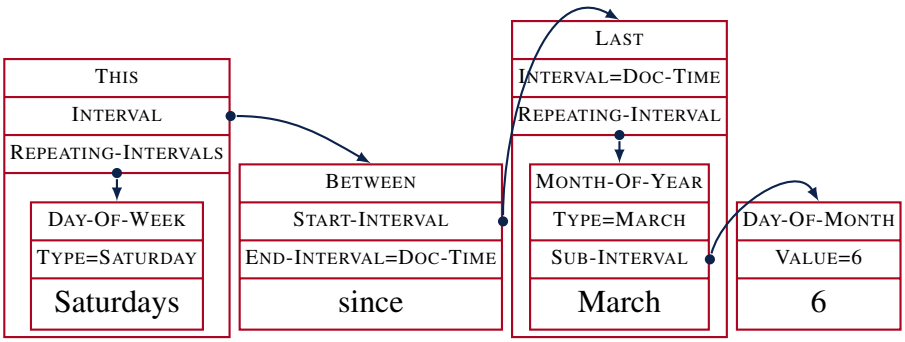

Figure 3.1: Annotation of the expression *Saturdays since March 6* following the SCATE schema.

semantics. These drawbacks of ISO-TimeML, especially the lack of compositionality, make the development of machine learning models difficult. Thus, most prior work has taken a rule-based approach, looking up each token of a time expression in a normalization lexicon and then mapping this sequence of lexical entries to the normalized form (Strötgen and Gertz, 2013; Bethard, 2013; Lee et al., 2014; Strötgen and Gertz, 2015).

As an alternative to TimeML, and inspired by previous works, Schilder (2004) and Han and Lavie (2004), Bethard and Parker (2016) proposed Semantically Compositional Annotation of Time Expressions (SCATE)[1]. In the SCATE schema, each time expression is annotated in terms of compositional time entity over intervals on the timeline. An example is shown in Figure 3.1, with every annotation corresponding to a formally defined time entity. For instance, the annotation on top of *since* corresponds to a BETWEEN operator that identifies an interval starting at the most recent *March 6* and ending at the document

[1] https://github.com/bethard/anafora-annotations/blob/master/.schema/timenorm-schema.xml

creation time (DCT). The BETWEEN operator is formally defined as:

$$\text{BETWEEN}([t_1, t_2)\text{: INTERVAL}, \quad [t_3, t_4)\text{: INTERVAL})\text{: INTERVAL} = [t_2, t_3). \tag{3.1}$$

The SCATE schema can represent a wide variety of time expressions, and provides a formal definition of the semantics of each annotation. Unlike TimeML, SCATE uses a graph structure to capture compositional semantics and can represent time expressions that are not expressed with contiguous phrases. The schema also has the advantage that it can be viewed as a semantic parsing task and, consequently, is more suitable for machine-learning approaches. Although SCATE is not based on any knowledge representation languages, as it contains all temporal concepts, their attributes, mutual relationships, and logical rules, it can in fact be understood as a formal representation of the conceptualization underlying the Temporal Normalization task. Thus in this work, we adopt SCATE scheme as our ontology, and the goal of temporal normalization is to present the extracted time expression using SCATE schema.

3.1.1 An interval-based evaluation metric for normalized times

Before attempting to construct machine-learned models from the SCATE corpus, we were interested in evaluating Bethard and Parker (2016)'s claim that the SCATE schema is able to represent a wider variety of time expressions than TimeML. To do so, we adopt an interval-based evaluation metric for normalized times (Laparra, Xu, and Bethard, 2018) to compare time normalizations annotated in both the ISO 8601 format of TimeML and

```
scala> val today =
     |   ThisRI(
     |     ThisRI(
     |       Year(2017),
     |       RepeatingField(MONTH_OF_YEAR, 4)),
     |     RepeatingField(DAY_OF_MONTH, 21))
scala> val result =
     |   ThisRIs(
     |     Between(
     |       LastRI(
     |         today,
     |         Intersection(Set(
     |           RepeatingField(MONTH_OF_YEAR, 3),
     |           RepeatingField(DAY_OF_MONTH, 6)))),
     |       today),
     |       RepeatingField(DAY_OF_WEEK, 6))
scala> for (Interval(start, end) <- result.intervals)
     |   println(start, end)
(2017-03-11T00:00,2017-03-12T00:00)
(2017-03-18T00:00,2017-03-19T00:00)
(2017-03-25T00:00,2017-03-26T00:00)
(2017-04-01T00:00,2017-04-02T00:00)
(2017-04-08T00:00,2017-04-09T00:00)
(2017-04-15T00:00,2017-04-16T00:00)
(2017-04-22T00:00,2017-04-23T00:00)
...
```

Figure 3.2: Interpretation of *every Saturday since March 6*.

the time entity format of SCATE. This evaluation interprets normalized annotations as intervals along the timeline and measures the overlap of the intervals.

TimeML TIMEX3 (time expression) annotations are converted to intervals following ISO 8601 semantics of their VALUE attribute. So, for example, 1989-03-05 is converted to the interval [1989-03-05T00:00:00, 1989-03-06T00:00:00), that is, the 24-hour period starting at the first second of the day on 1989-03-05 and ending just before the first second of the day on 1989-03-06. SCATE annotations are converted to intervals following the formal semantics of each entity, using the library provided by Bethard and Parker (2016). For example, Next(Year(2010), SimplePeriod(YEARS, 4)), is converted to [2011-01-01T00:00, 2015-01-01T00:00), i.e., the 4 years following 2010. Note that there may be more than one interval associated with a single annotation, as in the *every Saturday*

since March 6 example in Figure 3.2. Once all annotations have been converted into intervals along the timeline, we can calculate the overlap between the intervals of different annotations.

Given two sets of intervals, we define the interval precision, P_{int}, as the total length of the intervals in common between the two sets, divided by the total length of the intervals in the first set. Interval recall, R_{int} is defined as the total length of the intervals in common between the two sets, divided by the total length of the intervals in the second set. Formally:

$$I_S \bigcap I_H = \{i \cap j : i \in I_S \wedge j \in I_H\} \tag{3.2}$$

$$P_{\text{int}}(I_S, I_H) = \frac{\sum_{i \in \text{COMPACT}(I_S \bigcap I_H)} |i|}{\sum_{i \in I_S} |i|} \tag{3.3}$$

$$R_{\text{int}}(I_S, I_H) = \frac{\sum_{i \in \text{COMPACT}(I_S \bigcap I_H)} |i|}{\sum_{i \in \cup I_H} |i|} \tag{3.4}$$

where I_S and I_H are sets of intervals, $i \cap j$ is possibly the empty interval in common between the intervals i and j, $|i|$ is the length of the interval i, and COMPACT takes a set of intervals and merges any overlapping intervals.

Given two sets of annotations (e.g., one each from two time normalization systems), we define the overall precision, P, as the average of interval precisions where each annotation from the first set is paired with all annotations that textually overlap it in the second set. Overall recall is defined as the average of interval recalls where each annotation from the

	AQUAINT	TimeBank	Test
Documents	10	68	20
Sentences	251	1429	339
TimeML timex3	61	499	158
SCATE entities	333	1810	461
SCATE time exp.	114	715	209
SCATE bounded	67	403	93

Table 3.1: Number of documents, TimeML TIMEX3 annotations and SCATE annotations for the subset of the TempEval 2013 corpus annotated with both schemas.

second set is paired with all annotations that textually overlap it in the first set. Formally:

$$\text{OI}_a(B) = \bigcup_{b \in B : \text{OVERLAPS}(a,b)} \text{INTERVALS}(b) \tag{3.5}$$

$$P(S,H) = \frac{1}{|S|} \sum_{s \in S} P_{int}(\text{INTERVALS}(s), \text{OI}_s(H)) \tag{3.6}$$

$$R(S,H) = \frac{1}{|H|} \sum_{h \in H} R_{int}(\text{INTERVALS}(h), \text{OI}_h(S)) \tag{3.7}$$

where S and H are sets of annotations, $\text{INTERVALS}(x)$ gives the time intervals associated with the annotation x, and $\text{OVERLAPS}(a,b)$ decide whether the annotations a and b share at least one character of text in common.

It is important to note that these metrics can be applied only to time expressions that yield bounded intervals. Time expressions that refer to intervals with undefined boundaries are out of the scope, like in “it takes just a minute” or “I work every Saturday”.

3.1.2 Data analysis

TimeML vs. SCATE

Both TimeML and SCATE annotations are available on a subset of the TempEval 2013 corpus (UzZaman et al., 2013) that contains a collection of news articles from different sources, such as Wall Street Journal, New York Times, Cable News Network, and Voices of America. Table 3.1 shows the statistics of the data. Documents from the AQUAINT and TimeBank form the training and development dataset. The SCATE corpus contains 2604 time entities (individual components of a time expression, such as *every*, *month*, *last*, *Monday*, etc.) annotated in the train+dev set (i.e. AQUAINT+TimeBank). These entities compose a total of 1038 time expressions (*every month*, *last Monday*, etc.) of which 580 yield bounded intervals, i.e. intervals with a specified start and ending (*last Monday* is bounded, while *every month* is not).

We apply the interval-based evaluation metric to the AQUAINT and TimeBank datasets, treating the TimeML annotations as the system (S) annotator and the SCATE annotations as the human (H) annotator. Table 3.2 shows that the SCATE annotations cover different time intervals than the TimeML annotations. In the first row, we see that TimeML has a recall of only 92% of the time intervals identified by SCATE in the AQUAINT corpus and of only 83% in the TimeBank corpus. We manually analyzed all places where TimeML and SCATE annotations differed and found that the SCATE interpretation was always the correct one.

	AQUAINT			TimeBank		
	P	R	F1	P	R	F1
Body text	92.2	92.2	92.2	82.4	83.0	82.7
All text	92.2	67.1	77.7	82.4	71.2	76.4

Table 3.2: Comparison of TimeML and SCATE annotations.

For example, a common case where TimeML and SCATE annotations overlap, but are not identical, is time expressions preceded by a preposition like "since". The TimeML annotation for "Since 1985" (with a DCT of 1998-03-01T14:11) only covers the year, "1985", resulting in the time interval [1985-01-01T00:00,1986-01-01T00:00). The SCATE annotation represents the full expression and, consequently, produces the correct time interval [1986-01-01T00:00,1998-03-01T14:11).

Another common case of disagreement is where TimeML failed to compose all pieces of a complex expression. The TimeML annotation for "10:35 a.m. (0735 GMT) Friday" annotates two separate intervals, the time and the day (and ignores "0735 GMT" entirely). The SCATE annotation recognizes this as a description of a single time interval, [1998-08-07T10:35, 1998-08-07T10:36).

TimeML and SCATE annotations also differ in how references to particular past periods are interpreted. For example, TimeML assumes that "last year" and "a year ago" have identical semantics, referring to the most recent calendar year, e.g., if the DCT is 1998-03-04, then they both refer to the interval [1997-01-01T00:00,1998-01-01T00:00). SCATE has the same semantics for "last year", but recognizes that "a year ago" has different semantics: a period centered at one year prior to the DCT. Under SCATE, "a year ago"

Non-operators	Explicit-Operators	Implicit-Operators	Total
1497	305	219	2021
74%	15%	11%	100%

Table 3.3: Distribution of time entity annotations in AQUAINT+TimeBank.

refers to the interval [1996-09-03T00:00,1997-09-03T00:00).

Beyond these differences in interpretation, we also observed that, while the SCATE corpus annotates time expressions anywhere in the document (including in metadata), the TimeBank TIMEX3 annotations are restricted to the main text of the documents. The second row of Table 3.2 shows the evaluation when comparing overall text in the document, not just the body text. Unsurprisingly, TimeML has a lower recall of the time intervals from the SCATE annotations under this evaluation.

Types of SCATE annotations

Studying the training and development portion of the dataset, we noticed that the SCATE annotations can be usefully divided into three categories: non-operators, explicit operators, and implicit operators. We define non-operators as NUMBERS, PERIODS (e.g., *three months*), explicit intervals (e.g., YEARS like *1989*), and repeating intervals (DAY-OF-WEEKS like *Friday*, MONTH-OF-YEARS like *January*, etc.). Non-operators are basically atomic; they can be interpreted without having to refer to other annotations. Operators are not atomic; they can only be interpreted with respect to other annotations they link to. For example, the THIS operator in Figure 3.1 can only be interpreted by first interpreting the DAY-OF-WEEK non-operator and the BETWEEN operator that it links to. We split

operators into two types: explicit and implicit. We define an operator as explicit if it does not overlap with any other annotation. This occurs, for example, when the time connective *since* evokes the BETWEEN operator in Figure 3.1. An operator is considered to be implicit if it overlaps with another annotation. This occurs, for example, with the LAST operator in Figure 3.1, where *March* implies *last March*, but there is no explicit signal in the text, and it must be inferred from context.

We study how these annotation groups distribute in the AQUAINT and TimeBank documents. Table 3.3 shows that non-operators are much more frequent than operators (both explicit and implicit).

3.2 Models

We decompose the normalization of time expressions into two subtasks: a) *time entity identification* which detects the spans of characters that belong to each time expression and labels them with their corresponding time entity; and b) *time entity composition* that links relevant entities together while respecting the entity type constraints imposed by the SCATE schema. These two tasks are run sequentially using the output of the former as input to the latter. Once identification and composition steps are completed we can use the final product, i.e. semantic compositional of time entities, to feed the SCATE interpreter[2] and encode time intervals.

[2] `https://github.com/clulab/timenorm`

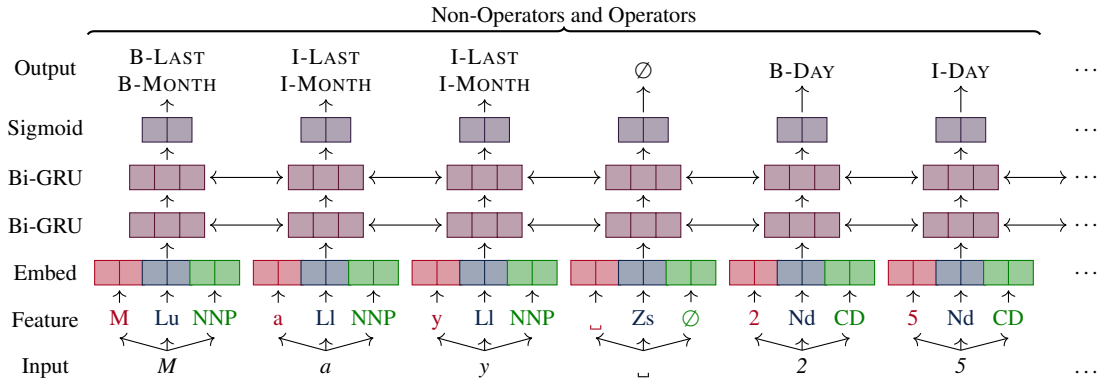

Figure 3.3: Architecture of the 1-Sigmoid model. The input is *May 25*. In SCATE-style annotation, *May* is a MONTH-OF-YEAR (a non-operator), with an implicit LAST (an operator) over the same span, and *25* is a DAY-OF-MONTH. At the feature layer, *M* is an uppercase letter (Lu), *a* and *y* are lowercase letters (Ll), space is a separator (Zs), and *May* is a proper noun (NNP).

3.2.1 Time entity identification

Time entity identification is a type of sequence tagging task where each piece of a time expression is assigned a label that identifies the time entity that it evokes. We express such labels using the BIO tagging system, where B stands for the beginning of an annotation, I for the inside, and O for outside any annotation. Differing somewhat from standard sequence tagging tasks, the SCATE schema allows multiple annotations over the same span of text (e.g., "Saturdays" in Figure 3.1 is both a DAY-OF-WEEK and a THIS), so entity identification models must be able to handle such multi-label classification.

Neural architectures

Recurrent neural networks (RNN) are the state-of-the-art on sequence tagging tasks (Lample et al., 2016; Graves, Mohamed, and Hinton, 2013; Plank, Søgaard, and Goldberg,

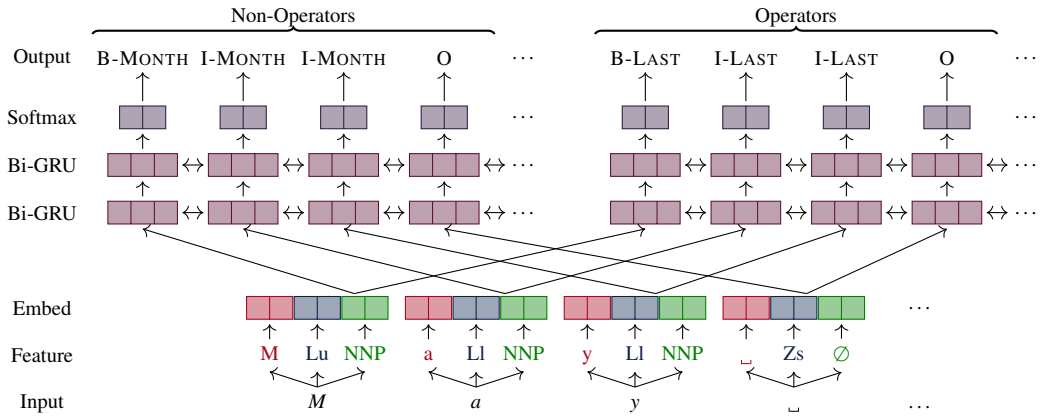

Figure 3.4: Architecture of the 2-Softmax model. The input is *May*. The SCATE annotations and features are the same as in Figure 3.3.

2016) thanks to their ability to maintain a memory of the sequence as they read it and make predictions conditioned on long distance features, so we also adopt them here. We introduce three RNN architectures that share a similar internal structure, but differ in how they represent the output. They convert the input into features that feed an embedding layer. The embedded feature vectors are then fed into two stacked bidirectional Gated Recurrent Units (GRUs), and the second GRU followed by an activation function, outputs one BIO tag for each input. We select GRU for our models as they can outperform another popular recurrent unit LSTM (Long Short Term Memory), in terms of parameter updates and convergence in CPU time with the same number of parameters (Chung et al., 2014).

Our 1-Sigmoid model (Figure 3.3) approaches the task as a multi-label classification problem, with a set of sigmoids for each output that allow zero or more BIO labels to be predicted simultaneously. This is the standard way of encoding multi-label classification problems for neural networks, but in our experiments, we found that these models perform poorly since they can overproduce labels for each input, e.g., *03* could be labeled with

both DAY-OF-MONTH and MONTH-OF-YEAR at the same time.

Our 2-Softmax model (Figure 3.4) splits the output space of labels into two sets: non-operators and operators (as defined in Section 3.1.2). It is very unlikely that any piece of text will be annotated with more than one non-operator or with more than one operator,[3] though it is common for text to be annotated with one non-operator and one operator (see Figure 3.1). As a result, we can use two softmaxes, one for non-operators and one for operators, and the 2-Softmax model thus can produce 0, 1, or 2 labels per input. We share input and embedding layers, but associate a separate set of stacked Bi-GRUs with each output category[4], as shown in Figure 3.4.

Our 3-Softmax further splits operators into explicit operators and implicit operators (again, as defined in Section 3.1.2). We expect this to help the model since the learning task is very different for these two cases: with explicit operators, the model just has to memorize which phrases evoke which operators, while with implicit operators, the model has to learn to infer an operator from context (verb tense, etc.). We use three softmaxes, one each for non-operators, explicit operators, and implicit operators, and, as with 2-Softmax, we share input and embedding layers, but associate a separate set of stacked Bi-GRUs with each output category. The model looks similar to Figure 3.4, but with three output groups instead of two.

We feed three features as input to the RNNs:

[3] In the training data, only 4 of 1217 non-operators overlap with another non-operator, and only 6 of 406 operators overlap with another operator. For example, a *NYT said in an editorial on Saturday, April 25*, *Saturday* is labeled as [DAY-OF-WEEK, LAST, INTERSECTION] where the last two labels are operators.

[4] In preliminary experiments, we tried sharing GRU layers as well, but this generally resulted in worse performance.

Text: The input word itself for the word-by-word model, or a the single input character for the character-by-character model.

Unicode character categories: The category of each character as defined by the Unicode standard.[5] This encodes information like the presence of uppercase (Lu) or lowercase (Ll) letters, punctuation (Po), digits (Nd), etc. For the word-by-word model, we concatenate the character categories of all characters in the word (e.g., *May* becomes LuLlLl).

Part-of-speech: The part-of-speech as determined by the Stanford POS tagger Toutanova et al., 2003. We expect this to be useful for, e.g., finding verb tense to help distinguish between implicit LAST and NEXT operators. For the character-by-character model, we repeat the word-level part-of-speech tag for each character in the word, and characters with no part-of-speech (e.g., spaces) get no tag.

Input: words vs. characters

Identifying SCATE-style time entity is a sequence tagging task, similar to named entity recognition (NER), so we take inspiration from recent work in neural architectures for NER. The first neural NER models followed the prior (non-neural) work in approaching NER as a word classification problem, applying architectures such as sliding-window feedforward neural networks (Qi et al., 2009), convolutional neural networks (CNNs) with conditional random field (CRF) layers (Collobert et al., 2011), and LSTM with CRF layers and hand-crafted features (Huang, Xu, and Yu, 2015). More recently, character-level

[5] See `http://unicode.org/notes/tn36/`

neural networks have also been proposed for NER, including several which combine a CNN or LSTM for learning character-based representations of words with an LSTM or LSTM-CRF for word-by-word labeling (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016), as well as character-by-character sequence-to-sequence networks (Gillick et al., 2016; Kuru, Can, and Yuret, 2016).

Based on these works, we consider two forms of input processing for our RNNs: word-by-word vs. character-by-character. Several aspects of the time normalization problem make the character-based approach especially appealing. First, many time phrases involve numbers that must be interpreted semantically (e.g., a good model should learn that months cannot be a number higher than 12), and digit-by-digit processing of numbers allows such interpretations, while treating each number as a word would result in a sparse, intractable learning problem. Second, word-based models assume that we know how to tokenize the text into words, but at times present challenging formats such as *overnight*, where *over* evokes a LAST operator and *night* is a PART-OF-DAY. Finally, character-based models can ameliorate out-of-vocabulary (OOV) words, which are a common problem when training sparse datasets. (Hybrid word-character models, such as the LSTM-CNNs-CRF (Ma and Hovy, 2016) can address this last problem, but not the previous two.)

For our word-based model, we apply the NLTK Tokenizer (Bird, Klein, and Loper, 2009) to each sentence. We further tokenize with the regular expression `"\d+|[^\d\W]+|\S"` to break apart alpha-numeric expressions like *1620EDT*. However, the tokenizer is unable to break-apart expressions such as *19980206* and *overnight*.

For our character-based model, no tokenization is applied and every character (including whitespace characters) is fed as input.

3.2.2 Time entity composition

Once the entities of the time-expressions are identified, they must be composed in order to obtain their semantic interpretation. This step of the analysis consists of two parts: linking the entities that make up a time-expression together and completing the entities' properties with the proper values. For both cases, we set a simple set of rules that follow the constraints imposed by the SCATE schema[6].

Time entity linking

Algorithm 1 shows the process followed to obtain the links between the time entities. First, we define an empty stack that will store the entities belonging to the same time expression. Then, we iterate over the list of entities of a document sorted by their starting character offsets (SORTBYSTART). For each of these entities ($entity_1$) and for each entity in the stack ($entity_2$), we check if the guidelines specify a possible link (LINKISVALID) between the types of $entity_1$ and $entity_2$. If such a link is possible, and it has not already been filled by another annotation, we greedily make the link (CREATELINK). When the distance in the number of characters between the entity and the end of the stack is bigger than 10, we assume that the entities do not belong to the time expression. Thus, we empty the stack.[7]

[6] `https://github.com/bethard/anafora-annotations/blob/master/.schema/timenorm-schema.xml`

[7] The distance threshold was selected based on the performance on the development dataset.

Algorithm 1 Linking time entities

```
stack = ∅
for entity_1 in SORTBYSTART(entities) do
  if START(entity_1) - END(stack) > 10 then
    stack = ∅
  end if
  for entity_2 in stack do
    if LINKISVALID(entity_1, entity_2) then
      CREATELINK(entity_1, entity_2)
    end if
  end for
  PUSH(stack, entity_1)
end for
```

For example, our time entity identification model gets the YEAR, MONTH-OF-YEAR and DAY-OF-MONTH for the time-expression *1992-12-23*. Our time entity composition algorithm then iterates over these entities. At the beginning the stack is empty, it just pushes the entity *1992* (YEAR) into the stack. For the entity *12* (MONTH-OF-YEAR) it checks if the guidelines define a possible link between this entity type and the one currently in the stack (YEAR). In this case, the guidelines establish that a YEAR can have a SUB-INTERVAL link to a SEASON-OF-YEAR, a MONTH-OF-YEAR or WEEK-OF-YEAR. Thus, the algorithm creates a SUB-INTERVAL link between *1992* and *12*. The entity *12* is then pushed into the stack. This process is repeated for the entity *23* (DAY-OF-MONTH) checking if there was a possible link to the entities in the stack (*1992*, *12*). The guidelines define a possible SUB-INTERVAL link between MONTH-OF-YEAR and DAY-OF-MONTH, so a link is created here as well. Now, suppose that the following time entity in the list is several words ahead of *23* so the character distance between both entities is larger than 10. If that is the case the stack is empty and the process starts again to compose a new time

expression.

Property completion

The last step is to associate each time entity of a time-expression with a set of properties that include information needed for its interpretation. Our system decides the value of these properties as follows:

TYPE: The SCATE schema defines that some entities can only have specific values. For example, a SEASON-OF-YEAR can only be SPRING, SUMMER, FALL or WINTER, a MONTH-OF-YEAR can only be JANUARY, FEBRUARY, MARCH, etc. To complete this property we take the text span of the time entity and normalize it to the values accepted in the schema. For example, if the span of a MONTH-OF-YEAR entity was the numeric value *01* we would normalize it to JANUARY, if its span was *Sep.* we would normalize it to SEPTEMBER, and so on.

VALUE: This property contains the value of a numerical entity, like DAY-OF-MONTH or HOUR-OF-DAY.To complete it, we just take the text span of the entity and convert it to an integer. If it is written in words instead of digits (e.g., *nineteen* instead of *19*), we apply a simple grammar[8] to convert to an integer.

SEMANTICS: In news-style texts, it is common that expressions like *last Friday*, when the DCT is a Friday, refer to the day as the DCT instead of the previous occurrence (as it would in more standard usage of *last*). SCATE indicates this with the SEMANTICS

[8] `https://github.com/ghewgill/text2num`

property, where the value INTERVAL-INCLUDED indicates that the current interval is included when calculating the last or next occurrence. For the rest of the cases the value INTERVAL-NOT-INCLUDED is used. In our system, when a LAST operator is found, if it is linked to a DAY-OF-WEEK (e.g. *Friday*) that matches the DCT, we set the value of this property to INTERVAL-INCLUDED.

INTERVAL-TYPE: Operators like NEXT or LAST need an interval as reference in order to be interpreted. Normally, this reference is the DCT. For example, *next week* refers to the week following the DCT, and in such a case the value of the property INTERVAL-TYPE for the operator NEXT would be DOCTIME. However, sometimes the operator is linked to an interval that serves as reference by itself, for example, "by the year 2000". In this cases the value of the INTERVAL-TYPE is LINK. Our system sets the value of this property to LINK if the operator is linked to a YEAR and DOCTIME otherwise. This is a very coarse heuristic; finding the proper anchor for a time expression is a challenging open problem for which future research is needed.

3.2.3 Automatically generated training data

Every document in the dataset starts with a document creation time. These time expressions are quite particular; they occur in isolation and not within the context of a sentence and they always yield a bounded interval. Thus their identification is a critical factor in an interval based evaluation metric. However, document times appear in many different formats: "Monday, July-24, 2017", "07/24/17 09:52 AM", "08-15-17 1337 PM", etc. Many

of these formats are not covered in the training data, which is drawn from a small number of news sources, each of which uses only a single format. We therefore designed a time generator to randomly generate an extra 800 isolated training examples for a wide variety of such expression formats. The generator covers 33 different formats[9] which include variants covering abbreviation, with/without delimiters, mixture of digits and strings, and different sequences of time units.

3.3 Experiments

We train and evaluate our models on the SCATE corpus described in Section 3.1.2. As a development dataset, 14 documents are taken as a random stratified sample from the TempEval 2013 (TimeBank + AQUAINT) portion shown in Table 3.1, including broadcast news documents (1 ABC, 1 CNN, 1 PRI, 1 VOA), and newswire documents (5 AP, 1 NYT, 4 WSJ). We use the interval-based evaluation metric described in Section 3.1.1, but also report more traditional information extraction metrics (precision, recall, and F_1) for the time entity identification and composition steps. Let S be the set of items predicted by the system and H is the set of items produced by the humans, precision (P), recall (R), and F_1

[9] We use the common formats available in office suites, specifically, LibreOffice.

are defined as:

$$P(S,H) = \frac{|S \cap H|}{|S|}$$
$$R(S,H) = \frac{|S \cap H|}{|H|}$$
$$F_1(S,H) = \frac{2 \cdot P(S,H) \cdot R(S,H)}{P(S,H) + R(S,H)}$$

For these calculations, each item is an annotation, and one annotation is considered equal to another if it has the same character span (offsets), type, and properties (with the definition applying recursively for properties that point to other annotations).

To make the experiments with different neural architectures comparable, we tuned the parameters of all models to achieve the best performance on the development data. We only list here the hyper-parameters for our best Char 3-Softmax: the embedding size of the character-level text, word-level text, POS tag, and unicode character category features are 128, 300, 32 and 64, respectively. To avoid overfitting, we used dropout with probabilities 0.25, 0.15 and 0.15 for the 3 features, respectively; the sizes of the first and second layer GRU units are set as 256 and 150. We trained the model with RMSProp optimization on mini-batches of size 120, and followed standard recommendations to leave the optimizer hyperparameter settings at their default values. Each model is trained for at most 800 epochs, the longest training time for Char 3-Softmax model is around 22 hours using 2x NVIDIA Kepler K20X GPU.

Model	P	R	F1
Word 1-Sigmoid	60.2	52.0	55.8
Char 1-Sigmoid	54.0	59.0	56.4
Word 2-Softmax	58.7	63.9	61.2
Char 2-Softmax	74.8	72.4	73.6
Word 3-Softmax	68.3	64.9	66.6
Char 3-Softmax	88.2	76.1	81.7
Char 3-Softmax extra	80.6	73.4	76.8

Table 3.4: Precision (P), recall (R), and F_1 for the different neural network architectures on *Time entity identification* on the development data.

	Char 3-Softmax			Char 3-Soft. extra		
	P	R	F_1	P	R	F_1
Non-Op	79.2	63.2	70.3	87.4	63.2	73.4
Exp-Op	52.6	36.6	43.2	39.8	38.7	39.3
Imp-Op	53.3	47.1	50.0	65.4	50.0	56.7
Ident	70.0	54.5	61.3	69.4	55.3	61.5
Comp	59.7	46.5	52.3	57.7	46.0	51.2

Table 3.5: Results on the test set for *Time entity identification* (Ident) and *Time entity composition* (Comp) steps. For the former we report the performances for each entity set: non-operators (Non-Op), explicit operators (Exp-Op) and implicit operators (Imp-Op).

3.3.1 Model selection

We compare the different time entity identification models described in Section 3.2.1, training them on the training data, and evaluating them on the development data. Among the epochs of each model, we select the epoch based on the output(s) which the model is good at predicting because based on its weakness, the model would yield unstable results in our preliminary experiments. For example, for 3-Softmax models, our selections rely on the performances of non-operators and implicit-operators. Table 3.4 shows the results of the development phase.

First, we find that the character-based models outperform the word-based models[10]. For example, the best character-based model achieves the F_1 of 81.7 (Char 3-Softmax),which is significantly better than the best word-based model achieving the F_1 of only 66.6 (p=0)[11]. Second, we find that Softmax models outperform Sigmoid models. For example, the Char 3-Softmax model achives the F_1 of 81.7, significantly better than 56.4 F_1 of the Char 1-Sigmoid model (p=0). Third, for both character- and word-based models, we find that 3-Softmax significantly outperforms 2-Softmax: the Char 3-Softmax F_1 of 81.7 is better than the Char 2-Softmax F_1 of 73.6 (p=0) and the Word 3-Softmax F_1 of 66.6 is better than the Word 2-Softmax F_1 of 61.2 (p=0.0254). Additionally, we find that all models are better at identifying non-operators than operators and that the explicit operators are the hardest to solve. For example, the Char 3-Softmax model gets 92.4 F_1 for non-operators, 36.1 F_1 for explicit operators and 79.1 F_1 for implicit operators. Finally, we also train the best model, Char 3-Softmax, using the generated annotations described in Section 3.2.3 and achieve 76.8 F_1 (Char 3-Softmax extra), i.e., the model performs better without the extra data (p=0). This is probably a result of overfitting due to the small variety of time formats in the training and development data.

From this analysis on the development set, we select two variants of the Char 3-softmax architecture for evaluation on the test set: Char 3-Softmax and Char 3-Softmax extra. These models were then coupled with the rule-based linking system described in Section 3.2.2 to

[10]We briefly explored using pre-trained word embeddings to try to improve the performance of the Word 1-Sigmoid model, but it yielded a performance that was still worse than the character-based model, so we didn't explore it further.

[11]We used a paired bootstrap resampling significance test.

Model	P	R	F_1
HeidelTime	70.9	76.8	73.7
Char 3-Softmax	73.8	62.4	67.6
Char 3-Softmax extra	82.7	71.0	76.4

Table 3.6: Precision (P), recall (R), and F_1 of our models on the test data producing bounded time intervals. For comparison, we include the results obtained by HeidelTime.

produce a complete SCATE-style parsing system.

3.3.2 Model evaluation

We evaluate both Char 3-Softmax and Char 3-Softmax extra on the test set for identification and composition tasks. Table 3.5 shows the results. On the identification task, Char 3-Softmax extra is no worse than using the original dataset with the overall F_1 61.5 v 61.3 (p=0.5899), and using extra generated data the model is better at predicting non-operators and implicit operators with higher precisions (p=0.0096), which is the key to produce correct bounded time intervals.

To compare our approach with the state-of-the-art, we run HeidelTime on the test documents and make use of the metric described in Section 3.1.1. This way, we can compare the intervals produced by both systems no matter the annotation schema. Table 3.6 shows that our model with additional randomly generated training data outperforms HeidelTime in terms of Precision, with a significant difference of 12.6 percentage points (p=0.011), while HeidelTime obtains a non-significant better performance in terms of Recall (p=0.1826). Overall, our model gets 3.3 more percentage points than HeidelTime in terms of F_1 (p=0.2485). Notice that, although the model trained without extra annotations

Model	P	R	F_1
HeidelTime	70.7	80.2	75.1
Char 3-Softmax	74.3	64.2	68.9
Char 3-Softmax extra	83.3	74.1	78.4

Table 3.7: Precision (P), recall (R), and F_1 on bounded intervals on the *TimeML/SCATE perfect overlapping* test data.

is better in time entity composition (see Table 3.5), it performs much worse at producing final intervals. This is caused by the fact that this model fails to identify the non-operators that compound dates in unseen formats (see Section 3.2.3).

However, evaluating HeidelTime in the SCATE annotations may not be totally fair. HeidelTime was developed following the TimeML schema and, as we show in Section 3.1.2, SCATE covers a wider set of time expressions. For this reason, we perform an additional evaluation. First, we compare the annotations in the test set using our interval-based metric, similar to the comparison reported in Table 3.2, and select those cases where TimeML and SCATE match perfectly. Then, we remove the rest of the cases from the test set. Consequently, we also remove the predictions given by the systems, both ours and HeidelTime, for those instances. Finally, we run the interval scorer using the new configuration. As can be seen in Table 3.7 all the models improve their performances. However, our model still performs better when it is trained with the extra annotations.

The SCATE interpreter that encodes the time intervals needs the compositional graph of a time-expression to have all its elements correct. Thus, failing in the identification of any entity of a time-expression results in totally uninterpretable graphs. For example, in the expression *next year*, if our model identifies *year* as a PERIOD instead of an

INTERVAL it cannot be linked to *next* because it violates the SCATE schema. The model can also fail in the recognition of some time-entities, like *summer* in the expression *last summer*. This identification errors are caused mainly by the sparse training data. As graphs containing these errors produce unsolvable logical formulae, the interpreter cannot produce intervals and hence the recall decreases. Within those intervals that are ultimately generated, the most common mistake is to confuse the LAST and NEXT operators, and as a result an incorrectly placed interval even with correctly identified non-operators. For example, if an *October* with an implicit NEXT operator is instead given a LAST operator, instead of referring to [2013-10-01T00:00, 2013-11-01T00:00), it will refer to [2012-10-01T00:00, 2012-11-01T00:00). Missing implicit operators is also the main source of errors for HeidelTime, which fails with complex compositional graphs. For example, *that January day in 2011* is annotated by HeidelTime as two different intervals, corresponding respectively to *January* and *2011*. As a consequence, HeidelTime predicts not one but two incorrect intervals, affecting its precision.

3.4 Discussion

As for the time entity identification task, the performance differences between development and test dataset could be attributed to the annotation distributions of the datasets. For example, there are 10 Season-Of-Year annotations in the test set while there are no such annotations in the development dataset; the relative frequencies of the annotations Minute-Of-Hour, Hour-Of-Day, Two-Digit-Year and Time-Zone in the test set are much lower, and

our models are good at predicting such annotations. Explicit operators are very lexically-dependent, e.g. LAST corresponds to one word from the set {*last*, *latest*, *previously*, *recently*, *past*, *over*, *recent*, *earlier*, *the past*, *before*}, and the majority of them appear once or twice in the training and development sets.

Our experiments verify the advantages of character-based-models in predicting SCATE annotations, which are in agreement with our explanations in Section 3.2.1: word-based-models tend to fail to distinguish numbers from digit-based time expressions. It's difficult for word-based-models to catch some patterns of time expressions, such as *24th* and *25th*, *August* and *Aug.*, etc., while character-based models are robust to such variance. We ran an experiment to see whether these benefits were unique to compositional annotations like those of SCATE, or more generally to simply recognizing time expressions. We used the TimeML annotations from AQUAINT and TimeBank (see Table 3.1) to train two multi-class classifiers to identify TIMEX3 annotations. The models were similar to our Char 3-Softmax and Word 3-Softmax models, using the same parameter settings, but with a single softmax output layer to predict the four types of TIMEX3: DATE, TIME, DURATION, and SET. As shown in Table 3.8, on the test set the word-based model significantly outperforms the character-based model in terms of both time expressions (p=0.0428) and the subset of time expressions that contain digits (p=0.0007). These results suggest that the reason character-based models are more successful on the SCATE annotations is that SCATE breaks time expressions down into meaningful sub-components. For example, TimeML would simply call *Monday, 1992-05-04* a DATE, and call *15:00:00*

	TIMEX3			TIMEX3-Digits		
	P	R	F_1	P	R	F_1
Char	70.2	62.7	66.2	73.8	71.4	72.6
Word	81.3	69.0	74.7	86.2	79.4	82.6

Table 3.8: Precision (P), recall (R), and F_1 for character-based and word-based models in predicting TimeML TIMEX3 annotations on the TempEval 2013 test set. TIMEX3-Digits is the subset of annotations that contain digits.

GMT Saturday a TIME. SCATE would identify four and five, respectively, different types of semantic entities in these expression; and each SCATE entity would be either all letters or all digits. In TimeML, the model is faced with difficult learning tasks, e.g., that sometimes a weekday name is part of a DATE and sometimes it is part of a TIME, while in SCATE, a weekday name is always a DAY-OF-WEEK.

On the other hand, running the entity composition step with gold entity identification achieves 72.6 in terms of F1. One of the main causes of errors in this step is the heuristic to complete the INTERVAL-TYPE property. As we explain in Section 3.2.2, we implement a too coarse set of rules for this case. Another source of errors is the distance of the 10 characters we use to decide if the time entities belong to the same time expression. This condition prevents the creation of some links, for example, the expression “Later” at the beginning of a sentence typically refers to another time interval in a previous sentence, so the distance between them is much longer.

CHAPTER 4

Contextualized Character Embeddings for Temporal Normalization

In the previous chapter, we have described how we approach temporal normalization task and shown the advantages of character-based multi-output neural network models. This chapter presents our advances in using pre-trained contextualized character embeddings in the time entity identification task. We first explain why we apply pre-trained contextualized character embeddings to temporal normalization in Section 4.1, and then describe our experiment settings in Section 4.2. Next, we present the experimental results on two datasets from SemEval 2018 Task 6 (Laparra et al., 2018) in Section 4.3 and we conduct an in-depth analysis of the neural models in Section 4.4 to understand better where the improvements come from and what they depend on. Finally, we discuss the limitations of our work on time normalization and future research directions in Section 4.5.

4.1 Background and motivation

Pre-trained language models (LMs) such as ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), OpenAI GPT (Radford et al., 2018), Flair (Akbik, Blythe, and Vollgraf, 2018) and Bert (Devlin et al., 2019) have shown great improvements in NLP tasks ranging from sentiment analysis to named entity recognition to question answering. These models are trained on huge collections of unlabeled data and produce contextualized word em-

beddings, i.e., each word receives a different vector representation in each context, rather than a single common vector representation regardless of context as in word2vec (Mikolov et al., 2013) and GloVe (Pennington, Socher, and Manning, 2014).

Research is ongoing to study these models and determine where their benefits are coming from (Peters et al., 2018; Radford et al., 2018; Khandelwal et al., 2018; Qi et al., 2018; Zhang and Bowman, 2018). The analyses have focused on word-level models, yet character-level models have been shown to outperform word-level models in some NLP tasks, such as text classification (Zhang, Zhao, and LeCun, 2015), named entity recognition (Kuru, Can, and Yuret, 2016), and our time normalization system (Laparra, Xu, and Bethard, 2018). Thus, there is a need to study pre-trained contextualized *character* embeddings, to see if they also yield improvements, and if so, to analyze where those benefits are coming from.

All of the pre-trained word-level contextual embedding models include some character or subword components in their architecture. For example, Flair is a forward-backward LM trained over characters using recurrent neural networks (RNNs), that generates pre-trained contextual word embeddings by concatenating the forward LM's hidden state for the word's last character and the backward LM's hidden state for the word's first character. Flair achieves state-of-the-art or competitive results on part-of-speech tagging and named entity tagging (Akbik, Blythe, and Vollgraf, 2018). Though they do not pre-train a LM, Bohnet et al. (2018) similarly apply a bidirectional long short term memory network (LSTM) layer on all characters of a sentence and generate contextual word embeddings by

	Newswire			Clinical		
	Train	Dev	Test	Train	Dev	Test
Documents	64	14	20	232	35	141
SCATE entities	1,628	402	398	14,936	2,896	9,530
SCATE time exp.	636	146	186	4,469	879	2,815
SCATE bounded	391	80	93	2,303	430	1,471

Table 4.1: Number of documents and SCATE annotations for newswire and clinical domains of the corpus following the SCATE schema.

concatenating the forward and backward LSTM hidden states of the first and last character in each word. Together with other techniques, they achieve state-of-the-art performance on part-of-speech and morphological tagging. However, both Akbik, Blythe, and Vollgraf (2018) and Bohnet et al. (2018) discard all other contextual character embeddings, and no analyses of the models are performed at the character-level.

In this work, we derive pre-trained contextual character embeddings from Flair's forward-backward LM trained on a 1-billion word corpus of English (Chelba et al., 2014), and observe if these embeddings yield the same large improvements for our time normalization task as yielded by pre-trained contextual word embeddings for word-level tasks. We aim to analyze where improvements come from (e.g., term variations, low frequency words) and what they depend on (e.g., embedding size, context size).

4.2 Experiment

We keep the original neural architecture and parameter settings in our 3-softmax model and experiment with the following embedding layers:

Rand(128): the original setting of our character-based neural network, where 128-

dimensional character embeddings are randomly initialized.

Rand(4096): 4096-dimensional character embeddings are randomly initialized, matching the dimensionality of the Flair forward-backward LM hidden states, i.e., matching the dimensionality of Cont(4096).

Cont(4096): 4096-dimensional pre-trained contextual character embeddings are derived by running Flair forward-backward character-level LM over the text, and concatenating the hidden states from forward and backward character-level LMs for each character.

In addition to the newswire domain corpus (shown at Table 3.1), we further evaluate our models on a subset of the THYME corpus used in the Clinical TempEvals (Bethard et al., 2015; Bethard et al., 2016; Bethard et al., 2017), which includes a set of de-identified clinical notes and pathology reports from cancer patients at the Mayo Clinic. Table 4.1 shows the statistics of these two copora, the clinical domain being more than 9 times larger and the newswire domain having a more diverse set of labels. We use the same evaluation metrics as shown in section 3.3.

4.3 Results

Table 4.2 shows that the model using pre-trained contextual character embeddings, Cont(4096), outperforms our previous best time normalization system on all three metrics: identification of time entities, parsing, and interval extraction. For identification, our

[0]We upgraded Keras from 1.2 to 2.1 and fixed a code bug that allowed predictions to be made on padding tokens.

Model	Domain	Identification	Parsing	Interval Extraction
Rand(128)-ori	News	61.5	51.2	76.4
Rand(128)	News	59.4	50.5	64.6
Rand(4096)	News	64.8	54.1	68.2
Cont(4096)	News	**80.3**	**66.8**	**81.5**
Rand(128)-ori	Clinical	84.7	57.9	72.1
Rand(128)	Clinical	92.8	65.3	82.1
Rand(4096)	Clinical	93.2	65.3	83.8
Cont(4096)	Clinical	**95.2**	**67.3**	**85.8**

Table 4.2: Results on Identification (Ident.), Parsing and Interval extraction (Interv.) of time expressions for News and Clinical domain. Rand(128)-ori refers to the original implementation of 3-softmax in the last chapter, and Rand(128) and Cont(4096) refer to our re-implementation of 3-softmax in this chapter.

	Domain	Dev	Test
Rand(128)	News	76.5	59.4
Rand(4096)	News	82.7	64.8
Cont(4096)	News	**87.4**	**80.3**
Rand(128)	Clinical	92.9	92.8
Rand(4096)	Clinical	92.6	93.2
Cont(4096)	Clinical	**94.7**	**95.2**

Table 4.3: Performance (F_1) of time entity identification.

primary focus as we are only modifying the identification model - 3-softmax, Cont(4096) reduces error by 51% (59.4 to 80.3 F_1) on news, and by 33% (92.8 to 95.2 F_1) on clinical notes. For the following experiments, we only use the identification metric to evaluate the performance.

		News		Clinical	
		Dev	Test	Dev	Test
Variation	+var	**+8.4**	**+15.0**	+1.2	+1.3
	-var	+1.6	+8.7	+1.2	**+1.4**
Frequency	$\leq$10	**+8.1**	**+17.6**	**+2.0**	**+4.2**
	$>$10	+2.4	+5.0	+1.1	+1.1

Table 4.4: Effect of term variations and frequency: improvement in F_1 of Cont(4096) over Rand(4096).

4.4 Where the improvements come from

4.4.1 Larger character embeddings

Table 4.3 compares different embedding sizes. Moving from random 128-dimensional to random 4096-dimensional embeddings improves the model: Rand(4096) statistically outperforms[1] Rand(128) on news dev ($p = 0.0001$), news test ($p = 0.0291$), and clinical test ($p = 0.0301$), though it is not statistically different on clinical dev ($p = 0.2524$). Pre-trained contextual embeddings provide additional benefits: Cont(4096) significantly outperforms Rand(4096) on all datasets ($p < 0.001$ in all cases). We conclude that pre-trained contextual character embeddings provide more than just greater model capacity.

4.4.2 Robustness to variants and frequency

Table 4.4 shows how pre-trained contextual character embeddings improve performance on both **term variations** and **low frequency words**.

We define **term variations** as time entities that appear in the training data in the

[1] We used a paired bootstrap resampling significance test.

following patterns: both upper-case and lower-case, e.g., *DAY*, *Day*, and *day*; abbreviation with and without punctuation, e.g., *AM* and *A.M.*; or same stem, e.g., *Month* and *Months*, *previously* and *previous*. In the dev and test sets, 30.4-35.6% of entities are term variations. The first 2 rows of Table 4.4 show the performance improvements in F_1 of Cont(4096) over Rand(4096) on time entities with (+var) and without (-var) term variations. Cont(4096) is always better than Rand(4096) so all differences are positive, but the improvements in +var are much larger than those of -var in the news domain (+8.4 vs. +1.6 and +15.0 vs. +8.7). In the clinical domain, where 9 times more training data is available, both +var and -var yield similar improvements. We conclude that pre-trained contextual character embeddings are mostly helpful with term variations in low data scenarios.

We define **infrequent terms** as time entities that occur in the training set 10 or fewer times. In the dev and test sets, 73.9-86.9% of terms are infrequent, with about one third of infrequent terms being numerical[2]. The bottom two rows of Table 4.4 show the improvements in F_1 of the Cont(4096) over Rand(4096) on frequent ($>$10) and infrequent ($\leq$10) terms. Cont(4096) is always better than Rand(4096), and in both domains the improvements on low frequency terms are always greater than those on high frequency terms (+8.1 vs. +2.4 in news dev, +17.6 vs. +5.0 in news test, etc.). We conclude that pre-trained contextual character embeddings improve the representations of low frequency words in both low and high data settings.

[2]Numbers are common in time expressions.

	Train	Target	Dev	Test
Rand(128)	Clinical	News	63.4	65.5
Rand(4096)	Clinical	News	62.6	66.9
Cont(4096)	Clinical	News	**68.3**	**78.5**
Rand(128)	News	Clinical	45.3	46.3
Rand(4096)	News	Clinical	43.8	44.3
Cont(4096)	News	Clinical	**57.1**	**59.5**

Table 4.5: Effect of domain change on performance: (F_1) on News and Clinical datasets.

4.4.3 Robustness to domain differences

To illustrate the ability of pre-trained contextual character embeddings to handle unseen data, we train the models in one domain and evaluate in the other, as shown in Table 4.5. We find that Rand(128) and Rand(4096) achieve similar cross-domain performance, e.g., Rand(128) achieves 63.4% of F_1 on news dev and Rand(4096) achieves 62.6% F_1. But Cont(4096) achieves much better cross-domain performance than Rand(128) or Rand(4096): 78.5% vs. 65.5% or 66.9% F_1 on news test, 59.5% vs. 46.3% or 44.3% on clinical test, etc. All these improvements are significant ($p < 0.001$). We conclude that pre-trained contextual character embeddings generalize better across domains.

4.4.4 Greater reliance on nearby context

Inspired by Khandelwal et al. (2018)'s analysis of the effective context size of a word-based language model, we present an ablation study to measure performance when contextual information is removed. Specifically, when evaluating models, we retain only the characters in a small window around each time entity in the dev and test sets, and replace all other characters with padding characters.

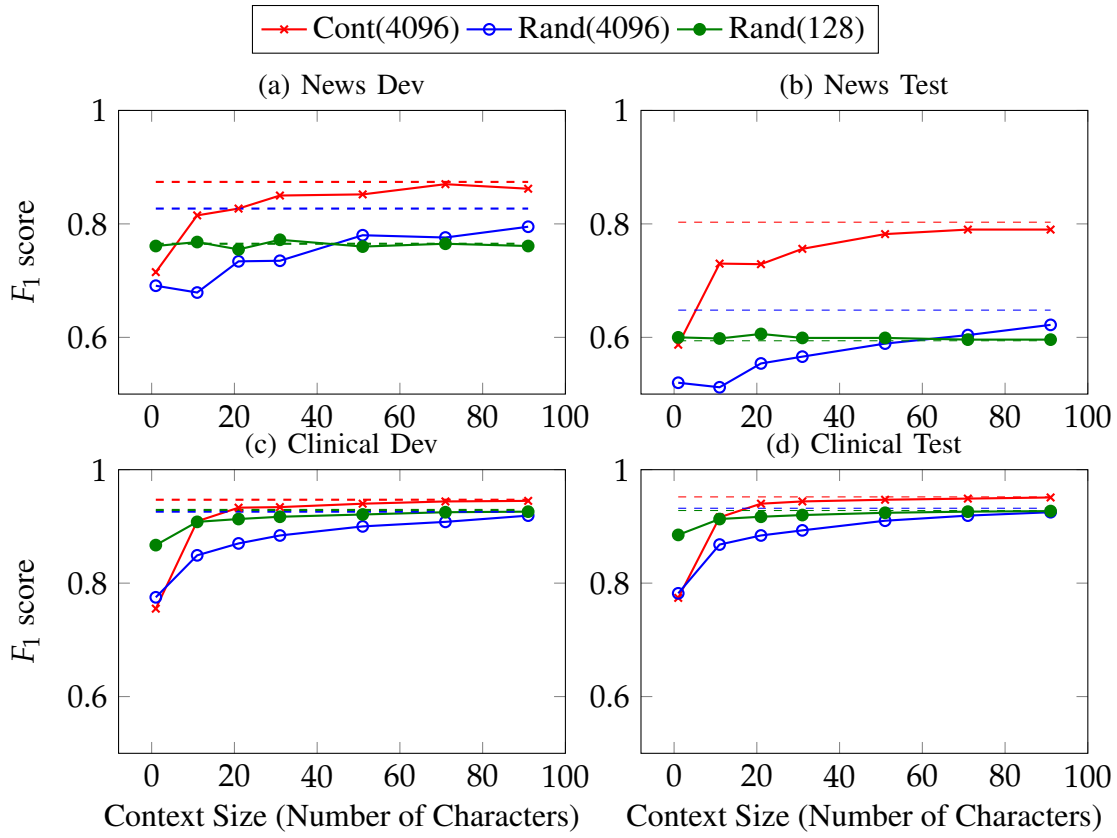

Figure 4.1: Effect of the context information on the performances for Cont(4096), Rand(4096) and Rand(128) on the dev and test sets. The dashed lines are the performances of models using the original context setting.

Figures 4.1a and 4.1b evaluate the Cont(4096), Rand(4096) and Rand(128) models across different context window sizes on the news dev and test set, respectively. Rand(128) performs similarly across all context sizes, suggesting that it makes little use of context information. Both Rand(4096) and Cont(4096) depend heavily of context: without any context information (context size 0), they perform worse than Rand(128). Cont(4096) is sensitive to the nearby context, with a $\sim$10 point gain on news dev and $\sim$15 point gain on news test from just the first 10 characters of context, putting it easily above Rand(128). Rand(4096) doesn't exceed the performance of Rand(128) until at least 50 characters of

		News		Clinical	
	Set	Dev	Test	Dev	Test
Rand(128)	C	73.6	56.1	91.9	92.1
Rand(128)	CUP	76.5	59.4	92.9	92.8
Rand(4096)	C	80.5	62.4	91.7	92.2
Rand(4096)	CUP	82.7	64.8	92.6	93.2
Cont(4096)	C	87.9	78.1	94.7	95.5
Cont(4096)	CUP	87.4	80.3	94.7	95.2

Table 4.6: Effect of features on performance: Performance (F_1) with different feature sets, including characters (C), part-of-speech tags (P), and unicode character categories (U).

context.

Figures 4.1c and 4.1d shows similar trends in the clinical domain, except that the Rand(128) model now shows a small dependence on context, with a $\sim$5 point drop on clinical dev and a $\sim$3 drop on clinical test in the no-context setting. Cont(4096) again makes large improvements in just the first 10 characters, and Rand(4096) now takes close to 100 characters of context to reach the performance of Rand(128). We conclude that pre-trained contextual character embeddings make better use of local context, especially within the first 10 characters.

4.4.5 Encoding word categories

We perform a feature ablation to see if pre-trained contextual character embeddings capture basic syntax (e.g., part-of-speech) like pre-trained contextual word embeddings do (Peters et al., 2018; Akbik, Blythe, and Vollgraf, 2018). Table 4.6 shows that removing both part-of-speech and unicode category features from Cont(4096) does not significantly change performance: news dev ($p = 0.8813$), news test ($p = 0.1672$), clinical dev ($p =$

0.5367), clinical test ($p = 0.8537$). But ablating part-of-speech tags and unicode character categories does decrease performance for both Rand(128) and Rand(4096) in all cases. For example, Rand(4096) with all features achieves 82.7 F_1 on news dev, significantly better than the 80.5 F_1 of using only characters ($p = 0.0467$). We conclude that pre-trained contextual character embeddings encode a variety of word category information such as part-of-speech, capitalization, and punctuation.

4.4.6 Examples of the improvement

We analyzed a few examples where Cont(4096) makes correct predictions, but Rand(4096) does not.

Robustness to variants

"... with year-earlier profit of millions..."

In this sentence, the Cont(4096) model labeled *earlier* correctly, while the Rand(4096) model missed it. In the news training set, *earlier* occurs a few times, but none of them have "-" nearby.

Robustness to frequency

"... in the first days after President..."

In this sentence, the Cont(4096) model labeled *first* correctly, while the Rand(4096) model labeled it incorrectly. In the news training set, *first* only occurred once when followed by another time entity, but there were several similar sentences for *second* and *third* in the

training set.

Robustness to word order

"... until twenty years after the first astronauts..."

"... comes barely a month after Qantas..."

"... Retaliating 13 days after the deadly..."

In each of the sentences above, the Cont(4096) model labeled *after* correctly, while Rand(4096) labeled it incorrectly. In the training set, there were a few examples where *after* occurred near a time entity, but always before the time entity (e.g., *after ten years*, *after 22 months*, *after three days*, *after a 16-hour flight*) rather than after it as in the examples above. Cont(4096) may have learned a better representation for *after* that allows it to be less dependent on exact word order.

4.5 Limitations and future research

Although character-based models have advantages for concept normalization over word-based ones, especially after adopting pre-trained contextualized character embeddings, these advantages are at the cost of running speed. Considering only the sequence length in recurrent neural networks, the average running time of character-based model is around 5 times larger than the word-based one[3]. However, in terms of batch processing, character-based model is even slower as longer sequences take up more processing memory which indicates a smaller batch size.

[3]The average English word length is 5.1 letters (Bochkarev, Shevlyakova, and Solovyev, 2015).

Our application of character-based RNN trained on SCATE-style annotations was in 2017, when none of the pre-trained transformer networks had yet been proposed. We believe pre-trained transformer networks such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019) should also be good choices for time normalization: 1) the word-piece tokenization allows the split of time phrases involving digits such as *2020-10-01* into multiple semantic units; 2) in terms of computational complexity, self-attention layers in transformer networks are faster than recurrent layers in RNN when the sequence length n is smaller than the representation dimensionality d, which is most often the case when word-piece or byte-pair encodings are adopted (Vaswani et al., 2017); and 3) pre-trained transformer networks have shown great performance in many NLP tasks.

CHAPTER 5

Biomedical Concept Normalization

In this chapter, we view biomedical concept normalization as another example of ontology informed information extraction. The goal of biomedical concept normalization is to assign suitable identifiers from ontology to recognized biomedical entities. Such a mapping process encourages the standardization and inter-connections of these entities that are used to represent knowledge about a domain. However, concept normalization is challenging because ontologies are large. In most cases, annotated datasets cover only a small sample of the concepts, yet concept normalizers are expected to predict all concepts in the ontology. This chapter presents a generate-and-rank framework consisting of a candidate generator and a list-wise ranker based on BERT (Devlin et al., 2019). The ranker considers pairings of concept mentions and candidate concepts, allowing it to make predictions for any concept, not just those seen during training.

This chapter is organized as follows. We first give a high-level overview of the biomedical concept normalization task and some notable progress in the past (Section 5.1). Secondly, we introduce a generate-and-rank framework with semantic type regularization for biomedical concept normalization (Section 5.2). We then further describe a comprehensive evaluation on a few biomedical concept normalization datasets(Section 5.3). In particular, we present our submitted system in the 2019 N2C2 Shared Task Track 3 Con-

cept Normalization (the third-highest performance) (Section 5.3.5). Next, we present the experimental results on all datasets (Section 5.4), conduct error analysis of our proposed models on the MCN dataset, and explain how UMLS improve sieve-based generation and BERT-based ranking (Section 5.5). Finally, we discuss current limitations and future directions in Section 5.6.

5.1 Background

Mining and analyzing the constantly-growing unstructured text in the biomedical domain offers great opportunities to advance scientific discovery (Gonzalez et al., 2015; Fleuren and Alkema, 2015) and improve clinical care (Rumshisky et al., 2016; Liu, Jagannatha, and Yu, 2019). However, the frequent use of abbreviations, acronyms, ambiguous terms, aliases, as well as misspellings in such text pose key challenges for data interoperability and the development of NLP techniques. For instance, *heart attack*, *MI*, *myocardial infarction*, and *cardiovascular stroke* all refer to the same concept. It is critical to disambiguate these terms by linking them with their corresponding concepts in an ontology or knowledge base. Such linking allows downstream tasks (relation extraction, information retrieval, text classification, etc.) to access the ontology's rich knowledge about biomedical entities, their synonyms, semantic types, and mutual relationships. Techniques for biomedical concept normalization have been advancing, thanks in part to recent shared tasks including clinical disorder normalization in 2013 ShARe/CLEF (Suominen et al., 2013) and 2014 SemEval Task 7 Analysis of Clinical Text (Pradhan et al., 2014), and adverse drug event normal-

Concept: [C0202194] Potassium measurement

Semantic Type

Laboratory Procedure [T059]

Definition

ALT/null - Analyzing a client/patient's blood, hair, urine and/or other sample to assess his or her potassium levels. Service is billed per test.

NCI/null - A quantitative measurement of the amount of potassium present in a sample.

NCI/null - A measurement of the potassium in a biological specimen.

Synonyms (16)

K
K+
Measurement of potassium
Potassium
Potassium Measurement
Potassium each test
Potassium measurement
Potassium measurement (procedure)
Potassium measurement, NOS
Potassium+
Test;potassium
potassium level test
potassium levels
potassium levels (lab test)
potassium measurement
potassium test

Relations (480) REL | RELA | RSAB| String | CUI

[: 1 - 10 :]

CHD | isa | NCI | Potassium measurement, urine | C0202195
CHD | isa | SCTSPA | Potassium measurement, urine | C0202195
CHD | isa | SNOMEDCT_US | Potassium measurement, urine | C0202195
CHD | isa | CPT | Potassium measurement, urine | C0202195
CHD | isa | NCI | Serum potassium measurement | C0302353
CHD | isa | SCTSPA | Serum potassium measurement | C0302353
CHD | isa | SNOMEDCT_US | Serum potassium measurement | C0302353
CHD | isa | CPT | Serum potassium measurement | C0302353
CHD | isa | MEDCIN | Serum potassium measurement | C0302353
CHD | isa | SNOMEDCT_US | Fluid sample potassium measurement | C0428290

Figure 5.1: The basic view of *potassium measurement* in UMLS.

ization in Social Media Mining for Health (SMM4H) (Sarker et al., 2018; Weissenbacher et al., 2019).

Biomedical concept normalization is very similar to named entity linking (NEL), both aiming to link the in-text natural-language mention to an entry in a structural resource. For NEL, the entity or instance to which the textual mention is linked typically comes from

a knowledge base. A knowledge base[1] typically contains information about the world's instances or entities such as people and organizations, their semantic classes, and their relationships. Some notable examples of knowledge bases include Wikipedia, DBpedia (Auer et al., 2007), YAGO (Fabian, Gjergji, Gerhard, et al., 2007), Freebase (Bollacker et al., 2008), Probase (Wu et al., 2012), etc. For concept normalization in the biomedical domain, ontologies are preferred to knowledge bases. Ontologies used in biomedical concept normalization include but not limited to: Systematized Nomenclature of Medicine –Clinical Terms (SNOMED CT), Medical Subject Headings (MeSH), International Statistical Classification of Diseases and Related Health Problems (ICD), Gene Ontology (GO), the Medical Dictionary for Regulatory Activities (MedDRA), the Unified Medical Language System (UMLS). Figure 5.1 shows the basic view of the concept *potassium measurement* in UMLS, which contains information such as the concept unique identifier (CUI), preferred name, semantic types, definitions, synonyms, and relations. Most of these ontologies fall into the form of vocabularies defined using natural language with tree-like inheritance structures.

Research on NEL (knowledge bases) vs. concept normalization (ontologies) also differs in the domains of the corpora studied. Corpora in entity linking are mostly from webpages (Ji et al., 2010) or newswire articles (Hoffart et al., 2011; Ji et al., 2010), where the contents are easy to understand, and the target audiences do not need to have

[1] An ontology together with a set of individual instances of classes constitutes a knowledge base. An ontology can be viewed as a level of abstraction of data models, while a knowledge base is intended for modeling knowledge about individuals (Gruber, 2009).

professional knowledge. Corpora in biomedical concept normalization are mostly from clinical notes (Suominen et al., 2013; Pradhan et al., 2014), scientific articles (Morgan et al., 2008; Lu et al., 2011; Li et al., 2016a; Doğan, Leaman, and Lu, 2014), and social media posts or medical forum blogs (Karimi et al., 2015; Limsopatham and Collier, 2016). There are remarkable differences among these three different text domains:

Scientific articles: texts in scientific articles are written for professional audiences with fewer lexical variants, and the medical terminologies are used consistently across the document as they are carefully constructed and meticulously proof-read. Besides, the entity mentions are more likely to be in canonical form, although new concepts may be introduced, such as a newly unraveled gene (De Bruijn and Martin, 2002).

Social media posts: online-based texts have been classified as noisy as they pose considerable problems both at the lexical and the syntactic levels (Boiy and Moens, 2009). At the lexical level, jargon, non-standard expressions, misspellings, contractions of existing words/abbreviations, the use of emoticons, and the creation of new words are the norm (Mostafa, 2013), while at the syntactic level, we can hardly speak of a complete real sentence, and the text to be processed always lacks rich context information and even shares no common words with target medical concepts (Luo et al., 2018). For example, social media text *head spinnnnnning a little* is required to translate into *dizziness* in formal medical language. A previous study shows that directly applying MetaMap (a popular biomedical concept normalization tool) on social media healthcare data leads to low precision of 43.75% with all clinical semantic types, and the performances are much lower for

the domain-specific concepts than general concepts (Tu et al., 2016).

Clinical notes: concept normalization using clinical notes is less-studied partially due to the relative scarcity of clinical narrative corpora and the informed consent for accessing personal health data. Most research has moved forward due to the efforts of several groups to provide annotated data through the context of a shared task. Clinical narratives are written by professionals under considerable time pressure, using a combination of ad-hoc formatting, eliding words, and with liberal use of parenthetical expressions, to communicate the status and history of a single patient to other health care professionals or as references. In other words, clinical narratives are written by health care professionals but recorded less formally. One study (Leaman, Khare, and Lu, 2015) shows that the vocabulary used to describe disorders in the clinical text is richer than in scientific articles. One of the biggest challenges in clinical notes is the prevalence of abbreviations, as abbreviations usually do not appear along with their expanded forms. Therefore, approaches based on abbreviation-definition patterns for scientific articles do not apply to the clinical notes. And meanwhile, the abbreviations are also ambiguous, e.g. *RA* could be *right atrium* or *rheumatoid arthritis*. Xu, Stetson, and Friedman (2007) report that 33.1% of abbreviations found in the UMLS were ambiguous.

5.1.1 Challenges of biomedical concept normalization

NEL is challenging due to the name variation and entity ambiguity, and absence of entities in the knowledge base (Rao, McNamee, and Dredze, 2013; Shen, Wang, and Han, 2014).

All these challenges are also common in biomedical concept normalization, but the severity of these challenges varies due to the domain differences. Rather than focusing on noun phrases or named entities as in NEL, biomedical concept normalization also cares about discontinuous words, short texts, adjective phrases. Unlike NEL in the general domain where ambiguity is the primary challenge, mentions in biomedical concept normalization are relatively unambiguous, i.e., it is more common that different words or phrases refer to the same concept than that same word or phrase in the biomedical text refers to multiple different concepts (D'Souza and Ng, 2015; Li et al., 2017; Ji, Wei, and Xu, 2020).

The main challenges of biomedical concept normalization are:

Lexical variants: There exist large lexical gaps between informal expressions of concepts and standard concept names. Concept mentions have great lexical variants, e.g., partial concept names, aliases, alternative spellings, misspellings, abbreviations, acronyms, etc. Concept mentions could be a noun phrase such as *physical examination*, discontinuous tokens *left atrium...dilated* from sentence *the left atrium is moderately dilated*, or even a short sentence *I am wide awake once again after 1:00 a.m* where it shares no common words with its target concept *initial insomnia*. In addition to the lexical variants, concept mentions also have complicated expression forms. For instance, nomenclature systems of medicine require their chemical expressions to be written as characters mixed with digits and/or punctuation marks, e.g., *asprin* also named as *2-Acetoxybenzoic acid*.

Various modifiers: Biomedical concepts are usually constrained or clarified with additional information such as modifier terms to meet the need of precision medicine. Inspired

by the categorization of phenotype character modifiers (Hagedorn, 2007; Endara et al., 2018), we find the similar patterns of modifiers used in concept mentions. Such modifiers include but are not limited to the followings: spatial modifiers such as *right leg pain* and *the left cerebellar infarction*; temporal modifiers indicating how fast or how long a disease or condition persists, e.g., *acute disease* and *chronic ulcer*; quantifiers such as *40% stenosis* and *10% body burns*; frequency modifiers indicating the probability of observing a true statement, e.g., *occasional palpitations* and *sometimes tired*; approximation modifiers indicating the degree of inaccuracy of a reported value, e.g., *slightly overweight*, *malignancy in approximately 10%*, etc.

Rather than clarifying the concepts, some modifiers used in compositional concept mentions such as *quite sedated* and *somewhat tender* could also cause vagueness. These compositional concept mentions could not be mapped to any concepts, and are typically assigned to *CUI-less* indicating that their concepts do not exist in the ontology. Besides the modifiers appearing in the compositional concept mentions, modifiers themselves alone can also be mapped to the concepts in ontologies, e.g., *slight* appearing alone is a qualitative concept in UMLS, which means *small or little in size, quantity, or degree*.

Synonym and ambiguity: Synonym refers to the different terms that can be potentially mapped to the same concept. One popular example for the lexical variant is that one concept has lots of different expressions, e.g., the concept name *sleeplessness* has more than 20 synonyms in UMLS, including *insomnia disorder*, *insomnia NOS*, etc. Ambiguity refers to mentions that can potentially map to multiple concepts. Addressing ambiguity

issues is an important step in natural language processing pipelines designed for information extraction and knowledge discovery. Similar to word sense disambiguation, where words and phrases are disambiguated by their context, ambiguity resolution in biomedical concept normalization also requires the processing of the context information of the concept mention. For example, *potassium* followed by medicine is more likely to be the pharmacologic substance *metallic element of the alkali group; many of whose salts are used in medicine*. While followed by measurements, it belongs to the laboratory procedure *the quantitative measurement of the amount of potassium present in a sample*.

CUI-less: In biomedical concept normalization, some concept mentions could not be mapped to any concepts in an ontology (Suominen et al., 2013; Pradhan et al., 2014; Elhadad et al., 2015), are typically assigned a *CUI-less* label. For example, in SemEval-2014 Task 7 (Pradhan et al., 2014), there are 28% *CUI-less* mentions in the training set. One reason is the incompleteness of the ontology, as the manual maintenance of an ontology is costly and time-consuming, and new facts or knowledge are continuously generated. Another more prevalent cause is that the inclusion of some concepts in the ontology does not follow the principles of developing an ontology. For instance, concept mentions such as *quite sedated* and *somewhat tender* are vague, which are not good for sharing common understandings. And concept mentions such as *CSF labeled tube # 1* are too practical, which belongs to the operational knowledge and is not good for the reuse of domain knowledge. Several recent works (Luo, Sun, and Rumshisky, 2019b; Roberts et al., 2015; Osborne et al., 2018) approach these *CUI-less* mentions by applying compositional

rules.

Post-coordinated concepts: Post-coordinated concepts refer to concepts composed of multiple single concepts from the ontology, in contrast to the pre-coordinated concept, which is explicitly predefined and represented in the ontology. Post-coordinated concepts apply the compositional rules to elucidate a broader range of concepts than is possible with pre-coordinated systems (Osborne et al., 2018), thus being able to handle CUI-less issues. For instance, compared to 30% CUI-less mentions in the CLEF/SemEval dataset, the compositional annotation approach used in Luo, Sun, and Rumshisky (2019b) reduced the percentage of CUI-less mentions to 2.7%. In general, there are two compositional rules: 1) "aggregate" or "composite" concepts that consist of multiple self-contained pre-coordinated concepts in the text mention, e.g., *breast and ovarian cancer* contains concepts *breast caner* and *ovarian cancer*; and 2) "composed" concepts which collectively act to describe a single concept, e.g., a single concept *left coronary artery stenosis* could be composed by multiple concepts *left* and *coronary artery stenosis*. For the second rule, there are usually a few possible different-but-equivalent splits for one single concept, e.g., *left coronary artery* and *artery stenosis* could also compose the concept *left coronary artery stenosis*. To handle these issues, some concept normalization tasks have annotation guidelines to prefer one split over the other.

Large mapping spaces: Another major challenge of biomedical concept normalization is the larger mapping space, i.e., the amount of concepts in the ontology. For instance, UMLS contains 3.5 million concepts, including entities and types; another widely used ontology

for concept normalization, SNOMED-CT US contains around 0.5 million concepts. As mentioned before, concepts from these ontologies have multiple synonyms; and terms to describe the concepts are also ambiguous. Although in real-life applications, the goal is to find one concept from the complete UMLS, some concept normalization tasks narrow down the mapping spaces by pre-defining some semantic groups, e.g., SemEval-2014 Task 7 (Pradhan et al., 2014) only focused on the disorder semantic groups from SNOMED-CT.

5.1.2 Related work

In Section 5.1.1, we have discussed the main challenges of biomedical concept normalization. As variation is much more common than ambiguity in the biomedical domain, most concept normalization systems focus on the variation problem and adopt approaches less relying on the context of concept mentions. In the remainder of this section, we focus on the approaches that do not use any context information. For approaches using context information, we refer the interested readers to research works of entity linking (Gupta, Singh, and Roth, 2017; Yin et al., 2019), word sense disambiguation (Stevenson et al., 2008; Yepes, 2017), entity disambiguation (Sun et al., 2015; Ganea and Hofmann, 2017), and concept normalization (Luo, Sun, and Rumshisky, 2019a; Schumacher, Mulyar, and Dredze, 2020).

Traditional approaches for biomedical concept normalization involve string match and dictionary look-up. These approaches differ in how they construct dictionaries, such as collecting concept mentions from the labeled data as extra synonyms (Leal, Martins,

and Couto, 2015; Lee, Hsu, and Kao, 2016), and in different string matching techniques, such as string overlap and edit distance (Kate, 2016). Two of the most commonly used knowledge-intensive concept normalization tools, MetaMap (Aronson, 2001) and cTAKES (Savova et al., 2010) both employ rules to first generate lexical variants for each noun phrase and then conduct dictionary look-up for each variant. Several systems (D'Souza and Ng, 2015; Jonnagaddala et al., 2016) have demonstrated that rule-based biomedical concept normalization systems achieve performance competitive with other approaches in a *sieve-based approach* that carefully selects combinations and orders of dictionaries, exact and partial matching, and heuristic rules. However, such rule-based approaches struggle when there are great variations between concept mention and concept, which is common, for example, when comparing social media text to medical ontologies.

Due to the availability of shared tasks and annotated data, the field has shifted toward machine learning techniques. We divide the machine learning approaches into two categories, classification (Savova et al., 2008; Stevenson et al., 2009; Limsopatham and Collier, 2016; Yepes, 2017; Festag and Spreckelsen, 2017; Lee et al., 2017; Tutubalina et al., 2018; Niu et al., 2019) and learning to rank (Leaman, Islamaj Doğan, and Lu, 2013; Liu and Xu, 2017; Li et al., 2017; Nguyen, Nguyen, and Dang, 2018; Murty et al., 2018).

Most classification-based approaches using deep neural networks have shown strong performance. They differ in using different architectures, such as Gated Recurrent Units (GRU) with attention mechanisms (Tutubalina et al., 2018), multi-task learning with auxiliary tasks to generate attention weights (Niu et al., 2019), or pre-trained transformer

networks (Li et al., 2019; Miftahutdinov and Tutubalina, 2019); different sources for training word embeddings, such as Google News (Limsopatham and Collier, 2016) or concept definitions from UMLS Metathesaurus (Festag and Spreckelsen, 2017); and different input representations, such as using character embeddings (Niu et al., 2019). All classification approaches share the disadvantage that the output space must be the same size as the number of concepts to be predicted, and thus the output space tends to be small such as 2,200 concepts in (Limsopatham and Collier, 2016) and around 22,500 concepts in (Weissenbacher et al., 2019). Classification approaches also struggle with concepts that have only a few example mentions in the training data.

Researchers have applied point-wise learning to rank (Liu and Xu, 2017; Li et al., 2017), pair-wise learning to rank (Leaman, Islamaj Doğan, and Lu, 2013; Nguyen, Nguyen, and Dang, 2018), and list-wise learning to rank (Murty et al., 2018; Ji, Wei, and Xu, 2020) on biomedical concept normalization. Generally, the learning-to-rank approach has the advantage of reducing the output space by first obtaining a smaller list of possible candidate concepts via a candidate generator and then ranking them. DNorm (Leaman, Islamaj Doğan, and Lu, 2013), based on a pair-wise learning-to-rank model where both mentions and concept names were represented as TF-IDF vectors, was the first to use learning-to-rank for concept normalization and achieved the best performance in the ShARe/CLEF eHealth 2013 shared task. List-wise learning-to-rank approaches are both computationally more efficient than pair-wise learning-to-rank (Cao et al., 2007) and empirically outperform both point-wise and pair-wise approaches (Xia et al., 2008). There are two implementations of

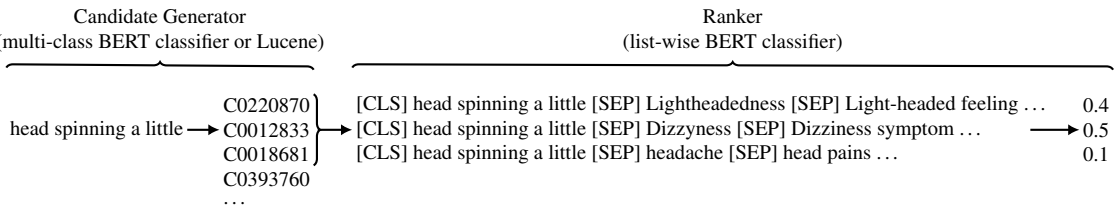

Figure 5.2: Proposed architecture for biomedical concept normalization: candidate generation and ranking.

list-wise classifiers using neural networks for concept normalization: Murty et al. (2018) treat the selection of the best candidate concept as a flat classification problem, losing the ability to handle concepts not seen during training; Ji, Wei, and Xu (2020) take a generate-and-rank approach similar to ours, but they do not leverage resources such as synonyms or semantic type information from UMLS in their BERT-based ranker.

5.2 Proposed biomedical concept normalization framework

We define a concept mention m as an abbreviation such as “MI", a noun phrase such as “heart attack", or even a short text such as “an obstruction of the blood supply to the heart". The goal is then to assign m with a concept c. Formally, given a list of pre-identified concept mentions $M = \{m_1, m_2, ..., m_n\}$ in the text and an ontology or knowledge base with a set of concepts $C = \{c_1, c_2, ..., c_t\}$, the goal of concept normalization is to find a mapping function $c_j = f(m_i)$ that maps each textual mention to its correct concept.

We approach biomedical concept normalization in two steps (shown in Figure 5.2): we first use a candidate generator $G(m, C) \rightarrow C_m$ to generate a list of candidate concepts C_m for each mention m, where $C_m \subseteq C$ and $|C_m| \ll |C|$. We then use a candidate

ranker $R(m, C_m) \rightarrow \hat{C_m}$, where $\hat{C_m}$ is a re-ranked list of candidate concepts sorted by their relevance, preference, or importance. But unlike information retrieval tasks where the order of candidate concepts in the sorted list $\hat{C_m}$ is crucial, in biomedical concept normalization we care only that the one true concept is at the top of the list.

The main idea of the two-step approach is that we first use a simple and fast system with high recall to generate candidates, and then a more precise system with more discriminative input to rank the candidates.

5.2.1 Candidate generator

We implement two kinds of candidate generators: a BERT-based multi-class classifier when the number of concepts in the ontology is small, and a Lucene-based[2] dictionary look-up when there are hundreds of thousands of concepts in the ontology.

BERT-based multi-class classifier

BERT (Devlin et al., 2019) is a contextualized word representation model that has shown great performance in many NLP tasks. Here, we use BERT in a multi-class text-classification configuration as our candidate concept generator. We use the final hidden vector $V_m \in \mathbb{R}^H$ corresponding to the first input token (`[CLS]`) generated from $BERT(m)$ and a classification layer with weights $W \in \mathbb{R}^{|C| \times H}$, and train the model using

[2] `https://lucene.apache.org/`

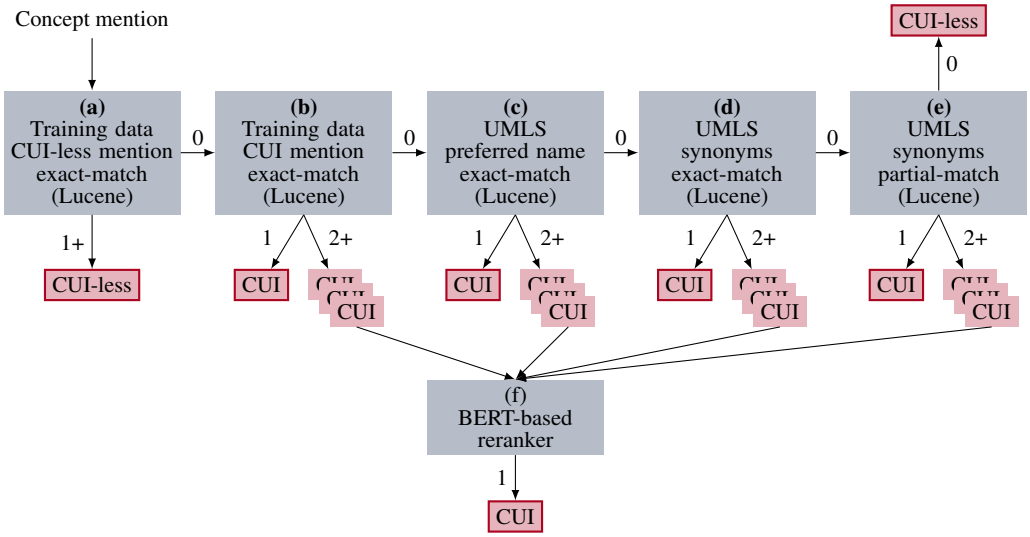

Figure 5.3: Architecture of the lucene-based dictionary look-up system. The edges out of a search process indicate the number of matches necessary to follow the edge. Outlined nodes are terminal states that represent the predictions of the system.

a standard classification loss:

$$L_G = y * log(softmax(V_m W^T)) \tag{5.1}$$

where y is a one-hot vector, and $|y| = |C|$. The score for all concepts is calculated as:

$$p(C) = softmax(V_m W^T) \tag{5.2}$$

We select the top k most probable concepts in $p(C)$ and feed that list C_m to the ranker.

Lucene-based dictionary look-up system

Multi-pass sieve rule based systems (D'Souza and Ng, 2015; Jonnagaddala et al., 2016; Luo, Sun, and Rumshisky, 2019b) achieve competitive performance when used with the right combinations and orders of different dictionaries, exact and partial matching, and heuristic rules. Such systems relying on basic lexical matching algorithms are simple and fast to implement, but they are only able to generate candidate concepts which are morphologically similar to a given mention.

Inspired by the work of Luo, Sun, and Rumshisky (2019b), we implement a Lucene-based sieve normalization system (shown in Figure 5.3) which consists of the following components:

(a) Lucene index over the training data finds all CUI-less mentions that exactly match mention m.

(b) Lucene index over the training data finds CUIs of all training mentions that exactly match mention m.

(c) Lucene index over UMLS finds CUIs whose preferred name exactly matches mention m.

(d) Lucene index over UMLS finds CUIs where at least one synonym of the CUI exactly matches mention m.

(e) Lucene index over UMLS finds CUIs where at least one synonym of the CUI has

high character overlap with mention m. To check the character overlap, we run the following three rules sequentially: token-level matching, fuzzy string matching with a maximum edit distance of 2, and character 3-gram matching.

If Lucene(**(b)**-**(e)**) generates C_m such that $|C_m| > 1$, then mention m and C_m are fed as input to the candidate ranker (f).

5.2.2 Candidate ranker

After the candidate generator produces a list of concepts, we use a BERT-based list-wise classifier to select the most likely candidate. BERT allows us to match morphologically dissimilar (but semantically similar) mentions and concepts, and the list-wise classifier takes both mention and candidate concepts as input, allowing us to handle concepts that appear infrequently (or never) in the training data.

Here, we use BERT similar to a question answering configuration, where given a concept mention m, the task is to choose the most likely candidate concept c_m from all candidate concepts C_m. As shown in Figure 5.2, our classifier input includes the text of the mention m and all synonyms of the candidate concept c_m, and takes the form `[CLS]` m `[SEP]` $syn_1(c_m)$ `[SEP]` ... `[SEP]` $syn_s(c_m)$ `[SEP]`, where $syn_i(c_m)$ is the i^{th} synonym of concept c_m[3]. We calculate the final hidden vector $V_{(m,c_m)} \in \mathbb{R}^H$ corresponding to the first input token (`[CLS]`) generated from BERT for each such input, and then concatenate the hidden vectors of all candidate concepts to form a matrix $V_{(m,C_m)} \in$

[3]In preliminary experiments, we tried only the concept's preferred term and several other ways of separating synonyms, but none of these resulted in better performance.

$\mathbb{R}^{|C_m| \times H}$. We use this matrix and classification layer weights $W \in \mathbb{R}^H$, and compute a standard classification loss:

$$L_R = y * log(softmax(V_{(m,C_m)} W^T)). \tag{5.3}$$

where y is a one-hot vector, and $|y| = |C_m|$.

5.2.3 Semantic type regularizer

To encourage the list-wise classifier towards a more informative ranking than just getting the correct concept at the top of the list, we propose a semantic type regularizer that is optimized when candidate concepts with the correct semantic type are ranked above candidate concepts with incorrect types. The semantic type of the candidate concept is assumed correct only if it exactly matches the semantic type of the gold truth concept. If the concept has multiple semantic types, all must match. Our semantic type regularizer consists of two components:

$$R_p(\hat{y}_t, \hat{y}_p) = \sum_{p \in P(y)} (m_1 + \hat{y}_p - \hat{y}_t) \tag{5.4}$$

$$R_n(\hat{y}_p, \hat{y}_n) = \sum_{p \in P(y)} \max_{n \in N(y)} (m_2 + \hat{y}_n - \hat{y}_p) \tag{5.5}$$

where $\hat{y} = V_{(m,c_m)} W^T$, $N(y)$ is the set of indexes of candidate concepts with incorrect semantic types (negative candidates), $P(y)$ (positive candidates) is the complement of

$N(y)$, $\hat{y}_t$ is the score of the gold truth candidate concept, and thus $t \in P(y)$. The margins m_1 and m_2 are hyper-parameters for controlling the minimal distances between $\hat{y}_t$ and $\hat{y}_p$ and between $\hat{y}_p$ and $\hat{y}_n$, respectively. Intuitively, R_p tries to push the score of the gold truth concept above all positive candidates at least by m_1, and R_n tries to push the best scored negative candidate below all positive candidates by m_2.

The final loss function we optimize for the BERT-based list-wise classifier is:

$$L = L_R + \lambda R_p(\hat{y}_t, \hat{y}_p) + \mu R_n(\hat{y}_p, \hat{y}_n) \tag{5.6}$$

where λ and μ are hyper-parameters to control the tradeoff between standard classification loss and the semantic type regularizer.

5.3 Experiments

5.3.1 Datasets

Our experiments are conducted on three social media datasets, AskAPatient (Limsopatham and Collier, 2016), TwADR-L (Limsopatham and Collier, 2016), and SMM4H-17 (Sarker et al., 2018), and one clinical notes dataset, MCN (Luo, Sun, and Rumshisky, 2019b). We summarize dataset characteristics in Table 5.1.

AskAPatient The AskAPatient dataset[4] contains 17,324 adverse drug reaction (ADR) annotations collected from blog posts. The mentions are mapped to 1,036 medical

[4] http://dx.doi.org/10.5281/zenodo.55013

Dataset	AskAPatient	TwADR-L	SMM4H-17	MCN
Ontology	SNOMED-CT & AMT	MedDRA	MedDRA (PT)	SNOMED-CT & RxNorm
Subset	Y	Y	N	N
$\|C_{ontology}\|$	1,036	2,220	22,500	434,056
$\|ST_{ontology}\|$	22	18	61	125
$\|C_{dataset}\|$	1,036	2,220	513	3,792
$\|M\|$	17,324	5,074	9,149	13,609
$\|M_{train}\|$	15665.2	4805.7	5,319	5,334
$\|M_{test}\|$	866.2	142.7	2,500	6,925
$\|M\|/\|C_{dataset}\|$	16.72	2.29	17.83	3.59
$\|C_{test} - C_{train}\|$	0	0	43	2,256
$\|M_{test} - M_{train}\|/M_{test}$	39.7%	39.5%	34.7%	53.9%
$\|M_{ambiguous}\|/\|M\|$	1.2%	12.8%	0.8%	4.5%

Table 5.1: Dataset statistics, where C is a set of concepts, ST is a set of semantic types, and M is a set of mentions.

concepts with 22 semantic types from the subset of Systematized Nomenclature Of Medicine-Clinical Term (SNOMED-CT) and the Australian Medicines Terminology (AMT). We follow the 10-fold cross validation (CV) configuration in Limsopatham and Collier (2016) which provides 10 sets of train/dev/test splits.

TwADR-L The TwADR-L dataset[3] contains 5,074 ADR expressions from social media. The mentions are mapped to 2,220 Medical Dictionary for Regulatory Activities (MedDRA) concepts with 18 semantic types. We again follow the 10-fold cross validation configuration defined by Limsopatham and Collier (2016).

SMM4H-17 The SMM4H-17 dataset [5] consists of 9,149 manually curated ADR expressions from tweets. The mentions are mapped to 22,500 concepts with 61 semantic types from MedDRA Preferred Terms (PTs). We use the 5,319 mentions from the released set as our training data, and keep the 2,500 mentions from the original test set as evaluation.

[5] http://dx.doi.org/10.17632/rxwfb3tysd.1

MCN The MCN dataset consists of 13,609 concept mentions drawn from 100 discharge summaries from the fourth i2b2/VA shared task (Uzuner et al., 2011). The mentions are mapped to 3792 unique concepts out of 434,056 possible concepts with 125 semantic types in SNOMED-CT and RxNorm. We take 40 clinical notes from the released data as training, consisting of 5,334 mentions, the remaining 10 as dev set, and the standard evaluation data with 6,925 mentions as our test set. Around 2.7% of mentions in MCN could not be mapped to any concepts in the terminology, and are assigned the *CUI-less* label.

A major difference between the datasets is the space of concepts that systems must consider. For AskAPatient and TwADR-L, all concepts in the test data are also in the training data, and in both cases only a couple thousand concepts have to be considered. Both SMM4H-17 and MCN define a much larger concept space: SMM4H-17 considers 22,500 concepts (though only 513 appear in the data) and MCN considers 434,056 (though only 3,792 appear in the data). AskAPatient and TwADR-L have no unseen concepts in their test data, SMM4H-17 has a few (43), while MCN has a huge number (2,256). Even a classifier that perfectly learned all concepts in the training data could achieve only 70.15% accuracy on MCN. MCN also has more unseen mentions: 53.9%, where the other datasets have less than 40%. The MCN dataset is thus harder to memorize, as systems must consider many mentions and concepts never seen in training.

Unlike the clinical MCN dataset, in the three social media datasets – AskAPatient, TwADR-L, and SMM4H-17 – it is common for the ADR expressions to share no words

with their target medical concepts. For instance, the ADR expression “makes me like a zombie” is assigned the concept “C1443060” with preferred term “feeling abnormal”. The social media datasets do not include context, only the mentions themselves, while the MCN dataset provides the entire note surrounding each mention. Since only 4.5% of mentions in the MCN dataset are ambiguous, for the current experiments we ignore this additional context information.

5.3.2 Unified Medical Language System

The UMLS Metathesaurus (Bodenreider, 2004) links similar names for the same concept from nearly 200 different vocabularies such as SNOMED-CT, MedDRA, RxNorm, etc. There are over 3.5 million concepts in UMLS, and for each concept, UMLS also provides the definition, preferred term, synonyms, semantic type, relationships with other concepts, etc.

In our experiments, we make use of synonyms and semantic type information from UMLS. We restrict our concepts to the three vocabularies, MedDRA, SNOMED-CT, and RxNorm in the UMLS version 2017AB. For each concept in the ontologies of the four datasets, we first find its concept unique identifier (CUI) in UMLS. We then extract synonyms and semantic type information according to the CUI. Synonyms (English only) are collected from level 0 terminologies containing vocabulary sources for which no additional license agreements are necessary.

5.3.3 Evaluation metrics

For all four datasets, the standard evaluation of biomedical concept normalization systems is accuracy.

$$Accuracy = \frac{|C_{predicted} \cap C_{gold}|}{|C_{gold}|}$$

For the AskAPatient and TwADR-L datasets, which use 10-fold cross validation, the accuracy metrics are averaged over 10 folds.

As there are lots of unseen concepts in MCN dataset, so in addition to the official accuracy evaluation metric, we also calculate the accuracy for seen concepts (C_{seen}) vs. unseen concepts (C_{unseen}), that is, concepts that were vs. were not seen during training. For the dev set, concepts in the training data are considered seen. For the test set, concepts in the training or dev data are considered seen. Formally,

$$Accuracy_{seen} = \frac{|C_{predicted} \cap C_{seen}|}{|C_{seen}|}$$
$$Accuracy_{unseen} = \frac{|C_{predicted} \cap C_{unseen}|}{|C_{unseen}|}$$

5.3.4 Implementation details

We use the BERT-based multi-class classifier as the candidate generator on the three social media datasets AskAPatient, TwADR-L, and SMM4H-17, and the Lucene-based candidate

	Multi-class			List-wise			
	AAP	TwADR-L	SMM4H-17	AAP	TwADR-L	SMM4H-17	MCN
learning_rate	1e-4	5e-5	5e-5	5e-5	5e-5	3e-5	3e-5
num_train_epochs	30	30	40	10	10	20	30
per_gpu_train_batch_size	32	16	32	16	16	16	8
save_steps	487	301	166	976	301	333	250
warmup_steps	1463	903	664	976	301	666	750
list size (k)	-	-	-	10	20	10	30
m_1	-	-	-	0.0	0.0	0.0	0.1
m_2	-	-	-	0.2	0.2	0.2	0.2
λ	-	-	-	0.6	0.4	0.4	0.4
μ	-	-	-	0.6	0.4	0.4	0.8

Table 5.2: Hyper-parameters for BERT-based multi-class and list-wise classifiers. AAP=AskAPatient. Terms with underscores are hyper-parameters in huggingface's pytorch implementation of BERT.

generator for the MCN dataset. In the social media datasets, the number of concepts in the data is small, few test concepts are unseen in the training data, and there is a greater need to match expressions that are morphologically dissimilar from medical concepts. In the clinical MCN dataset, the opposites are true.

For all experiments, we use BioBERT-base (Lee et al., 2019), which further pre-trains BERT on PubMed abstracts (PubMed) and PubMed Central full-text articles (PMC). We use huggingface's pytorch implementation of BERT[6]. We select the best hyper-parameters based on the performance on dev set. Table 5.2 shows the hyper-parameters for our models. We use huggingface's pytorch implementation of BERT. We tune the hyperparameters via grid search, and select the best BERT hyper-parameters based on the performance on the dev set.

To keep the size of the candidate list equal to k for every mention, we apply the

[6]https://github.com/huggingface/transformers

following rules: if the list does not contain the gold concept and is already of length k, we inject the correct one and remove an incorrect candidate; if the list is not length of k, we inject the gold concept and the most frequent concepts in the training set to reach k.

As some experiments in MCN dataset are based on our participation in the N2C2 Shared Task Track 3 Concept Normalization, in the following subsection, we describe our submitted systems and updated systems during the post-evaluation.

5.3.5 Participation in the N2C2 shared task

Submitted Runs For the shared task, when multiple candidate concepts were matched by Lucene components **(b)**, **(c)**, or **(d)**, they were sent to a rule-based reranker that considered section types, and concepts matched by Lucene component **(e)** were sent to the BERT-based list-wise classifier. We also implemented an acronym matcher following component e that expanded the abbreviations and fed the expanded form as new input to component **(e)**. The BERT-based list-wise classier f-cde was trained on examples from Lucene(c + d + e), and took the first 10 matched candidate concepts as input. We submitted three system runs in this task. The first run used Lucene(a + b + c + d + e), the acronym matcher, and the rule-based reranker. The second and third runs used the same candidate generator as the first run, but paired it with BERT-based list-wise classier f-cde. The second run used only 40 files to train f-cde, tuning hyper-parameters on the 10 development files. The third run used all 50 files to train f-cde, but unfortunately was submitted with a row-alignment bug. The final submitted result for BERT-based

list-wise classier is an ensemble of three different training runs.

Post-Evaluation After the shared task, we removed the acronym matcher and rule-based ranker, and used Lucene(e) instead of Lucene(c + d + e) to generate training instances for the BERT-based list-wise classifier f-e as it generated many more training examples and resulted in better performance on the dev set. For the BERT-based list-wise classier, we take the first 30 matched candidate concepts as input, and we enhance f-e with a semantic type regularizer. The result of our final model BERT-based list-wise classifier is a single training run. Figure 5.3 shows our complete architecture.

5.3.6 Models

We separate out the different contributions from the following components of our architecture.

BERT The BERT-based multi-class classifier. When used alone, we select the most probable concept as the prediction.

Lucene The Lucene-based dictionary look-up. We experiment with different combinations of components in Lucene(a-e), e.g., indexing the training set, Lucene(a + b), or indexing components of UMLS, Lucene(c + d + e). When used alone, we take the first matched concept as the prediction.

+BERT-rank The BERT-based list-wise classifier, always used in combination with either BERT or Lucene as a canddiate generator. For MCN dataset, we experiment with

different inputs to BERT-based list-wise classifier: during training, we experiment with Lucene(e) or Lucene(c + d + e) to generate training instances for the BERT-based list-wise classifier, with these candidate generators being run over the training set, and any mentions that have multiple matched candidate concepts becoming training instances; during prediction, we experiment with Lucene(e), Lucene(c + d +e) and Lucene(a + b + c + d + e) to generate candidates. While for other datasets, inputs always come from the BERT-based multi-class classifier.

+ST The semantic type regularizer, always used in combination with BERT-ranker.

We also consider the case (**+gold**) where we artificially inject the correct concept into the candidate generator's list if it was not already there.

5.3.7 Comparisons with related methods

We compare our proposed architecture with the following state-of-the-art systems.

WordCNN Limsopatham and Collier (2016) use convolutional neural networks over pre-trained word embeddings to generate a vector representation for each mention, and then feed these into a softmax layer for multi-class classification.

WordGRU+Attend+TF-IDF Tutubalina et al. (2018) use a bidirectional GRU with attention over pre-trained word embeddings to generate a vector representation for each mention, concatenate such vector representations with the cosine similarities of the TF-IDF vectors between the mention and all other concept names, and then feed the

concatenated vector to a softmax layer for multi-class classification.

BERT+TF-IDF Miftahutdinov and Tutubalina (2019) take similar approach as Tutubalina et al. (2018), but use BERT to generate a vector representation for each mention. They concatenate the vector representations with the cosine similarities of the TF-IDF vectors between the mention and all other concept names, and then feed the concatenated vector to a softmax layer for multi-class classification.

CharCNN+Attend+MT Niu et al. (2019) use a multi-task attentional character-level convolution neural network. They first convert the mention into a character embedding matrix. The auxiliary task network takes the embedding matrix as input for a CNN to learn to generate character-level domain-related importance weights. Such learned importance weights are concatenated with the character embedding matrix and fed as input to another CNN model with a softmax layer for multi-class classification.

CharLSTM+WordLSTM Han et al. (2017) first use a forward LSTM over each character of the mention and its corresponding character class such as lowercase or uppercase to generate a character-level vector representation, then use another bi-directional LSTM over each word of the mention to generate a word-level representation. They concatenate character-level and word-level representations and feed them as input to a softmax layer for multi-class classification.

LR+MeanEmbedding Belousov, Dixon, and Nenadic (2017) calculate the mean of three different weighted word embeddings pre-trained on GoogleNews, Twitter and

	TwADR-L		AskAPatient		SMM4H-17	
Approach	Dev	Test	Dev	Test	Dev	Test
WordCNN (Limsopatham and Collier, 2016)	-	44.78	-	81.41	-	-
WordGRU+Attend+TF-IDF (Tutubalina et al., 2018)	-	-	-	85.71	-	-
BERT+TF-IDF (Miftahutdinov and Tutubalina, 2019)	-	-	-	-	-	**89.64**
CharCNN+Attend+MT (Niu et al., 2019)	-	46.46	-	84.65	-	-
CharLSTM+WordLSTM (Han et al., 2017)	-	-	-	-	-	87.20
LR+MeanEmbedding (Belousov, Dixon, and Nenadic, 2017)	-	-	-	-	-	87.70
BERT	47.08	44.05	88.63	**87.52**	84.74	87.36
BERT + BERT-rank	48.07	46.32	88.14	87.10	84.44	87.66
BERT + BERT-rank + ST	47.98	**47.02**	88.26	87.46	84.66	88.24
BERT + gold + BERT-rank	52.70	49.69	89.06	87.92	88.57	90.16
BERT + gold + BERT-rank + ST	52.84	50.81	89.68	88.51	88.87	91.08

Table 5.3: Comparisons of our proposed biomedical concept normalization architecture against the current state-of-the-art performances on *TwADR-L*, *AskAPatient*, and *SMM4H-17* datasets.

DrugTwitter as vector representations for the mention, where word weights are calculated as inverse document frequency. Such vector representations are fed as input to a multinomial logistic regression (LR) model for multi-class classification.

Sieve-based Luo, Sun, and Rumshisky (2019b) build a sieve-based normalization model which contains exact-match and MetaMap (Aronson, 2001) modules. Given a mention as input, the exact-match module first looks for mentions in the training data that exactly match the input, and then looks for concepts from the ontology whose synonyms exactly match the input. If no concepts are found, the mention is fed into MetaMap. They run this sieve-based normalization model twice. In the first round, the model lower-cases the mentions and includes acronym/abbreviation tokens during dictionary lookup. In the second round, the model lower-cases the mentions spans and also removes special tokens such as “'s”, “"”, etc.

Since our focus is individual systems, not ensembles, we compare only to other non-ensembles[7].

5.4 Results

Table 5.3 shows that our complete model, BERT + BERT-rank + ST, achieves a new state-of-the-art on two of the social media test sets. The TwADR-L dataset is the most difficult, with our complete model achieving 47.02% accuracy. In the other datasets, performance of our complete model is much higher: 87.46% for AskAPatient, 88.24% for SMM4H-17[8]. On the TwADR-L and SMM4H-17, adding the BERT-based ranker improves performance over the candidate generator alone, and adding the semantic type regularization further improves performance. On AskAPatient, performance of the full model is similar to just the BERT multi-class classifier, perhaps because in this case BERT alone already successfully improves the state-of-the-art from 85.71% to 87.52%. The +gold setting allows us to answer how well our ranker would perform if our candidate generator made no mistakes. First, we can see that if the correct concept is always in the candidate list, our list-based ranker (+BERT-rank) outperforms the multi-class classifier (BERT) on all test sets. We also see in this setting that the benefits of the semantic type regularizer are amplified, e.g.,with test sets of TwADR-L (table 5.3) showing more than 1.00% gain in accuracy from using the regularizer.

[7] An ensemble of three systems (including CharLSTM+WordLSTM and LR+MeanEmbedding) achieved 88.7% accuracy on the SMM4H-17 dataset (Sarker et al., 2018).

[8] Miftahutdinov and Tutubalina (2019) use the same architecture as our BERT-based multi-class classifier (row 7), but they achieve 89.28% of accuracy on SMM4H-17. We were unable to replicate this result as their code and parameter settings were unavailable.

Systems		Dev				Test			
		Recall @30	Accuracy	Seen Accuracy	Unseen Accuracy	Recall @30	Accuracy	Seen Accuracy	Unseen Accuracy
Sieve-based (Luo, Sun, and Rumshisky, 2019b)		-	-	-	-	-	76.35%	-	-
Submitted Run	Submitted run #1	-	-	-	-	-	79.44%	-	-
	Submitted run #2	-	-	-	-	-	81.66%	-	-
	Submitted run #3	-	-	-	-	-	75.95%	-	-
	Submitted run #3 (after fixing row-alignment bug)	-	-	-	-	-	81.68%	-	-
Lucene Only	Lucene(a + b)	52.30%	52.07%	77.00%	0	55.08%	54.66%	77.91%	0
	Lucene(c)	32.59%	32.59%	36.36%	24.71%	34.56%	34.35%	38.43%	24.77%
	Lucene(d)	62.22%	53.48%	58.49%	43.02%	62.86%	53.26%	57.20%	43.98%
	Lucene(e)	86.15%	58.07%	58.93%	56.29%	85.73%	57.50%	57.76%	56.89%
	Lucene(c + d)	58.74%	55.56%	60.35%	45.54%	60.06%	56.43%	60.95%	45.82%
	Lucene(d + e)	81.78%	63.04%	64.95%	59.04%	82.20%	62.11%	63.15%	59.65%
	Lucene(c + d + e)	78.30%	65.11%	66.81%	61.56%	79.21%	65.29%	66.90%	61.49%
	Lucene(a + b + e)	88.15%	76.67%	86.64%	55.84%	88.38%	77.43%	86.23%	56.75%
	Lucene(a + b + c + d + e)	87.41%	78.96%	87.62%	60.87%	87.87%	79.25%	86.89%	61.30%
Lucene + BERT-rank	Lucene(e) + BERT-rank (f-e)	86.15%	78.96%	84.88%	66.59%	85.73%	77.36%	83.78%	62.26%
	Lucene(a + b + e) + BERT-rank(f-e)	88.15%	83.13%	91.05%	66.59%	88.38%	82.06%	90.41%	62.46%
	Lucene(a + b + c + d + e) + BERT-rank(f-e)	87.41%	83.56%	91.02%	67.96%	87.87%	82.75%	90.30%	64.99%
	Lucene(e) + BERT-rank(f-e) + ST	86.15%	79.85%	85.10%	68.88%	85.73%	77.98%	83.92%	64.01%
	Lucene(c + d + e) + BERT-rank(f-e) + ST	78.30%	75.41%	77.77%	70.48%	79.21%	75.00%	78.59%	66.57%
	Lucene(a + b + e) + BERT-rank(f-e) + ST	88.15%	84.05%	91.71%	68.05%	88.38%	82.90%	90.90%	64.10%
	Lucene(a + b + c + d + e) + BERT-rank(f-e) + ST	87.41%	**84.44%**	91.57%	69.57%	87.87%	**83.56%**	90.80%	66.52%

Table 5.4: Accuracy of our proposed biomedical concept normalization architecture and different combinations of components in Lucene(a-e) and BERT-rank on *MCN* dataset. Recall@30: how often the correct candidate is within the first 30 matched candidate concepts. The numbers in bold are the best performance.

Table 5.4 shows the accuracy of multiple systems on the dev and test sets of MCN corpus. The submitted run rows show that our Lucene-based lookup (submitted run #1) outperforms the previous state-of-the-art [10], 79.44% vs. 76.35%, and our BERT-rank

System	Overall	$\lvert C_m \rvert > 1$		
	accuracy	$\lvert m \rvert$	accuracy	recall@30
Lucene(a + b + c + d + e) + BERT-rank(f-e) + ST	84.44%	374	62.23%	72.99%
Lucene(a + b + c + d + e) + gold + BERT-rank(f-e) + ST	88.07%	374	75.40%	100%
Lucene(e) + BERT(f-e) + ST	79.85%	1327	80.11%	86.51%
Lucene(e) + gold + BERT(f-e) + ST	89.19%	1327	89.60%	100%

Table 5.5: Accuracies of our proposed architectures and their oracle versions on *MCN* dev set. Accuracy: how often the first matched candidate concept is correct. Recall@30: how often the correct candidate is within the first 30 matched candidate concepts.

(submitted run #2) further improves performance to 81.66%. Comparing submitted run #1 and Lucene(a + b + c + d + e), which was the same but without the acronym matcher and rule-based ranker, we see that these two components make little difference (79.25% vs. 79.44%), and hence we excluded them from the remaining experiments. Looking at the Lucene-only rows, we can see that using only UMLS preferred terms - Lucene(c) - yields poor accuracy (34.35%) and low recall@30 (34.56%), while using all synonyms with either exact matching or partial matching – Lucene(d) or Lucene(e) - yields better accuracy (53.26% or 57.50%) and higher recall@30 (62.86% or 85.73%). But we can also see that UMLS preferred terms have low coverage but high precision: when added to any pipeline, they improve that pipeline's accuracy while lowering its recall@30: Lucene(c + d) is more accurate than Lucene(d), and Lucene(c + d + e) is more accurate than Lucene(d + e), but the reverse is true for recall@30.

Comparing the Lucene + BERT-rank rows to the Lucene-only rows (table 5.4), we see that adding the BERT-rank always improves performance. For example, while Lucene(e) achieves only 57.50% accuracy, adding BERT(f -e + ST) increases accuracy to 77.98%.

Applying the same BERT model on top of Lucene(a + b + c + d + e) yields the best accuracy we achieved, 83.56%. Encouragingly, performance improves not only for seen concepts but also for unseen concepts with all BERT models. This suggests that our generate-and-rank approach to CN is successful, and generalizes better than the Lucene-based lookup alone. In contrast to our results with Lucene alone, we find that specially handling preferred terms is no longer necessary with the BERT ranker: for example, the Lucene(e) + BERT-rank(f -e + ST) that ignores preferred terms actually outperforms the Lucene(c + d + e) + BERT-rank(f-e + ST) model that handles them, 77.98% to 75.00%, probably in part because Lucene(e) has substantially higher recall than Lucene(c + d + e) (85.73% vs. 79.21%). Comparing the rows with and without semantic type regularizer (+ST) we can see that encouraging the model to learn about UMLS semantic types leads to small but consistent gains across all experiments, with larger gains for unseen concepts than for seen ones. For example, on unseen concepts, Lucene(e) + BERT-rank (f -e + ST) outperforms Lucene(e) + BERT-rank (f -e), 64.01% to 62.26%.

As MCN corpus has the most realistic setting where the number of candidate concepts is large and many test concepts were never seen during training, we quantify the remaining error in the system by examining where there is still room for improvement (shown at table 5.5). Since Lucene(a + b + c + d + e) achieves an recall@30 of 87.41 on the dev set, that is also the upper bound for the accuracy performance of the BERT-based ranker. Lucene(a + b + c + d + e) + BERT-rank(f-e + ST) achieves accuracy of 84.44%, a 5.48% improvement over the Lucene accuracy of 78.96%, and 2.97% away from the upper bound.

Components	overall			$\|C_m\| = 1$		$\|C_m\| > 1$		
	$\|m\|$	accuracy	recall@30	$\|m\|$	accuracy	$\|m\|$	accuracy	recall@30
Lucene(a + b)	705	97.16%	97.59%	681	97.65%	24	83.33%	95.83%
Lucene(c)	165	84.42%	84.42%	161	84.47%	4	75%	75%
Lucene(d)	164	73.78%	81.10%	127	82.68%	37	43.24%	75.68%
Lucene(e)	315	38.41%	69.84%	6	16.67%	309	38.83%	70.87%
Lucene(a + b + c + d + e)	1350	78.96%	87.41%	976	92.93%	374	42.51%	72.99%

Table 5.6: Accuracy for each component of the candidate generator in our best complete system Lucene(a + b + c + d + e) + BERT-rank(f-e + ST) on dev set of *MCN* corpus. $|C_m|$: the size of the candidate concepts. $|m|$: number of mentions predicted by each component.

We can also ask what would happen if the candidate generator always included the correct concept somewhere in the top-k. Table 4 shows that even with a top-k list that's guaranteed to include the correct concept, Lucene(a + b + c + d + e) + BERT-rank(f-e + ST) achieves only 88.07%, 11.03% away from the upper bound in this case of 100%. So we can conclude that while there is a need to improve the candidate generator so that it can find the correct candidate for the 12.59% of mentions where it fails to do so, the BERT-based reranker also needs to learn more about ranking such mentions to close its 11.03% performance gap.

5.5 Discussion

Table 5.6 shows the performance for each component in the candidate generator. Components run first have better performance, as is standard in a sieve-based system. We analyze the errors from each component:

(a + b): Exact matching against the training data makes only 705 predictions, even though 913 concepts in the dev data were seen during training. The missing concepts are string mismatches between mentions and indexing, including different determiners such as

"your incision" vs "the incision", abbreviations such as "an exercise tolerance test" vs "ETT", typos such as "dressing changes" vs "adressing change", prepositional phrases with different orders such as "debulking of tumor" vs "tumor debulking", etc. These missed matches demonstrate the fragility of simply memorizing the training data, and the importance of using other resources such as UMLS synonyms (which are able to predict more than 70% of these missing concepts). Most false positive errors come from ambiguous mentions where the same string is mapped to more than one concept, e.g., "acute" could mean "sudden onset (attribute)" or "acute phase", "ANTERIOR MYOCARDIAL INFARCTION" could mean "Old anterior myocardial infarction" or "Acute Anterior Wall Myocardial Infarction". Extra context information would be required to disambiguate these mentions.

(c): Exact matching against UMLS preferred terms has a great potential for CN because of its 84.42% accuracy and fewer predictions with $|C_m| > 1$ than other components. Most false positive errors come from the lack of context information or domain knowledge. For instance, c finds a spatial concept "brachial" for mention "brachial" in "... 2+ brachial , 1+ radial ...", while the correct concept is "brachial pulse, function". There are still many gaps though between the terms the clinicians use and the preferred terms in UMLS.

(d): Using synonyms and lexical variants for each concept increases the search space, which makes it possible to find concepts whose preferred term is lexically dissimilar

to the mention, e.g., d finds concept “pansystolic murmur” for mention “holosystolic murmur” as one synonym of the concept exactly matches the mention. However, it also has the disadvantage of finding more than one candidate concept, e.g., d finds three candidate concepts “Examination of reflexes”, “Observation of reflex”, and “Reflex action”, where all of them share the same synonym “reflexes”. Additional information such as semantic type, other synonyms, or the context is required to select the most probable one.

(e): Among 315 mentions predicted by partially matching against the UMLS synonyms, accuracy is 38.41%, while recall@30 is 69.84%; 98.10% of mentions have more than one mapped candidate concepts. Most false positive errors come from: 1) small character overlaps between mention and concept synonyms, e.g., mention “upgoing” with concept “upward”, mention “acute cardiopulmonary process” with concept “acute pulmonary heart disease”; 2) abbreviations such as mention “C02” with “Carbon dioxide content measurement”; 3) lack of domain knowledge such as mention “an 80% stenosis” with concept “partial stenosis”; (4) ambiguous mention such as “an ulcer” with concept “pressure ulcer” and “ulcer”.

5.5.1 Qualitative analysis of semantic type regularizer

Table 5.7 shows an example that is impossible for the multi-class classifier approach to concept normalization. The concept mention “an abdominal wall hernia” in the clinical MCN dataset needs to be mapped to the concept with the preferred name “Hernia of

Candidates	L	BR
Repair of abdominal wall hernia	1	3
Repair of anterior abdominal wall hernia	2	4
Obstructed hernia of anterior abdominal wall	3	5
Hernia of abdominal wall	4	1
Abdominal wall hernia procedure	5	2

Table 5.7: Predicted candidate concepts for mention *An abdominal wall hernia* and their rankings among the outputs of Lucene (L) and BERT-Ranker (BR). Gold concept is *Hernia of abdominal wall*.

Candidates	BR	STR	ST
Influenza-like illness	1	2	DS
Influenza	2	4	DS
Influenza-like symptoms	3	1	SS
Feeling tired	4	5	F
Muscle cramps in feet	5	3	SS

Table 5.8: Predicted candidate concepts for mention *felt like I was coming down with flu* and their rankings among the outputs of BERT-Ranker (BR) and BERT-Ranker + semantic type regularizer (STR). Gold concept is *flu-like symptoms*. Semantic types (ST) of the candidates include: disease or syndrome (DS), sign or symptom (SS), finding (F)

abdominal wall", but that concept never appeared in the training data. The Lucene-based candidate generator finds this concept, but only through character overlap (step **(e)**) and several other concepts have high overlap as well. Thus Lucene ranks the correct concept 4th in its list. The BERT ranker is able to compare "an abdominal wall hernia" to "Hernia of abdominal wall" and recognize that as a better match than the other options, re-assigning it to rank 1.

Table 5.8 shows an example that illustrates why the semantic type regularizer helps. The mention "felt like I was coming down with flu" in the social media AskAPatient dataset needs to be mapped to the concept with the preferred name "influenza-like symptoms",

which has the semantic type of a sign or symptom. The BERT ranker ranks two disease or syndromes higher, placing the correct concept at rank 3. After the semantic type regularizer is added, the system recognizes that the mention should be mapped to a sign or symptom, and correctly ranks it above the disease or syndromes. Note that this happens even though the ranker does not get to see the semantic type of the input mention at prediction time.

5.6 Limitations and future research

The available biomedical concept normalization datasets are somewhat limited. Lee et al. (2017) notes that AskAPatient and TwADR-L have issues including duplicate instances, which can lead to bias in the system; many phrases have multiple valid mappings to concepts but the context necessary to disambiguate is not part of the dataset; and the 10-fold cross-validation makes training complex models unnecessarily expensive. These datasets are also unrealistic in that all concepts in the test data are seen during training. Future research should focus on more realistic datasets that follow the approach of MCN in annotating mentions of concepts from a large ontology and including the full context.

Our ability to explore the size of the candidate list was limited by our available computational resources. As the size of the candidate list increases, the true concept is more likely to be included, but the number of training instances also increases, making the computational cost larger, especially for the datasets using 10-fold cross-validation. We chose candidate list sizes as large as we could afford, but there are likely further gains possible with larger candidate lists.

Our semantic type regularizer is limited to exact matching: it checks only whether the semantic type of a candidate exactly matches the semantic type of the true concept. The UMLS ontology includes many other relations, such as is-a and part-of relations, and extending our regularizer to encode such rich semantic knowledge may yield further improvements in the BERT-based ranker.

CHAPTER 6

Vector Space Model for Biomedical Concept Normalization

In the previous chapter, we have described the biomedical concept normalization task and our proposed generate-and-rank framework. In this chapter, we discuss another approach for biomedical concept normalization, where we view biomedical concept normalization as an information retrieval task.

We first briefly review the literature using vector space models for biomedical concept normalization in Section 6.1. We then introduce a vector-space model for concept normalization in Section 6.2, where mentions and concepts are encoded via transformer networks that are trained via a triplet objective with online hard triplet mining. We also explore a variety of strategies for searching with the trained vector-space model in Section 6.2.2. Next, we present our experimental results on five datasets with three in the domain of scientific articles and two in the domain of clinical notes in Section 6.5 and Section 6.5. Finally, we discuss current limitations and future work in Section 6.6.

6.1 Related work

In the last chapter, we proposed a two-step framework for concept normalization, including a non-trained candidate generator and a supervised candidate ranker that takes both mention and candidate concept as input. Generally, such approaches (Leaman, Islamaj Doğan, and

Lu, 2013; Li et al., 2017; Liu and Xu, 2017; Nguyen, Nguyen, and Dang, 2018; Murty et al., 2018; Mondal et al., 2019; Ji, Wei, and Xu, 2020; Xu, Zhang, and Bethard, 2020) have the advantage of reducing the output space by first obtaining a smaller list of possible candidate concepts via a candidate generator and then ranking them. However, they require complex pipelines and fail if the candidate generator does not find the gold truth concept. Besides, the training of a candidate ranker requires negative sampling beforehand, and it is unclear if these pre-selected negative samples are informative for the whole training process (Hermans, Beyer, and Leibe, 2017; Sung et al., 2020).

Inspired by Schroff, Kalenichenko, and Philbin (2015), we propose a triplet network with online hard triplet mining for concept normalization. Our framework sets up concept normalization as a one-step process, calculating the similarity between vector representations of the mention and all ontology concepts. Online hard triplet mining allows such a vector space model to generate triplets of (mention, true concept, false concept) within a mini-batch, leading to efficient training and fast convergence (Schroff, Kalenichenko, and Philbin, 2015). There are two advantages of applying the vector space model for concept normalization: 1) it is computationally cheap as we only compute the representations for all concepts in ontology once after training the network; 2) it allows concepts and synonyms to be added or deleted after the network is trained, a flexibility that is important for the biomedical domain where frequent updates to ontologies like the Unified Medical Language System (UMLS) Metathesaurus[1] are common.

[1] https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/index.html

In contrast with previous vector space models where mention and candidate concepts are mapped to vectors via TF-IDF (Leaman, Islamaj Doğan, and Lu, 2013), TreeLSTMs (Liu and Xu, 2017), CNNs (Nguyen, Nguyen, and Dang, 2018) or ELMO (Schumacher, Mulyar, and Dredze, 2020), we generate vector representations with BERT (Devlin et al., 2019), since it can encode both surface and semantic information (Ma et al., 2019).

The most similar work to ours is BIOSYN (Sung et al., 2020), which also trains a vector space model based on the similarity between mention and concept synonym vectors. However, their approach requires iterative candidate retrieval over the entire training data during each training step, requires both BERT-based and TF-IDF-based representations, and performs a variety of pre-processing such as acronym expansion. In contrast, our approach achieves competitive results both more efficiently (no need for iterative candidate retrieval) and more simply (no need for additional TF-IDF representations or acronym expansion).

6.2 Proposed methods

We define a concept mention m as a text string in a corpus D, and a concept c as a unique identifier in an ontology O. The goal of concept normalization is to find a mapping function f that maps each textual mention to its correct concept, i.e., $c = f(m)$. We define $t \in T(c)$ as either $t \in O(c)$, where $O(c)$ is the synonyms of the concept c from the ontology O, or $t \in D(c)$, where $D(c)$ is the mentions of the concept c in an annotated corpus D. $T(c)$ will allow the generation of tuples (t, c) such as (*MI,C0027051*) and (*Myocardial*

Infarction,C0027051). Note that, for a concept c, it is common to have $|O(c)| > |D(c)|$, $O(c) \cap D(c) = \emptyset$, or even $D(c) = \emptyset$, i.e., it is common for there to be more concept synonyms in the ontology than the annotated corpus, and it is common for the ontology and annotated corpus to provide different concept synonyms.

We implement f as a vector space model:

$$f(m) = \underset{\substack{c \in O \\ t \in T(c)}}{\operatorname{argmax}} Sim(V(m), V(t)) \tag{6.1}$$

where $V(x)$ is a vector representation of text x and Sim is a similarity measure such as cosine similarity, inner product, or euclidean distance. We learn the vector representations $V(x)$ using a triplet network architecture (Hoffer and Ailon, 2015), which learns from triplets of (anchor text t_i, positive text t_p, negative text t_n) where t_i and t_p are texts for the same concept, and t_n is a text for a different concept. The triplet network attempts to learn V such that for all training triplets:

$$Sim(V(t_i), V(t_{ip})) > Sim(V(t_i), V(t_{in})) \tag{6.2}$$

The triplet network architecture has been adopted in learning representations for images (Schroff, Kalenichenko, and Philbin, 2015; Gordo et al., 2016) and text (Neculoiu, Versteegh, and Rotaru, 2016; Reimers and Gurevych, 2019). It consists of three instances of the same sub-network (with shared parameters). When fed a (t_i, t_{ip}, t_{in}) triplet of texts, the sub-network outputs vector representations for each text, which are then fed into a

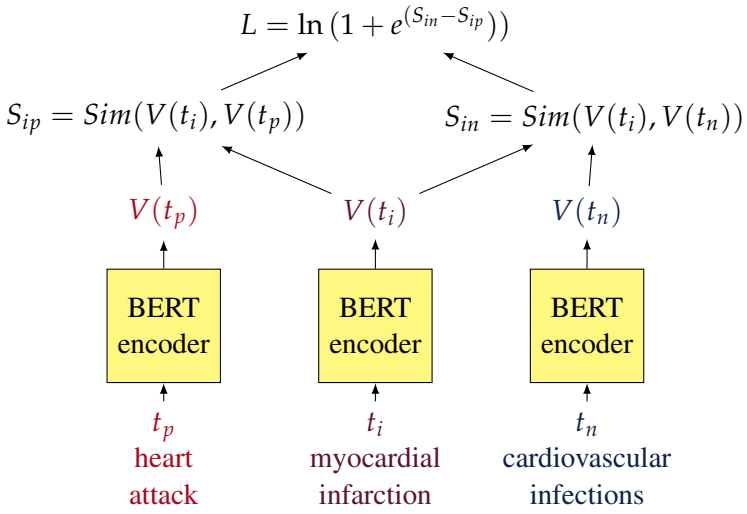

Figure 6.1: Example of loss calculation for a single instance of triplet-based training. The same BERT model is used for encoding t_i, t_p, and t_n.

triplet loss. We adopt PubMed-BERT (Gu et al., 2020) as the sub-network, where the representation for the concept text is an average pooling of the representations for all sub-word tokens[2]. This architecture is shown in Figure 6.1.

6.2.1 Online hard triplet mining

An essential part of learning using triplet loss is how to generate triplets. As the number of synonyms gets larger, the number of possible triplets grows cubically, making training impractical. We follow the idea of online triplet mining (Schroff, Kalenichenko, and Philbin, 2015) which considers only triplets within a mini-batch. We first feed a mini-batch of b concept texts to the PubMed-BERT encoder to generate a d-dimensional representation for each concept text, resulting in a matrix $M \in \mathbb{R}^{b \times d}$. We then compute the pairwise

[2]We also experimented with using the output of the *CLS*-token, and max-pooling of the output representations for the sub-word tokens as proposed by (Reimers and Gurevych, 2019), but neither resulted in better performance.

similarity matrix:

$$S = Sim(M, M^T) \tag{6.3}$$

where each entry S_{ij} corresponds to the similarity score between the i^{th} and j^{th} concept texts in the mini-batch. As the easy triplets would not contribute to the training and result in slower convergence (Schroff, Kalenichenko, and Philbin, 2015), for each concept text t_i, we only select a hard positive t_p and a hard negative t_n from the mini-batch such that:

$$p = \underset{j \in [1,b]: j \neq i \wedge C(j) = C(i)}{\operatorname{argmin}} S_{ij} \tag{6.4}$$

$$n = \underset{k \in [1,b]: k \neq i \wedge C(k) \neq C(i)}{\operatorname{argmax}} S_{ik} \tag{6.5}$$

where $C(x)$ is the ontology concept from which t_x was taken, i.e., if $t_x \in T(c)$ then $C(x) = c$.

We train the triplet network using batch hard soft margin loss (Hermans, Beyer, and Leibe, 2017):

$$L(i) = \ln\left(1 + e^{(S_{in} - S_{ip})}\right) \tag{6.6}$$

where S, n, and p are as in eqs. (6.3) to (6.5), and the hinge function, $\max(\cdot, 0)$, in the traditional triplet loss is replaced by a softplus function, $\ln(1 + e^{(\cdot)})$.

	Searching Over		Representation Type	
	Ontology	Training Data	Text	Concept
O-T	✓		✓	
O-C	✓			✓
D-T		✓	✓	
D-C		✓		✓
OD-T	✓	✓	✓	
OD-C	✓	✓		✓

Table 6.1: Similarity search modules with different representation sources and representation types.

6.2.2 Similarity search

Once our vector space model has been trained, we consider several options for how to find the most similar concept c to a text mention m. First, we must choose a search target: we can search over the concepts from the ontology, or the training data, or both. Second we must choose a representation type: we can compare m directly to each text (ontology synonym or training data mention) of each concept, or we can calculate a vector representation of each concept and then compare m directly to the concept vector. Table 6.1 summarizes these options.

We consider the following search targets:

Data We search over the concepts in the annotated data. These mentions will be more domain-specific (e.g., *PT* may refer to *patient* in clinical notes, but to *physical therapy* in scientific articles), but may be more predictive if the evaluation data is from the same domains. We search over the train subset of the data for dev evaluation, and train + dev subset for test evaluation.

Ontology We search over the concepts in the ontology. The synonyms will be more domain-independent, and the ontology will cover concepts that were never seen in the annotated training data.

Data and ontology We search over the concepts in both the training data and the ontology. For concepts in the annotated training data, their representations are averaged over mentions in the training data and synonyms in the ontology.

We consider the following representation types:

Text We represent each text (ontology synonym or training data mention) as a vector by running it through our triplet-fine-tuned PubMed-BERT encoder. Concept normalization then compares the mention vector to each text vector:

$$f(m) = \underset{\substack{c \in O \\ t \in T(c)}}{\operatorname{argmax}} Sim(V(m), V(t)) \tag{6.7}$$

Concept We represent each concept as a vector by taking an average over the triplet-fine-tuned PubMed-BERT representations of that concept's texts (ontology synonyms and/or training data mentions). Concept normalization then compares the mention vector to each concept vector:

$$f(m) = \underset{c \in O}{\operatorname{argmax}} Sim\left(V(m), \underset{t \in T(c)}{\operatorname{mean}} V(t)\right) \tag{6.8}$$

The averages here mean that different concepts with some (but not all) overlapping

First component	Second component
D-T	O-T
D-T	O-C
D-C	O-T
D-C	O-C
D-T	OD-T
D-T	OD-C
D-C	OD-T
D-C	OD-C

Table 6.2: Options for first and second components in the sieve-based search.

synonyms (e.g., *C0426579*, *C0003123*, *C1971624* in UMLS all have the synonym *no appetite*) will end up with different vector representations.

Sieve-based search

Traditional sieve-based approaches for concept normalization (D'Souza and Ng, 2015; Jonnagaddala et al., 2016; Luo, Sun, and Rumshisky, 2019b) achieved competitive performance by ordering a sequence of searches over dictionaries from most precise to least precise.

Inspired by this work, we consider a sieve-based similarity search with two components: 1) search over the annotated training data, then 2) search over the ontology (possibly combined with the annotated training data). Table 6.2 lists all possible combinations of first and second components in sieve-based search. For instance, in sieve-based search **D-T + O-C**, we first search over the annotated corpus using training-data-mention vectors (D-T), and then search over the ontology using concept vectors (O-C).

	Scientific Articles			Clinical Notes	
Dataset	NCBI	BC5CDR-D	BC5CDR-C	ShARe/CLEF	MCN
Ontology	MEDIC	MEDIC	CTD-Chemical	SNOMED-CT	SNOMED-CT & RxNorm
# of Concepts (Ontology)	11,915	11,915	171,203	126,524	434,056
# of Synonyms (Ontology)	71,923	71,923	407,247	520,665	1,550,586
# of Documents (Datasets)	792	1,500	1,500	298	100
# of Concepts (Datasets)	750	1,075	1,164	1,313	3,792
# of Mentions (Datasets)	6,881	12,850	15,935	11,167	13,609

Table 6.3: Statistics of 5 datasets in our experiments.

6.3 Experiments

6.3.1 Datasets

We conduct experiments on five datasets with three in the domain of scientific articles, NCBI (Doğan, Leaman, and Lu, 2014), BC5CDR-D and BC5CDR-C (Li et al., 2016b), and two in the domain of clinical notes, MCN (Luo, Sun, and Rumshisky, 2019b) and ShARe/CLEF (Suominen et al., 2013). The statistics of each dataset are described in table 6.3.

NCBI The NCBI disease corpus[3] contains 17,324 manually annotated disorder mentions from 792 PubMed abstracts. The disorder mentions are mapped to 750 MEDIC lexicon (Davis et al., 2012) concepts. We split the released training set into use 5,134 training mentions and 787 development mentions, and keep the 960 mentions from the original test set as evaluation. We use the 2012 version of MEDIC ontology which contains 11,915 concepts and 71,923 synonyms.

BC5CDR-D & BC5CDR-C The BC5CDR-D and BC5CDR-C corpora[4] are used in

[3] https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/
[4] https://biocreative.bioinformatics.udel.edu/tasks/

the BioCreative V chemical-induced disease (CID) relation extraction challenge. BC5CDR-D and BC5CDR-C contain 12,850 disease mentions and 15,935 chemical mentions, respectively. The annotated disease mentions are mapped into 1075 unique concepts out of 11,915 concepts in the 2012 version of MEDIC ontology. The chemical mentions are mapped to 1164 unique concepts out of 171,203 possible concepts from the 2019 version of Comparative Toxicogenomics Database (CTD) chemical ontology. We use the configuration in the BioCreative V challenge to keep the same train/dev/test splits.

ShARe/CLEF The ShARe/CLEF corpus is from the ShARe/CLEF eHealth 2013 Challenge[5], where 11,167 disorder mentions in 298 clinical notes are annotated with their concepts mapping to the 12,6524 disorder concepts from the SNOMED-CT subset of the 2011AA version of UMLS. We take the 199 clinical notes consisting of 5,816 mentions as the train set and 5,351 mentions from the 99 clincal notes as test. Around 30.4% of the mentions in the corpus could not be mapped to any concepts in the ontology, and are assigned the *CUI-less* label.

MCN The MCN corpus used in the 2019 n2c2 Shared-Task track 3[6] consists of 13,609 concept mentions drawn from 100 discharge summaries. The mentions are mapped to 3,792 unique concepts out of 434,056 possible concepts from the SNOMED-CT and RxNorm subset of the 2017AB version of UMLS. We take 40 clinical notes

biocreative-v/

[5]https://sites.google.com/site/shareclefehealth/data

[6]https://n2c2.dbmi.hms.harvard.edu/track3

from the released data as training, consisting of 5,334 mentions, and the standard evaluation data with 6,925 mentions as our test set. Around 2.7% of mentions in MCN are assigned the *CUI-less* label.

6.3.2 Implementation details

Unless specifically noted otherwise, we use the same training procedure and hyper-parameter settings across all experiments and on all datasets. As the triplet mining requires at least one positive text in a batch for each anchor text, we randomly sample one positive text for each anchor text and group them into batches. Like previous work (Schroff, Kalenichenko, and Philbin, 2015; Hermans, Beyer, and Leibe, 2017), we adopt euclidean distance to calculate similarity score during training, while at inference time, we compute cosine similarity as it is simpler to interpret. For the sieve-based search, if the cosine similarity score between the mention and the prediction of the first sieve is above 0.95, we use the prediction of first sieve, otherwise, we use the prediction of the second sieve. When training the triplet network on the combination of the ontology and annotated corpus, we repeat the concept texts in the annotated corpus such that $\frac{|D|}{|O|} = \frac{1}{3}$[7].

We use the pytorch implementation of sentence-transformers[8] to train the Triplet Network for concept normalization. For all experiments, we use PubMed-BERT (Gu et al., 2020) as the starting point, which pre-trains a BERT-style model from scratch on PubMed abstracts and full texts. We use the following hyper-parameters during the training of the

[7] In preliminary experiments we found that large ontologies overwhelmed small annotated corpora. We did not thoroughly explore other ratios here, and leave that to future work.

[8] `https://github.com/UKPLab/sentence-transformers`

triplet network: sequence_length = 8, batch_size = 1500, epoch_size = 100, optimizer = Adam, learning_rate = 3e-5, warmup_steps = 0.

6.3.3 Evaluation metrics

The standard evaluation metric for concept normalization is accuracy because the most similar concept in prediction is of primary interest. For the composite mentions like *breast and ovarian cancer* that are mapped to more than one concept in NCBI, BC5CDR-D, and BC5CDR-C datasets, we adopt the evaluation strategy that composite entity is correct if every prediction for each separate mention is correct (Sung et al., 2020).

6.4 Model selection

We use the development data to choose whether to train the triplet network on just the ontology or also the training data, and to choose which among the similarity search strategies described in section 6.2.2. Table 6.4 shows the performance of all such systems across the five different corpora. The top half of the table focuses on settings where the triplet network only needs to be trained once, on the ontology, and the bottom half focuses on settings where the triplet network is retrained for each new dataset. For each half of the table, the last column gives the average of the ranks of each setting's performance across the five corpora. For example, when training the triplet network only on the ontology, the strategy D-C (search the training data using concept vectors) is almost always the worst performing, ranking 14th of 14 in four corpora and 12th of 14 in one corpus, for an average

	Train	Search	NCBI	BC5CDR-D	BC5CDR-C	ShARe/CLEF	MCN	Avg. Rank
1	O	O-T	83.74	82.65	97.00	82.76	69.11	10.2
2	O	O-C	85.01	82.43	92.62	81.12	70.96	12
3	O	D-T	85.39	77.29	74.21	79.76	61.26	12.6
4	O	D-C	85.26	75.18	74.11	69.70	59.70	13.6
5	O	OD-T	89.58	88.87	97.75	88.12	72.67	4.8
6	O	OD-C	88.56	85.85	93.30	82.23	72.59	9.4
7	O	D-T + O-T	90.34	89.66	97.62	87.26	81.33	3.6
8	O	D-T + O-C	89.96	89.40	96.88	83.73	81.93	5
9	O	D-C + O-T	86.28	83.72	97.14	82.98	76.67	7.4
10	O	D-C + O-C	88.56	83.51	95.77	81.58	76.52	9.8
11	O	D-T + OD-T	91.36	**90.50**	97.64	**90.50**	81.85	2
12	O	D-T + OD-C	90.85	89.90	96.88	84.69	**82.15**	3.6
13	O	D-C + OD-T	**91.99**	89.47	**97.76**	86.83	79.19	3.2
14	O	D-C + OD-C	88.82	86.93	96.32	82.55	77.41	7.6
15	OD	O-T	89.58	87.82	96.71	86.62	72.37	9.8
16	OD	O-C	91.36	89.85	96.32	88.11	80.52	9.6
17	OD	D-T	86.40	79.01	74.23	79.87	63.33	13.2
18	OD	D-C	86.40	78.41	74.23	80.19	62.52	13.4
19	OD	OD-T	91.11	90.38	97.85	88.87	76.15	8.2
20	OD	OD-C	91.61	89.92	96.32	88.33	81.4	7.8
21	OD	D-T + O-T	91.25	91.10	97.81	90.15	84.37	4
22	OD	D-T + O-C	91.49	90.88	96.22	88.76	84.52	6.4
23	OD	D-C + O-T	92.25	90.71	97.87	89.61	83.78	4
24	OD	D-C + O-C	91.49	90.47	96.28	88.65	83.93	7.8
25	OD	D-T + OD-T	91.61	**91.22**	97.81	**90.21**	84.37	2.4
26	OD	D-T + OD-C	91.61	90.83	96.22	89.08	**84.67**	5.2
27	OD	D-C + OD-T	**92.25**	90.95	**97.91**	90.15	83.70	3.4
28	OD	D-C + OD-C	91.61	90.55	96.28	89.40	84.00	5.8

Table 6.4: Dev performances of the triplet network trained on ontology and ontology + data with different similarity search strategies. The last column *Avg. Rank* shows the average rank of each similarity search strategy across multiple datasets. Models with best average rank are highlighted in grey; models with best accuracy are bolded.

rank of 13.6.

Table 6.4 shows that the best models search over both the ontology and the training data. Models that only search over the training data (D-T and D-C) perform worst, with average ranks of 12.6 or higher regardless of what the triplet network is trained on, most likely because the training data covers only a fraction of the concepts in the test data.

Models that only search over the ontology (O-T and O-C) are only slightly better, with average ranks between 9.6 and 12, though the models in the first two rows of the table at least have the advantage that they require no annotated training data (they train on and search over only the ontology).

Table 6.4 also shows that searching based on text (ontology synonyms or training data mentions) vectors typically outperforms searching based on concept (average of text) vectors. Each pair of rows in the table shows such a comparison, and only in the cases of rows 15-16 and 19-20 are the average ranks of the -C models higher than the corresponding -T models.

Finally, table 6.4 shows that sieve-based models outperform their non-sieve-based counterparts. For example, D-T + O-T has better average ranks than O-T, D-T, or OD-T (rows 7 vs. 1, 3, and 5; and rows 21 vs. 15, 17, and 19).

From this analysis on the development set, we select the following models for evaluation on the test set:

Train:O + Search:O-T This is the best approach that requires only the ontology; no annotated training data is used.

Train:O + Search:D-T+OD-T This is the best approach that only needs to be trained once (on the ontology), as the training data is only used to add extra concept text during search time. This is similar to a real-world scenario where a user manually adds some extra domain-specific synonyms for concepts they care about.

Approach	NCBI	BC5CDR-D	BC5CDR-C	ShARe/CLEF	MCN
Sieve-based (D'Souza and Ng, 2015)	84.65			90.75	-
Sieve-based (Luo, Sun, and Rumshisky, 2019b)	-	-	-	-	76.35
TaggerOne (Leaman and Lu, 2016)	88.80	88.9	94.1	-	-
CNN-based ranking (Li et al., 2017)	86.10	-	-	90.30	-
BERT-based ranking (Ji, Wei, and Xu, 2020)	89.06	-	-	**91.10**	-
BERT-based ranking (Xu, Zhang, and Bethard, 2020)	-	-	-	-	83.56
BIOSYN (Sung et al., 2020)	91.1	**93.2**	96.6		-
PubMed-BERT + Search:O-T	76.56	76.60	91.78	73.64	59.97
PubMed-BERT + Search:D-T+OD-T	82.19	90.53	94.24	85.35	75.81
Train:O + Search:O-T	82.60	84.44	95.79	83.48	69.62
Train:O + Search:D-T+OD-T	89.48	92.30	96.67	89.19	82.19
Train:OD + Search:D-T+OD-T	88.96	92.92	96.81	90.41	83.23
Train:OD + Search:tuned	**91.15**	92.82	**96.91**	90.41	**83.70**

Table 6.5: Comparisons of our proposed approaches against the current state-of-the-art performances on *NCBI*, *BC5CDR-D*, *BC5CDR-C*, *ShARe/CLEF*, and *MCN* datasets. Approaches with best accuracy are bolded.

Train:OD + Search:D-T+OD-T This is the best approach that can be created from any combination of ontology and training data. The triplet network must be retrained for each new domain.

Train:OD + Search:tuned This is the bold models in the second half of table 6.4. It requires not only retraining the triplet network for each new domain, but also trying out all search strategies on the new domain and selecting the best one.

6.5 Results

Table 6.5 shows the results of our selected models on test set, alongside the best models in the literature. Our Train:OD+Search:tuned model achieves new state-of-the-art on NCBI, BC5CDR-C, and MCN data, and within 1 point of state-of-the-art on BC5CDR-D and ShARe/CLEF. Yet this model is simpler than most prior work: it requires no two-step

generate-and-rank framework (Li et al., 2017; Ji, Wei, and Xu, 2020; Xu, Zhang, and Bethard, 2020), no iterative candidate retrieval over the entire training data (Sung et al., 2020), no hand-crafted rules or features (D'Souza and Ng, 2015; Leaman and Lu, 2016; Luo, Sun, and Rumshisky, 2019b), and no acronym expansion or TF-IDF transformations (D'Souza and Ng, 2015; Ji, Wei, and Xu, 2020; Sung et al., 2020).

The PubMed-BERT rows in Table 6.5 demonstrate that the triplet training is a critical part of the success: if we use PubMed-BERT without triplet training, performance is 2 to 8 points worse than our best models, depending on the dataset. Yet, we can see that our proposed search strategies are also important, as on the BC5CDR datasets, PubMed-BERT can get within 3 points of the state-of-the-art using the D-T+OD-T search strategy (though it is much further away on the other datasets).

Perhaps most interestingly, our triplet network trained only on the ontology and no annotated training data, Train:O+Search:D-T+OD-T, achieves within 2 points of state-of-the-art on all datasets. We believe this represents a more realistic scenario: unlike prior work, our triplet network does not need to be retrained for each new domain. Instead, the model can be adapted to a new domain by simply pointing out any extra domain-specific synonyms for concepts, and the search can integrate these directly. Domain-specific synonyms do seem to be necessary for all datasets; without them (i.e., Train:O+Search:O-T), performance is about 10 points below state-of-the-art.

As a small qualitative analysis of the models, Table 6.6 shows an example of similarity search results, where the systems have been asked to normalize the mention *primary HPT*.

	PubMed-BERT + Search:OD-T			Train:O + Search:OD-T			Train:OD + Search:OD-T		
Rank	Text	Concept	Score	Text	Concept	Score	Text	Concept	Score
1	HNSCC	C535575	0.9194	Hyperparathyroidism, Primary	D049950	0.7666	Hyperparathyroidism, Primary	D049950	0.8384
5	NPC2	C536119	0.9033	Hyperparathyroidism 1	C564166	0.692	Primary Hyperparathyroidism	D049950	0.83
10	MPNST	D009442	0.9	HRPT1	C564166	0.611	HRPT1	C564166	0.6724
15	HPNS	D006610	0.897	Hyperparathyroidism 2	C563273	0.595	Parathyroid Adenoma, Familial	C564166	0.644
20	PBC2	C567817	0.895	Hyperparathyroidism, Secondary	D006962	0.566	Hyperparathyroidisms, Secondary	D006962	0.608

Table 6.6: The top 20 similar texts, their concepts, and similarity scores for mention *primary HPT* (*D049950*) predicted from models PubMed-BERT + Search:OD-T, Train:O + Search:OD-T and Train:OD + Search:OD-T.

PubMed-BERT fails, producing unrelated acronyms, while both triplet network models find the concept and rank it with the highest similarity score.

6.6 Limitations and future research

Our ability to normalize polysemous concept mentions is limited by their context-independent representations. Although our PubMed-BERT encoder is a pre-trained contextual model, we feed in only the mention text, not any context, when producing a representation vector. This is not ideal for mentions with multiple meanings, e.g., *potassium* in clinical notes may refer to the substance (C0032821) or the measurement (C0202194), and only the context will reveal which one. A better strategy to generate the contextualized representation for the concept mention, e.g., Schumacher, Mulyar, and Dredze (2020), may yield improvements for such mentions.

We currently train a separate triplet network for each ontology (one for MEDIC, one for CTD, one for SNOMED-CT, etc.) but in the future we would like to train on

a comprehensive ontology like the UMLS Metathesaurus (Bodenreider, 2004), which includes nearly 200 different vocabularies (SNOMED-CT, MedDRA, RxNorm, etc.), and more than 3.5 million concepts. We expect such a general vector space model would be more broadly useful to the biomedical NLP community.

CHAPTER 7

Conclusion

In this thesis, we present how we employ ontologies in two IE tasks - temporal normalization and biomedical concept normalization. In both research directions, we leverage resources from ontologies to implement neural network algorithms to map natural language inputs to output spaces that have formally defined semantics.

In Chapter 2, we briefly summarize how prior research works use ontological resources to create rule-based and learning-based IE systems, and model the extracted information, from annotating unstructured text with the components in an ontology, to organizing the extracted information with knowledge representation techniques.

Chapters 3 and 4 present our work on time normalization, where we use the SCATE scheme to present the extracted time expressions. Following the taxonomy of the SCATE schema, we deploy a multi-output neural network architecture, and show it outperforms single-output models that do not consider the structural information of time entities. While keeping the same architecture presented in chapter 3, we further improved our approach by applying the pre-trained contextual character embeddings into our multi-output neural models in chapter 4.

In Chapter 3, we perform a comparison between annotations based on both SCATE and TimeML schema, and find that SCATE can model a wider variety of time expressions.

We also present the first neural network model for time normalization trained on SCATE-style annotations. The model outperforms the rule-based state-of-the-art, proving that describing time expressions in terms of compositional time entities is suitable for machine learning approaches. These findings of using the SCATE scheme for time normalization produce several valuable lessons: 1) an ontology used in ontology informed information extraction (OIIE) system lays the basis for how capable the OIIE system is for translating the extracted information to knowledge, and 2) an appropriate ontology can reduce human labor in developing algorithms for a high-performance and robust OIIE system.

Chapters 5 and 6 present our work on biomedical concept normalization, where we aim to link biomedical entities to their corresponding concepts in the ontology. These two chapters show different angles of how ontological resources can help to deploy neural network algorithms. In Chapter 5, we propose a concept normalization framework consisting of a candidate generator and a list-wise classifier based on BERT. This proposed framework incorporates resources from UMLS, such as preferred terms, synonyms, and semantic types. Specifically, our candidate generator in the N2C2 shared task, a Lucene-based sieve normalization system, combines dictionary look-up over training data, UMLS preferred terms, and their synonyms; and our BERT-based candidate ranker incorporates semantic types through a regularizer which encourages the model to consider the semantic type information of the candidate concepts. In Chapter 6, we propose a framework of using only ontology synonyms and/or training data mentions to train a vector-space model for concept normalization, where mentions and concepts are encoded as vectors

– via transformer networks that are trained via a triplet objective with online hard triplet mining – and mentions are matched to concepts by vector similarity. This framework also produces models trained on only the ontology – no domain-specific training data – that can incorporate domain-specific concept synonyms at search time without retraining, and such models achieve within 2 points of state-of-the-art on all datasets.

All together, we are excited about making contributions to the field of time normalization and concept normalization. We have seen that the automatically generated training data using the permutation of temporal concepts and their semantic relationships from the SCATE scheme (Section 3.2.3), and the contextualized character embeddings derived from the pre-trained language models (Section 4.3) yield performance improvements in time normalization. While the latter strategy of applying contextual embeddings has been widely studied in lots of NLP tasks, but automatically generating new annotations based on ontological resources has not yet been well explored. We hope to encourage more researchers to work in this direction and apply it to other NLP tasks. We also hope our ideas of leveraging synonyms and semantic types in designing neural network algorithms for biomedical concept normalization could provide clues on how to use ontological resources in deep learning.

Another research direction that captures our interest is how to design a joint-learning framework for time normalization and concept recognition (consisting of the named entity recognition and biomedical concept normalization). In time normalization, we decompose the normalization of time expressions into two subtasks; in biomedical concept

normalization, we assume the biomedical entities are pre-identified and only focus on the normalization part. We believe such joint training enables information such as semantic types to be shared between two subtasks and avoids cascading errors.

In terms of OIIE, we are interested in modularizing the usages of the ontology in information extraction. Ontologies are typically domain-dependent, as they contain knowledge about a particular domain. However, knowledge from the unstructured texts is not limited to one domain, especially for the open domain information extraction task where the relation tuples can be *(rabbits, is-a, mammal)* or *(BRCA2 gene, lead-to, breast cancer)*. We believe an ontology-portable OBIE framework will lead us towards building information extraction systems that work across multiple domains, and will be more broadly useful to the NLP community.

Bibliography

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). "Contextual String Embeddings for Sequence Labeling". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1638–1649. URL: http://www.aclweb.org/anthology/C18-1139.

Appelt, Douglas E et al. (1993). "FASTUS: A finite-state processor for information extraction from real-world text". In: *IJCAI*. Vol. 93, pp. 1172–1178.

Aronson, Alan R. (2001). "Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program". In: *Proceedings of the AMIA Symposium*. American Medical Informatics Association, pp. 17–21.

Auer, Sören et al. (2007). "Dbpedia: A nucleus for a web of open data". In: *The semantic web*. Springer, pp. 722–735.

Belousov, Maksim, William Dixon, and Goran Nenadic (2017). "Using an Ensemble of Generalised Linear and Deep Learning Models in the SMM4H 2017 Medical Concept Normalisation Task". In: *CEUR Workshop Proceedings*. Vol. 1996, pp. 54–58. URL: http://ceur-ws.org/Vol-1996/paper10.pdf.

Bethard, Steven (Oct. 2013). "A Synchronous Context Free Grammar for Time Normalization". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 821–826. URL: http://www.aclweb.org/anthology/D13-1078.

Bethard, Steven and Jonathan Parker (May 2016). "A Semantically Compositional Annotation Scheme for Time Normalization". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1. URL:

http://www.lrec-conf.org/proceedings/lrec2016/pdf/288_Paper.pdf.

Bethard, Steven et al. (2015). "SemEval-2015 Task 6: Clinical TempEval". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 806–814. URL: http://www.aclweb.org/anthology/S15-2136.

Bethard, Steven et al. (2016). "SemEval-2016 Task 12: Clinical TempEval". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, pp. 1052–1062. URL: http://www.aclweb.org/anthology/S16-1165.

Bethard, Steven et al. (2017). "SemEval-2017 Task 12: Clinical TempEval". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 565–572. URL: http://www.aclweb.org/anthology/S17-2093.

Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Bochkarev, Vladimir V, Anna V Shevlyakova, and Valery D Solovyev (2015). "The average word length dynamics as an indicator of cultural changes in society". In: *Social Evolution & History* 14.2, pp. 153–175.

Bodenreider, Olivier (2004). "The Unified Medical Language System (UMLS): integrating biomedical terminology". In: *Nucleic Acids Research* 32.suppl_1, pp. D267–D270. DOI: 10.1093/nar/gkh061.

Bohnet, Bernd et al. (2018). "Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2642–2652.

Boiy, Erik and Marie-Francine Moens (2009). "A machine learning approach to sentiment analysis in multilingual Web texts". In: *Information retrieval* 12.5, pp. 526–558.

Bollacker, Kurt et al. (2008). “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pp. 1247–1250.

Buitelaar, Paul et al. (2006). “Ontology-based information extraction with soba”. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Cao, Zhe et al. (2007). “Learning to Rank: From Pairwise Approach to Listwise Approach”. In: *Proceedings of the 24th International Conference on Machine Learning*. Association for Computing Machinery, pp. 129–136. DOI: 10.1145/1273496.1273513.

Chelba, Ciprian et al. (2014). “One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling”. In: *Fifteenth Annual Conference of the International Speech Communication Association*.

Chieu, Hai Leong, Hwee Tou Ng, and Yoong Keok Lee (2003). “Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods”. In: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 216–223.

Chiu, Jason P. C. and Eric Nichols (2016). “Named Entity Recognition with Bidirectional LSTM-CNNs.” In: *Transactions of the Association for Computational Linguistic* 4, pp. 357–370. URL: https://transacl.org/ojs/index.php/tacl/article/view/792.

Chung, Junyoung et al. (2014). “Empirical evaluation of gated recurrent neural networks on sequence modeling”. Version v1. In: *arXiv preprint arXiv:1412.3555v1*.

Ciravegna, Fabio (2001). “Adaptive information extraction from text by rule induction and generalisation.” In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI’01*, pp. 1251–1256.

Collobert, Ronan et al. (Nov. 2011). “Natural Language Processing (Almost) from Scratch”. In: *The Journal of Machine Learning Research* 12, pp. 2493–2537. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2078186.

Contini, Carlo et al. (2020). “The novel zoonotic COVID-19 pandemic: An expected global health concern”. In: *The Journal of Infection in Developing Countries* 14.03, pp. 254–264.

Cui, Hong (2012). “CharaParser for fine-grained semantic annotation of organism morphological descriptions”. In: *Journal of the American Society for Information Science and Technology* 63.4, pp. 738–754.

Cui, Hong et al. (2016). “Introducing Explorer of Taxon Concepts with a case study on spider measurement matrix building”. In: *BMC bioinformatics* 17.1, p. 471.

Cunningham, Hamish (2002). “GATE: A framework and graphical development environment for robust NLP tools and applications”. In: *Proc. 40th annual meeting of the association for computational linguistics (ACL 2002)*, pp. 168–175.

Davis, Allan Peter et al. (2012). “MEDIC: a practical disease vocabulary used at the Comparative Toxicogenomics Database”. In: *Database* 2012.

De Bruijn, Berry and Joel Martin (2002). “Getting to the (c) ore of knowledge: mining biomedical literature”. In: *International journal of medical informatics* 67.1-3, pp. 7–18.

Devlin, Jacob et al. (June 2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

Doddington, George R et al. (2004). “The automatic content extraction (ace) program-tasks, data, and evaluation.” In: *Lrec*. Vol. 2. 1. Lisbon, pp. 837–840.

Doğan, Rezarta Islamaj, Robert Leaman, and Zhiyong Lu (2014). “NCBI disease corpus: a resource for disease name recognition and concept normalization”. In: *Journal of biomedical informatics* 47, pp. 1–10.

D’Souza, Jennifer and Vincent Ng (July 2015). “Sieve-Based Entity Linking for the Biomedical Domain”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association

for Computational Linguistics, pp. 297–302. DOI: 10.3115/v1/P15-2049. URL: https://www.aclweb.org/anthology/P15-2049.

Elhadad, Noémie et al. (2015). "SemEval-2015 task 14: Analysis of clinical text". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 303–310.

Endara, Lorena et al. (2018). "Modifier Ontologies for frequency, certainty, degree, and coverage phenotype modifier". In: *Biodiversity data journal* 6.

Eppler, Martin J and Jeanne Mengis (2004). "The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines". In: *The information society* 20.5, pp. 325–344.

Fabian, MS, K Gjergji, WEIKUM Gerhard, et al. (2007). "Yago: A core of semantic knowledge unifying wordnet and wikipedia". In: *16th International World Wide Web Conference, WWW*, pp. 697–706.

Festag, Sven and Cord Spreckelsen (2017). "Word Sense Disambiguation of Medical Terms via Recurrent Convolutional Neural Networks". In: *Health Informatics Meets EHealth: Digital Insight–Information-Driven Health & Care. Proceedings of the 11th EHealth2017 Conference*. Vol. 236, pp. 8–15.

Fischer, Frank and Jannik Strötgen (2015). "When Does (German) Literature Take Place? On the Analysis of Temporal Expressions in Large Corpora". In: *Proceedings of DH 2015: Annual Conference of the Alliance of Digital Humanities Organizations*. Vol. 6. Sydney, Australia.

Fleuren, Wilco W.M. and Wynand Alkema (2015). "Application of text mining in the biomedical domain". In: *Methods* 74, pp. 97–106. DOI: 10.1016/j.ymeth.2015.01.015.

Ganea, Octavian-Eugen and Thomas Hofmann (2017). "Deep Joint Entity Disambiguation with Local Neural Attention". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2619–2629.

Gangemi, Aldo (2013). “A comparison of knowledge extraction tools for the semantic web”. In: *Extended semantic web conference*. Springer, pp. 351–366.

Gerber, Daniel et al. (2013). “Real-time RDF extraction from unstructured data streams”. In: *International semantic web conference*. Springer, pp. 135–150.

Gillick, Dan et al. (2016). “Multilingual Language Processing From Bytes”. In: *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. Ed. by Kevin Knight, Ani Nenkova, and Owen Rambow. The Association for Computational Linguistics, pp. 1296–1306. URL: http://aclweb.org/anthology/N/N16/N16-1155.pdf.

Gonzalez, Graciela H. et al. (2015). “Recent Advances and Emerging Applications in Text and Data Mining for Biomedical Discovery”. In: *Briefings in Bioinformatics* 17.1, pp. 33–42. DOI: 10.1093/bib/bbv087.

Gordo, Albert et al. (2016). “Deep image retrieval: Learning global representations for image search”. In: *European conference on computer vision*. Springer, pp. 241–257.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 6645–6649.

Gruber, Thomas R (1993). “A translation approach to portable ontology specifications”. In: *Knowledge Acquisition* 5.2, pp. 199–220.

Gruber, Tom (2009). “Ontology”. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, pp. 1963–1965. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_1318. URL: https://doi.org/10.1007/978-0-387-39940-9_1318.

Gu, Yu et al. (2020). “Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing”. In: *arXiv preprint arXiv:2007.15779*.

Gupta, Nitish, Sameer Singh, and Dan Roth (2017). "Entity linking via joint encoding of types, descriptions, and context". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2681–2690.

Hagedorn, Gregor (2007). "Structuring descriptive data of organisms—requirement analysis and information models". PhD thesis.

Han, Benjamin and Alon Lavie (Mar. 2004). "A Framework for Resolution of Time in Natural Language". In: 3.1, pp. 11–32. ISSN: 1530-0226. DOI: 10.1145/1017068.1017070. URL: http://doi.acm.org/10.1145/1017068.1017070.

Han, Sifei et al. (2017). "Team UKNLP: Detecting ADRs, Classifying Medication Intake Messages, and Normalizing ADR Mentions on Twitter". In: *CEUR Workshop Proceedings*. Vol. 1996, pp. 49–53. URL: http://ceur-ws.org/Vol-1996/paper9.pdf.

Hazman, Maryam, Samhaa R El-Beltagy, and Ahmed Rafea (2011). "A survey of ontology learning approaches". In: *International Journal of Computer Applications* 22.9, pp. 36–43.

Hermans, Alexander, Lucas Beyer, and Bastian Leibe (2017). "In defense of the triplet loss for person re-identification". In: *arXiv preprint arXiv:1703.07737*.

Hoffart, Johannes et al. (2011). "Robust disambiguation of named entities in text". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 782–792.

Hoffer, Elad and Nir Ailon (2015). "Deep metric learning using triplet network". In: *International Workshop on Similarity-Based Pattern Recognition*. Springer, pp. 84–92.

Howard, Jeremy and Sebastian Ruder (2018). "Universal language model fine-tuning for text classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1, pp. 328–339.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015). "Bidirectional LSTM-CRF Models for Sequence Tagging". In: *CoRR* abs/1508.01991. arXiv: 1508.01991. URL: http://arxiv.org/abs/1508.01991.

Ji, Heng et al. (2010). “Overview of the TAC 2010 knowledge base population track”. In: *Proceedings of the 2010 Text Analysis Conference*.

Ji, Zongcheng, Qiang Wei, and Hua Xu (2020). “BERT-based Ranking for Biomedical Entity Normalization”. In: *AMIA Summits on Translational Science Proceedings* 2020, 269—277.

Jiang, Xing and Ah-Hwee Tan (2010). “CRCTOL: A semantic-based domain ontology learning system”. In: *Journal of the American Society for Information Science and Technology* 61.1, pp. 150–168.

Jonnagaddala, Jitendra et al. (Aug. 2016). “Improving the dictionary lookup approach for disease normalization using enhanced dictionary and query expansion”. In: *Database* 2016, baw112. ISSN: 1758-0463. DOI: 10.1093/database/baw112. eprint: http://oup.prod.sis.lan/database/article-pdf/doi/10.1093/database/baw112/8224995/baw112.pdf. URL: https://doi.org/10.1093/database/baw112.

Jovanović, Jelena and Ebrahim Bagheri (2017). “Semantic annotation in biomedicine: the current landscape”. In: *Journal of biomedical semantics* 8.1, p. 44.

Jurisica, Igor, John Mylopoulos, and Eric Yu (2004). “Ontologies for knowledge management: an information systems perspective”. In: *Knowledge and Information systems* 6.4, pp. 380–401.

Karimi, Sarvnaz et al. (2015). “Cadec: A corpus of adverse drug event annotations”. In: *Journal of biomedical informatics* 55, pp. 73–81.

Kate, Rohit J. (2016). “Normalizing clinical terms using learned edit distance patterns”. In: *Journal of the American Medical Informatics Association* 23.2, pp. 380–386. DOI: 10.1093/jamia/ocv108.

Khandelwal, Urvashi et al. (2018). “Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: As-

sociation for Computational Linguistics, pp. 284–294. URL: http://aclweb.org/anthology/P18-1027.

Kuru, Onur, Ozan Arkan Can, and Deniz Yuret (Dec. 2016). "CharNER: Character-Level Named Entity Recognition". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 911–921. URL: https://www.aclweb.org/anthology/C16-1087.

Lample, Guillaume et al. (2016). "Neural Architectures for Named Entity Recognition". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 260–270. DOI: 10.18653/v1/N16-1030. URL: http://www.aclweb.org/anthology/N16-1030.

Laparra, Egoitz, Dongfang Xu, and Steven Bethard (2018). "From Characters to Time Intervals: New Paradigms for Evaluation and Neural Parsing of Time Normalizations". In: *Transactions of the Association of Computational Linguistics* 6, pp. 343–356.

Laparra, Egoitz et al. (2018). "SemEval 2018 Task 6: Parsing Time Normalizations". In: *Proceedings of The 12th International Workshop on Semantic Evaluation*, pp. 88–96.

Leal, André, Bruno Martins, and Francisco Couto (June 2015). "ULisboa: Recognition and Normalization of Medical Concepts". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, pp. 406–411. DOI: 10.18653/v1/S15-2070. URL: https://www.aclweb.org/anthology/S15-2070.

Leaman, Robert, Rezarta Islamaj Doğan, and Zhiyong Lu (2013). "DNorm: disease name normalization with pairwise learning to rank". In: *Bioinformatics* 29.22, pp. 2909–2917. DOI: 10.1093/bioinformatics/btt474.

Leaman, Robert, Ritu Khare, and Zhiyong Lu (2015). "Challenges in clinical natural language processing for automated disorder normalization". In: *Journal of biomedical informatics* 57, pp. 28–37.

Leaman, Robert and Zhiyong Lu (2016). “TaggerOne: joint named entity recognition and normalization with semi-Markov Models”. In: *Bioinformatics* 32.18, pp. 2839–2846.

Lee, Hsin-Chun, Yi-Yu Hsu, and Hung-Yu Kao (2016). “AuDis: an automatic CRF-enhanced disease normalization in biomedical text”. In: *Database* 2016. baw091. DOI: 10.1093/database/baw091.

Lee, Jinhyuk et al. (Sept. 2019). “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics*. btz682. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz682. eprint: http://oup.prod.sis.lan/bioinformatics/advance-article-pdf/doi/10.1093/bioinformatics/btz682/30132027/btz682.pdf. URL: https://doi.org/10.1093/bioinformatics/btz682.

Lee, Kathy et al. (2017). “Medical Concept Normalization for Online User-Generated Texts”. In: *2017 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, pp. 462–469. DOI: 10.1109/ICHI.2017.59.

Lee, Kenton et al. (June 2014). “Context-dependent Semantic Parsing for Time Expressions”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 1437–1447. URL: http://www.aclweb.org/anthology/P14-1135.

Li, Fei et al. (2019). “Fine-Tuning Bidirectional Encoder Representations From Transformers (BERT)–Based Models on Large-Scale Electronic Health Record Notes: An Empirical Study”. In: *JMIR Med Inform* 7.3, e14830. DOI: 10.2196/14830.

Li, Haodi et al. (2017). “CNN-based ranking for biomedical entity normalization”. In: *BMC Bioinformatics* 18.11, p. 385. DOI: 10.1186/s12859-017-1805-7.

Li, Jiao et al. (May 2016a). “BioCreative V CDR task corpus: a resource for chemical disease relation extraction”. In: *Database* 2016. baw068. ISSN: 1758-0463. DOI: 10.1093/database/baw068. eprint: http://oup.prod.sis.lan/database/

article-pdf/doi/10.1093/database/baw068/8224483/baw068.pdf. URL: https://dx.doi.org/10.1093/database/baw068.

Li, Jiao et al. (2016b). "BioCreative V CDR task corpus: a resource for chemical disease relation extraction". In: *Database* 2016.

Li, Qi et al. (2013). "A sequence labeling approach to link medications and their attributes in clinical notes and clinical trial announcements for information extraction". In: *Journal of the American Medical Informatics Association* 20.5, pp. 915–921.

Liao, Xiaofeng and Zhiming Zhao (2019). "Unsupervised Approaches for Textual Semantic Annotation, A Survey". In: *ACM Computing Surveys (CSUR)* 52.4, pp. 1–45.

Limsopatham, Nut and Nigel Collier (Aug. 2016). "Normalising Medical Concepts in Social Media Texts by Learning Semantic Representation". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1014–1023. DOI: 10.18653/v1/P16-1096. URL: https://www.aclweb.org/anthology/P16-1096.

Lin, Chen et al. (2015). "Automatic identification of methotrexate-induced liver toxicity in patients with rheumatoid arthritis from the electronic medical record". In: *Journal of the American Medical Informatics Association* 22.e1, e151–e161. DOI: 10.1136/amiajnl-2014-002642. URL: https://doi.org/10.1136/amiajnl-2014-002642.

Liu, Feifan, Abhyuday Jagannatha, and Hong Yu (2019). "Towards Drug Safety Surveillance and Pharmacovigilance: Current Progress in Detecting Medication and Adverse Drug Events from Electronic Health Records." In: *Drug Saf* 42, pp. 95–97. DOI: 10.1007/s40264-018-0766-8.

Liu, Hongwei and Yun Xu (2017). "A Deep Learning Way for Disease Name Representation and Normalization". In: *Natural Language Processing and Chinese Computing*. Springer International Publishing, pp. 151–157. DOI: 10.1007/978-3-319-73618-1_13.

Liu, Yinhan et al. (2019). “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692*.

Llorens, Hector et al. (2012). “TIMEN: An Open Temporal Expression Normalisation Resource.” In: *Language Resources and Evaluation Conference*. European Language Resources Association (ELRA), pp. 3044–3051. ISBN: 978-2-9517408-7-7.

Lu, Zhiyong et al. (2011). “The gene normalization task in BioCreative III”. In: *BMC bioinformatics* 12.8, S2.

Luo, Yen-Fu, Weiyi Sun, and Anna Rumshisky (2019a). “A Hybrid Normalization Method for Medical Concepts in Clinical Narrative using Semantic Matching”. In: *AMIA Summits on Translational Science Proceedings* 2019, p. 732.

— (2019b). “MCN: A comprehensive corpus for medical concept normalization”. In: *Journal of Biomedical Informatics*, pp. 103–132. DOI: 10.1016/j.jbi.2019.103132.

Luo, Yi et al. (2018). “Multi-Task Medical Concept Normalization Using Multi-View Convolutional Neural Network”. In:

Ma, Xiaofei et al. (2019). “Universal Text Representation from BERT: An Empirical Study”. In: *arXiv preprint arXiv:1910.07973*.

Ma, Xuezhe and Eduard Hovy (2016). “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Vol. 1. Association for Computational Linguistics.

Marsh, Elaine and Dennis Perzanowski (1998). “MUC-7 evaluation of IE technology: Overview of results”. In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*.

Martinez-Rodriguez, Jose L, Aidan Hogan, and Ivan Lopez-Arevalo (2020). “Information extraction meets the semantic web: a survey”. In: *Semantic Web* Preprint, pp. 1–81.

Mazur, Pawet and Robert Dale (2010). “WikiWars: A New Corpus for Research on Temporal Expressions”. In: *Proceedings of the 2010 Conference on Empirical Methods*

in Natural Language Processing. EMNLP '10. Cambridge, Massachusetts: Association for Computational Linguistics, pp. 913–922. URL: http://dl.acm.org/citation.cfm?id=1870658.1870747.

McCarthy, John (1980). "Circumscription—a form of non-monotonic reasoning". In: *Artificial intelligence* 13.1-2, pp. 27–39.

Miftahutdinov, Zulfat and Elena Tutubalina (July 2019). "Deep Neural Models for Medical Concept Normalization in User-Generated Texts". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. Florence, Italy: Association for Computational Linguistics, pp. 393–399. DOI: 10.18653/v1/P19-2055. URL: https://www.aclweb.org/anthology/P19-2055.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Misra, Shalini and Daniel Stokols (2012). "Psychological and health outcomes of perceived information overload". In: *Environment and behavior* 44.6, pp. 737–759.

Mondal, Ishani et al. (June 2019). "Medical Entity Linking using Triplet Network". In: *Proceedings of the 2nd Clinical Natural Language Processing Workshop*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, pp. 95–100. DOI: 10.18653/v1/W19-1912. URL: https://www.aclweb.org/anthology/W19-1912.

Morgan, Alexander A et al. (2008). "Overview of BioCreative II gene normalization". In: *Genome biology* 9.2, S3.

Mostafa, Mohamed M (2013). "More than words: Social networks' text mining for consumer brand sentiments". In: *Expert Systems with Applications* 40.10, pp. 4241–4251.

Murty, Shikhar et al. (July 2018). "Hierarchical Losses and New Resources for Fine-grained Entity Typing and Linking". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne,

Australia: Association for Computational Linguistics, pp. 97–109. DOI: 10.18653/v1/P18-1010. URL: https://www.aclweb.org/anthology/P18-1010.

Neculoiu, Paul, Maarten Versteegh, and Mihai Rotaru (Aug. 2016). "Learning Text Similarity with Siamese Recurrent Networks". In: *Proceedings of the 1st Workshop on Representation Learning for NLP*. Berlin, Germany: Association for Computational Linguistics, pp. 148–157. DOI: 10.18653/v1/W16-1617. URL: https://www.aclweb.org/anthology/W16-1617.

Nédellec, Claire (2005). "Ontology and Information Extraction: a necessary symbiosis". In: *Ontology Learning from Text: Methods, Evaluation and Applications* 123, p. 155.

Nguyen, Thanh Ngan, Minh Trang Nguyen, and Thanh Hai Dang (2018). *Disease Named Entity Normalization Using Pairwise Learning To Rank and Deep Learning*. Tech. rep. VNU University of Engineering and Technology.

Niu, Jinghao et al. (2019). "Multi-task Character-Level Attentional Networks for Medical Concept Normalization". In: *Neural Process Lett* 49.3, pp. 1239–1256. DOI: 10.1007/s11063-018-9873-x.

Ono, Toshihide et al. (2001). "Automated extraction of information on protein–protein interactions from the biological literature". In: *Bioinformatics* 17.2, pp. 155–161.

Osborne, John D et al. (2018). "CUILESS2016: a clinical corpus applying compositional normalization of text mentions". In: *Journal of biomedical semantics* 9.1, p. 2.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Peters, Matthew et al. (2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Vol. 1, pp. 2227–2237.

Piskorski, Jakub and Roman Yangarber (2013). “Information extraction: Past, present and future”. In: *Multi-source, multilingual information extraction and summarization*. Springer, pp. 23–49.

Plank, Barbara, Anders Søgaard, and Yoav Goldberg (2016). “Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 412–418. URL: http://anthology.aclweb.org/P16-2067.

Pradhan, Sameer et al. (Aug. 2014). “SemEval-2014 Task 7: Analysis of Clinical Text”. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, pp. 54–62. DOI: 10.3115/v1/S14-2007. URL: https://www.aclweb.org/anthology/S14-2007.

Pustejovsky, James et al. (2003a). “The TimeBank Corpus”. In: *Proceedings of Corpus Linguistics 2003*. Lancaster.

Pustejovsky, James et al. (2003b). “TimeML: Robust Specification of Event and Temporal Expressions in Text”. In: *IWCS-5, Fifth International Workshop on Computational Semantics*.

Pustejovsky, James et al. (2010). “ISO-TimeML: An International Standard for Semantic Annotation”. In: *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC’10)*. Valletta, Malta: European Language Resources Association (ELRA).

Qi, Yanjun et al. (2009). “Combining Labeled and Unlabeled Data with Word-Class Distribution Learning”. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, pp. 1737–1740.

Qi, Ye et al. (2018). “When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation?” In: *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). Vol. 2, pp. 529–535.

Radford, Alec et al. (2018). "Improving language understanding by generative pre-training". In: *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Rao, Delip, Paul McNamee, and Mark Dredze (2013). "Entity linking: Finding extracted entities in a knowledge base". In: *Multi-source, multilingual information extraction and summarization*. Springer, pp. 93–115.

Reimers, Nils and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". 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)*. Hong Kong, China: Association for Computational Linguistics, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. URL: https://www.aclweb.org/anthology/D19-1410.

Riloff, Ellen, Rosie Jones, et al. (1999). "Learning dictionaries for information extraction by multi-level bootstrapping". In: *AAAI/IAAI*, pp. 474–479.

Rindflesch, Thomas C et al. (1999). "EDGAR: extraction of drugs, genes and relations from the biomedical literature". In: *Biocomputing 2000*. World Scientific, pp. 517–528.

Roberts, Kirk et al. (2015). "Automatic extraction and post-coordination of spatial relations in consumer language". In: *AMIA Annual Symposium Proceedings*. Vol. 2015. American Medical Informatics Association, p. 1083.

Roetzel, Peter Gordon (2019). "Information overload in the information age: a review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development". In: *Business Research* 12.2, pp. 479–522.

Rumshisky, Anna et al. (2016). "Predicting early psychiatric readmission with natural language processing of narrative discharge summaries". In: *Transl Psychiatry* 6.10, e921–e921. DOI: 10.1038/tp.2015.182.

Saggion, Horacio et al. (2007). “Ontology-based information extraction for business intelligence”. In: *The Semantic Web*. Springer, pp. 843–856.

Sarker, Abeed et al. (Oct. 2018). “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”. In: *Journal of the American Medical Informatics Association* 25.10, pp. 1274–1283. DOI: 10.1093/jamia/ocy114.

Savova, Guergana K. et al. (2008). “Word sense disambiguation across two domains: Biomedical literature and clinical notes”. In: *Journal of Biomedical Informatics* 41.6, pp. 1088–1100. DOI: 10.1016/j.jbi.2008.02.003.

Savova, Guergana K. et al. (2010). “Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications”. In: *Journal of the American Medical Informatics Association* 17.5, pp. 507–513. DOI: 10.1136/jamia.2009.001560.

Schilder, Frank (Mar. 2004). “Extracting Meaning from Temporal Nouns and Temporal Prepositions”. In: *ACM Transactions on Asian Language Information Processing (TALIP) - Special Issue on Temporal Information Processing* 3.1, pp. 33–50. ISSN: 1530-0226. DOI: 10.1145/1017068.1017071. URL: http://doi.acm.org/10.1145/1017068.1017071.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.

Schumacher, Elliot, Andriy Mulyar, and Mark Dredze (July 2020). “Clinical Concept Linking with Contextualized Neural Representations”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 8585–8592. DOI: 10.18653/v1/2020.acl-main.760. URL: https://www.aclweb.org/anthology/2020.acl-main.760.

Shen, Wei, Jianyong Wang, and Jiawei Han (2014). “Entity linking with a knowledge base: Issues, techniques, and solutions”. In: *IEEE Transactions on Knowledge and Data Engineering* 27.2, pp. 443–460.

Smith, Andrew and Miles Osborne (2006). “Using gazetteers in discriminative information extraction”. In: *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pp. 133–140.

Smith, Barry and Christopher Welty (2001). “Ontology: Towards a new synthesis”. In: *Formal Ontology in Information Systems*. Vol. 10. 3. ACM Press, pp. 3–9.

Stevenson, Mark et al. (2008). “Knowledge sources for word sense disambiguation of biomedical text”. In: *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*. Association for Computational Linguistics, pp. 80–87.

Stevenson, Mark et al. (June 2009). “Disambiguation of Biomedical Abbreviations”. In: *Proceedings of the BioNLP 2009 Workshop*. Boulder, Colorado: Association for Computational Linguistics, pp. 71–79. URL: https://www.aclweb.org/anthology/W09-1309.

Strötgen, Jannik and Michael Gertz (2013). “Multilingual and cross-domain temporal tagging”. English. In: *Language Resources and Evaluation* 47.2, pp. 269–298. ISSN: 1574-020X. DOI: 10.1007/s10579-012-9179-y. URL: http://dx.doi.org/10.1007/s10579-012-9179-y.

— (2015). “A Baseline Temporal Tagger for all Languages”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 541–547. URL: http://aclweb.org/anthology/D15-1063.

Strötgen, Jannik, Julian Zell, and Michael Gertz (2013). “Heideltime: Tuning English and developing Spanish resources for TempEval-3”. In: *Proceedings of the Seventh International Workshop on Semantic Evaluation, SemEval ’13*. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 15–19.

Studer, Rudi, V Richard Benjamins, and Dieter Fensel (1998). “Knowledge engineering: principles and methods”. In: *Data & knowledge engineering* 25.1-2, pp. 161–197.

Sun, Yaming et al. (2015). “Modeling mention, context and entity with neural networks for entity disambiguation.” In: *IJCAI*. Vol. 15, pp. 1333–1339.

Sundheim, Beth M (1995). *Overview of results of the MUC-6 evaluation*. Tech. rep. NAVAL COMMAND CONTROL and OCEAN SURVEILLANCE CENTER SAN DIEGO CA.

Sung, Mujeen et al. (July 2020). “Biomedical Entity Representations with Synonym Marginalization”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 3641–3650. DOI: 10.18653/v1/2020.acl-main.335. URL: https://www.aclweb.org/anthology/2020.acl-main.335.

Suominen, Hanna et al. (2013). “Overview of the ShARe/CLEF eHealth Evaluation Lab 2013”. In: *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*. Springer Berlin Heidelberg, pp. 212–231. DOI: 10.1007/978-3-642-40802-1_24.

Surdeanu, Mihai et al. (2011). “Customizing an information extraction system to a new domain”. In: *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pp. 2–10.

Thelen, Michael and Ellen Riloff (2002). “A bootstrapping method for learning semantic lexicons using extraction pattern contexts”. In: *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP 2002)*, pp. 214–221.

Toutanova, Kristina et al. (2003). “Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network”. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL ’03. Edmonton, Canada: Association for Computational Linguistics, pp. 173–180. DOI: 10.3115/1073445.1073478. URL: https://doi.org/10.3115/1073445.1073478.

Tu, Hongkui et al. (2016). “When MetaMap Meets Social Media in Healthcare: Are the Word Labels Correct?” In: *Asia Information Retrieval Symposium*. Springer, pp. 356–362.

Tutubalina, Elena et al. (2018). “Medical concept normalization in social media posts with recurrent neural networks”. In: *Journal of Biomedical Informatics* 84, pp. 93–102. DOI: 10.1016/j.jbi.2018.06.006.

Unbehauen, Jörg et al. (2012). “Knowledge extraction from structured sources”. In: *Search computing*. Springer, pp. 34–52.

Uren, Victoria et al. (2006). “Semantic annotation for knowledge management: Requirements and a survey of the state of the art”. In: *Journal of Web Semantics* 4.1, pp. 14–28.

Uzuner, Özlem et al. (2011). “2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text”. In: *Journal of the American Medical Informatics Association* 18.5, pp. 552–556. DOI: 10.1136/amiajnl-2011-000203.

UzZaman, Naushad et al. (June 2013). “SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations”. In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA: Association for Computational Linguistics, pp. 1–9. URL: https://www.aclweb.org/anthology/S13-2001.

Valenzuela-Escárcega, Marco A, Gus Hahn-Powell, and Mihai Surdeanu (2016). “Odin’s runes: A rule language for information extraction”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pp. 322–329.

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *Advances in neural information processing systems* 30, pp. 5998–6008.

Verhagen, Marc et al. (2007). “SemEval-2007 Task 15: TempEval Temporal Relation Identification”. In: *Proceedings of the 4th International Workshop on Semantic Evaluations*. SemEval ’07. Prague, Czech Republic, pp. 75–80.

Verhagen, Marc et al. (July 2010). “SemEval-2010 Task 13: TempEval-2”. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, pp. 57–62. URL: http://www.aclweb.org/anthology/S10-1010.

Vossen, Piek et al. (2016). “NewsReader: Using knowledge resources in a cross-lingual reading machine to generate more knowledge from massive streams of news”. In: *Special Issue Knowledge-Based Systems, Elsevier*. ISSN: 0950-7051. DOI: dx.doi.org/10.1016/j.knosys.2016.07.013. URL: http://www.sciencedirect.com/science/article/pii/S0950705116302271.

Weissenbacher, Davy et al. (Aug. 2019). “Overview of the Fourth Social Media Mining for Health (SMM4H) Shared Tasks at ACL 2019”. In: *Proceedings of the Fourth Social Media Mining for Health Applications (#SMM4H) Workshop & Shared Task*. Florence, Italy: Association for Computational Linguistics, pp. 21–30. DOI: 10.18653/v1/W19-3203. URL: https://www.aclweb.org/anthology/W19-3203.

Wimalasuriya, Daya C and Dejing Dou (2010). “Ontology-based information extraction: An introduction and a survey of current approaches”. In: *Journal of Information Science* 36.3, pp. 306–323.

Wong, Wilson, Wei Liu, and Mohammed Bennamoun (2012). “Ontology learning from text: A look back and into the future”. In: *ACM Computing Surveys (CSUR)* 44.4, pp. 1–36.

Wu, Wentao et al. (2012). “Probase: A probabilistic taxonomy for text understanding”. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, pp. 481–492.

Xia, Fen et al. (2008). “Listwise approach to learning to rank: theory and algorithm”. In: *Proceedings of the 25th International Conference on Machine Learning*. Association for Computing Machinery, pp. 1192–1199. DOI: 10.1145/1390156.1390306.

Xu, Dongfang, Egoitz Laparra, and Steven Bethard (2019). “Pre-trained Contextualized Character Embeddings Lead to Major Improvements in Time Normalization: a Detailed Analysis”. In: *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM 2019)*, pp. 68–74.

Xu, Dongfang, Zeyu Zhang, and Steven Bethard (July 2020). “A Generate-and-Rank Framework with Semantic Type Regularization for Biomedical Concept Normalization”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 8452–8464. URL: https://www.aclweb.org/anthology/2020.acl-main.748.

Xu, Dongfang et al. (2018). “Resolving “orphaned” non-specific structures using machine learning and natural language processing methods”. In: *Biodiversity data journal* 6.

Xu, Dongfang et al. (July 2020). “Unified Medical Language System resources improve sieve-based generation and Bidirectional Encoder Representations from Transformers (BERT)–based ranking for concept normalization”. In: *Journal of the American Medical Informatics Association* 27.10, pp. 1510–1519. DOI: 10.1093/jamia/ocaa080. URL: https://doi.org/10.1093/jamia/ocaa080.

Xu, Hua, Peter D Stetson, and Carol Friedman (2007). “A study of abbreviations in clinical notes”. In: *AMIA annual symposium proceedings*. Vol. 2007. American Medical Informatics Association, p. 821.

Xu, Yan et al. (2012). “Feature engineering combined with machine learning and rule-based methods for structured information extraction from narrative clinical discharge summaries”. In: *Journal of the American Medical Informatics Association* 19.5, pp. 824–832.

Yepes, Antonio Jimeno (2017). “Word embeddings and recurrent neural networks based on Long-Short Term Memory nodes in supervised biomedical word sense disambiguation”. In: *Journal of biomedical informatics* 73, pp. 137–147.

Yin, Xiaoyao et al. (2019). “Deep Entity Linking via Eliminating Semantic Ambiguity With BERT”. In: *IEEE Access* 7, pp. 169434–169445.

Yosef, Mohamed Amir et al. (2011). “Aida: An online tool for accurate disambiguation of named entities in text and tables”. In: *Proceedings of the VLDB Endowment* 4.12, pp. 1450–1453.

Zhang, Kelly and Samuel Bowman (2018). “Language Modeling Teaches You More than Translation Does: Lessons Learned Through Auxiliary Syntactic Task Analysis”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 359–361.

Zhang, Xiang, Junbo Zhao, and Yann LeCun (2015). “Character-level Convolutional Networks for Text Classification”. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., pp. 649–657. URL: http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf.