Information Extraction From User Generated Noisy Texts

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Jeniya Tabassum Binte Jafar

Graduate Program in Department of Computer Science and Engineering

The Ohio State University

2020

Dissertation Committee:

Wei Xu, Alan Ritter, Feng Qin, Co-Advisor

Micha Elsner

Eric Fosler-Lussier

Abstract

Social media websites provide an ideal environment for people to express their experiences on the latest events and share their knowledge about the current technologies along with research advancements. This presents an opportunity for Natural Language Processing (NLP) and Information Extraction (IE) technology to facilitate large scale data-analysis applications by extracting machine-processable information from user generated unstructured texts. However, information extraction from social media is particularly challenging due to the inherent noise induced by different writing styles of its users and their writing errors such as: typos and non-grammatical sentences. In this thesis, we explore the supervised and semi-supervised approaches to extract structured information from the noisy user generated texts of three widely used social web spaces: Twitter, StackOverflow and ProtocolIO.

Twitter is one of the most popular micro-blogging sites that enables people to share the information about different events happening around the world. To extract the time information of the events mentioned in a Twitter post, we propose a minimally supervised temporal resolver that does not require any in-domain annotations or hand-crafted rules. Our proposed time entity resolver learns from the large quantity of unlabeled text in conjunction with a database of known events. Hence, it is capable of learning the robust time expressions from the informal style of text found on social media without any human-annotated data.

Our proposed minimally supervised model outperforms a broad range of state-of-the-art systems for the task of resolving date expressions.

StackOverflow is a question-answering site where programmers around the world help each other to resolve their programming related queries. To extract the information from the code-mixed StackOverflow posts, we propose a code and name-entity recognizer, tailored for the software domain. We present a comprehensive study that investigates the unique challenges of named entity recognition of this domain. Our proposed named entity recognizer (NER) utilizes an attention network to combine the local textual context with the corpus level code snippet knowledge. Using our newly annotated corpus of 15,372 sentences in StackOverflow, we rigorously test our proposed SoftNER model, which outperforms BiLSTM-CRF model and fine-tuned BERT model for identifying 20 types of software-related named entities. We also demonstrate that our NER tagger is capable of extracting named entities and in-line codes with reasonable performance from other social computer programming domain texts such as GitHub, even without any in-domain training-data.

ProtocolsIO is a research-recipe sharing site where experimental scientists around the world can share detailed guidelines of their research methods. To extract the relation and entity information from these noisy ProtocolsIO texts, we conducted a shared task with a newly annotated corpus. Our corpus consists of 17,658 sentences collected from 726 wet lab protocols. Using our annotated corpus, we analyze the results of the 15 participating systems, which in turns aided us to draw conclusions about the potential of different approaches for the noisy user generated lab protocols.

Dedicated to my loving husband, Shahadat Sohel.

Acknowledgments

My PhD life at The Ohio State University has been a memorable journey. As with all the PhD stories, it has its ups and down, but I have been fortunate to be surrounded by an excellent group of people, who helped me survive the challenges that came along.

I would like to thank my advisors, Prof. Alan Ritter and Wei Xu for their mentoring. I want to express my heartiest gratitude to Prof. Feng Qin to help me through out the exam procedures. I want to thank my defense committee members, Prof. Micha Elsner and Prof. Eric Fosler and for their time and helpful feedback on my research. Along with the discussions about the research, I also learned a lot from their courses. Their vast knowledge about information extraction from natural language helped me to get better understanding of my research work. I also want to thank Prof. Marie-Catherine de Marneffe for her wonderful course on 'Analyzing language in social media' at OSU and also for all the meaningful discussion we had on social media research.

Thanks to all the members of Xu-Ritter group for being so awesome. Special thanks to Sandesh Swamy, Mounica Maddela, Ashutosh Baheti, Fan Bai and Wuwei Lan for being there for me whenever I needed. I am grateful to all of my friends in the Columbus, for making me feel at home throughout my graduate life.

Lastly, I would like to thank my partner, Sohel for being patient and understanding during the difficult times. I would never be able to complete my PhD journey without you by my side.

Vita

2012 ..... BS, Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh

2020 ..... PhD, Computer Science and Engineering, The Ohio State University, USA.

Publications

Research Publications

Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. 2020. Code and Named Entity Recognition in StackOverflow. In *Proceedings of the 2020 Annual Meeting of the Association for Computational Linguistic.*

Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. 2020. WNUT-2020 Task 1: Extracting Entities and Relations from Wet Lab Protocols. In *Proceedings of the 2020 Workshop on Noisy User-generated Text.*

Jeniya Tabassum, Alan Ritter, and Wei Xu. 2016. Tweetime: A Minimally Supervised Method for Recognizing and Normalizing Time Expressions in Twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.*

Fields of Study

Major Field: Computer Science and Engineering

Studies in:

Artificial Intelligence	Prof. Alan Ritter
Natural Language Processing	Prof. Wei Xu

Contents

List of Tables

List of Figures

Figure **Page**

Chapter 1: Introduction

Social media facilitates the creation and sharing of information among people across the globe with its virtual communities and networks. Thus, these social media is a continuously growing storage of collective human knowledge. However, due to the quick and real-time content-sharing nature of the social networks, the shared information is often chaotic and unorganized. This makes detection and extraction of structured information from these user-generated text streams quite challenging. In this thesis, we present a comprehensive study that investigates the unique challenges of named entity recognition from such user generated texts and propose supervised and semi-supervised models to learn the entities from the noisy texts of three widely used social domains: Twitter (§1.1), StackOverflow (§1.2) and ProtocolsIO (§1.3) .

1.1 Temporal Entity Extraction from Tweet

Temporal entities are words or phrases that refer to dates, times or duration. Resolving such time entities is an important task in information extraction (IE) that enables downstream applications such as event timeline extraction (Derczynski and Gaizauskas, 2013; Do et al., 2012; Ritter et al., 2012; Ling and Weld, 2010), knowledge base population (Ji et al., 2011), information retrieval (Alonso et al., 2007), automatically scheduling meetings from email

and more. Previous work in this area has applied rule-based systems (Mani and Wilson, 2000; Bethard, 2013b; Chambers, 2013) or supervised machine learning on small collections of hand-annotated news documents (Angeli et al., 2012; Lee et al., 2014) which are designed for well-edited text. These supervised and rule-based time resolvers suffer from reduced performance on social media data due to domain mismatch. In Chapter 2, we present TweeTIME, a minimally supervised model that learns the time entities from large quantities of unlabeled data (Tabassum et al., 2016). TweeTIME extracts the time entities without using any hand-engineered rules or hand-annotated training corpora, which allows it to successfully extract the immense amount of diversified and noisy time entities from the tweets.

We would like to note that resolving time entities from social media text is a non-trivial problem. Besides many spelling variations, time expressions are more likely to refer to future dates than in newswire. For the example consider the tweet: "Watch Mercury infornt of SUN on Monday", we need to recognize that *Monday* refers to the upcoming Monday and not the previous one, in order to map the event of this tweet to its correct normalized date. We also need to identify that the word *Sun* is not referring to a Sunday in this context. To extract and normalize such time entities, we designed TweeTIME, the first time resolver for social media text. TweeTIME achieves a 0.68 F_1 score on the end-to-end task of resolving date expressions, outperforming a broad range of state-of-the-art systems[1].

1.2 Software Entity Extraction from StackOverflow

Software entities are the words and phrases that represent the atomic-informative terms associated with programming related entities inside the code-mixed natural language texts.

[1]Our code and data are available at https://github.com/jeniyat/TweeTime

Extraction of such software domain named entities could help a wide range of downstream applications. For example, extracting software knowledge bases from text (Movshovitz-Attias and Cohen, 2015), developing better quality measurements of StackOverflow posts (Ravi et al., 2014), finding similar questions (Amirreza et al., 2019) and more. However, there is a lack of NLP resources and techniques for identifying software-related named entities (e.g., variable names or application names) within natural language texts. In Chapter 3, we present a new named entity recognition (NER) corpus for the computer programming domain, consisting of 15,372 sentences annotated with 20 fine-grained entity types, which is the first fined grained NER corpus for software domain, to the best of our knowledge.

The extraction of software-related named entities is quite challenging as texts of this domain contain significant amount of unseen tokens with inherent polysemy and salient reliance on the context. Unlike news or biomedical data, spelling patterns and long-distance dependencies are more crucial in the software domain to resolve ambiguities and categorize unseen words. Taken in isolation, many tokens are highly ambiguous and can refer to either programming concepts or common English words, such as: '*go*', '*react*', '*spring*', '*while*', '*if*', '*select*'. To address these challenges, we propose a software-related named entity recognizer (SoftNER) that utilizes an attention network to combine the local sentence-level context with corpus-level information extracted from the code snippets (Tabassum et al., 2020b). Our proposed NER tagger outperforms the fine-tuned in-domain $\text{BERT}_{\text{base}}$ model with an absolute increase of +10.98 F_1 score for task of identifying fine-grained software named entities.[2]

[2]Our code and data are available at: https://github.com/jeniyat/StackOverflowNER/.

1.3 Protocol Entity an Relation Extraction from Wet Lab

Protocol entities are the words and phrases that refer to constituents and actions inside the procedural texts of laboratory instructions. Extracting such protocol related entities along with the relations that connects them could enable creation of machine-readable instructions (King et al., 2009; Ananthanarayanan and Thies, 2010; Soldatova et al., 2014; Vasilev et al., 2011), which in turn would aid in avoiding mistakes due to human error. This will enhance the reproducibility of experimental biological research (Bates et al., 2017). This motivates the need for NLP resources to extract and identify the protocol entities and their relations from the the noisy wet lab instructions. In Chapter 4, we present a named entity recognition (NER) and relation extraction (RE) corpus for these lab protocols (Tabassum et al., 2020a), consisting of 17,658 sentences annotated with 20 fine-grained entities and 30 relation types, which is the largest corpus for wet lab domain, to the best of our knowledge. [3]

Extraction of entity and relations form the ProtocolsIO text are difficult as they contain experimental jargon and colloquial language constructs that emerge as a byproduct of ad-hoc protocol documentation. Hence, performance of state-of-the-art tools for extracting named entity and relations from these lab protocols still lags behind well-edited text genres (Jiang et al., 2020). This persuades the need for continued research for creating new datasets and tools adapted to this noisy text genre, and motivates us to organize a shared task on named entity and relation extraction from the noisy wet lab protocols. In Chapter 4, we present the overview of the shared task on wet lab protocols along with a summary of the systems developed by selected teams. We provide a detailed analysis on the performance of the

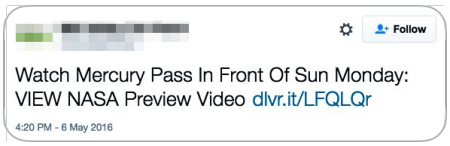
[3]Our annotated corpus is available at: https://github.com/jeniyat/WNUT_2020_NER/.

participating systems, which aided us to suggest potential research directions for both future shared tasks and noisy text processing in user generated lab protocols.

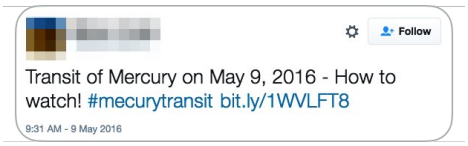
Chapter 2: TweeTime: A Minimally Supervised Method for Recognizing and Normalizing Time Entities in Twitter

Social media especially contains time-sensitive information and requires accurate temporal analysis, for many downstream tasks including detecting real-time cyber-security events (Ritter et al., 2015; Chang et al., 2016), determine disease outbreaks (Kanhabua et al., 2012) and extracting personal information (Schwartz et al., 2015). However, most of the works on social media suffer from sub-optimal performance because of using the generic temporal resolvers. Basically resolving time entities in social media is a non-trivial problem. Besides many spelling variations, time entities are more likely to refer to future dates than in newswire. For the example in Figure 2.1(a), we need to recognize that *Monday* refers to the upcoming Monday and not the previous one to resolve to its correct normalized date (*5/9/2016*). We also need to identify that the word *Sun* is not referring to a Sunday in this context.

In this chapter, we present a novel minimally supervised approach to extract these challenging temporal entities withour any in-domain annotation or hand-crafted rules. Our proposed model learns from large quantities of unlabeled text in conjunction with a database of known events and hence it is capable of learning robust time entities from the informal social media texts.



(a)



(b)

Figure 2.1: Figure 2.1(b) presents a tweet that contains simple explicit time mention and an event (*Mercury, 5/9/2016*) that can be identified by an open-domain information extraction system.), which a generic temporal tagger failed to resolve correctly. Figure 2.1(a) presents a tweet that contains simple explicit time mention and an event (*Mercury, 5/9/2016*) that can be identified by an open-domain information extraction system.

For popular events, some related tweets (e.g. Figure 2.1(b)) may contain explicit or other simple time mentions that can be captured by a generic temporal tagger. An open-domain information extraction system (Ritter et al., 2012) can then identify events (e.g. [*Mercury*, *5/9/2016*]) mentioned in those tweets. To automatically generate temporally annotated data for training, we make the following novel *distant supervision assumption:*

> Tweets posted near the time of a known event that mention central entities are likely to contain time entities that refer to the date of the event.

Based on this assumption[4], tweets that contain the same named entity (e.g. 'Mercury' in Figure 2.1(a)) are heuristically labeled as training data. Each tweet is associated with multiple overlapping labels that indicate the day of the week, day of the month, whether the event is in the past or future and other time properties of the event date in relation to the tweet's creation date. In order to learn a tagger that can recognize temporal entities at the word-level, we present a multiple-instance learning approach to model sentence and word-level tags jointly and handle overlapping labels. Using heuristically labeled data and

[4]We focus on resolving dates, arguably the most important and frequent category of time entities in social media data, and leave other phenomenon such as times and durations to traditional methods or future work.

the temporal tags predicted by the multiple-instance learning model as input, we then train a log-linear model that normalizes time entities to calendar dates.

Building on top of the multiple-instance learning model, we further improve performance using a missing data model that addresses the errors introduced during the heuristic labeling process. Our best model achieves a 0.68 F1 score when resolving date mentions in Twitter. This is a 17% increase over SUTime (Chang and Manning, 2012), outperforming other state-of-the-art time entity resolvers HeidelTime (Strötgen and Gertz, 2013), TempEX (Mani and Wilson, 2000) and UWTime (Lee et al., 2014) as well. Our approach also produces a confidence score that allows us to trade recall for precision. To the best of our knowledge, TweeTIME is the first time resolver designed specifically for social media data.[5] This is also the first time that distant supervision is successfully applied for end-to-end temporal entity recognition and normalization. Previous distant supervision approaches (Angeli et al., 2012; Angeli and Uszkoreit, 2013) only address the normalization problem, assuming gold time mentions are available at test time.

2.1 TweeTIME: Time Entity Resolver for Tweet

Our TweeTIME system, consists of two major components namely **Temporal Recognizer** and **Temporal Normalizer**, is shown in Figure 2.2.

1. A **Temporal Recognizer** which identifies time expressions (e.g. *Monday*) in English text and outputs 5 different temporal types (described in Table 2.1) indicating timeline direction, month of year, date of month, day of week or no temporal information (NA).

[5]The closest work is HeidelTime's colloquial English version (Strötgen and Gertz, 2012) developed from annotated SMS data and slang dictionary. Our TweeTIME significantly outperforms on Twitter data.

It is realized as a multiple-instance learning model, and in an enhanced version, as a missing data model.

2. A **Temporal Normalizer** that takes a tweet with its creation time and temporal expressions tagged by the above step as input, and outputs their normalized forms (e.g. *Monday* $\rightarrow$ *5/9/2016*). It is a log-linear model that uses both lexical features and temporal tags.

To train these two models without any manually annotated corpora, we leverage a large database of known events as distant supervision source. The event database is extracted automatically from tweets using an open-domain event-extraction system (Ritter et al., 2012). Each event consists of one or more named entities, in addition to the date on which the event takes place, for example [*Mercury*, *5/9/2016*]. Tweets are first processed by a Twitter named entity recognizer (Ritter et al., 2011b), and a generic date resolver (Mani and Wilson, 2000). Events are then extracted based on the strength of association between each named entity and calendar date, as measured by a G^2 test on their co-occurrence counts (details in §2.2.1). The following two sections describe the model architecture of our Temporal Recognizer (§2.1.1) and Temporal Normalizer (§2.1.2) separately.

2.1.1 Temporal Entity Recognition with Distant Supervision

The goal of the temporal entity recognizer is to predict the temporal tag of each word, given a sentence (or a tweet) $\mathbf{w} = w_1, \ldots, w_n$. We propose a multiple-instance learning model and a missing data model that are capable of learning word-level tags from the given sentence-level labels.

Temporal Types	Possible Values (tags)
Timeline (TL)	$past, present, future$
Day of Week (DOW)	$Mon, Tue, \ldots, Sun$
Day of Month (DOM)	*1, 2, 3, . . . , 31*
Month of Year (MOY)	$Jan, Feb, \ldots, Dec$
None (NA)	NA

Table 2.1: Our Temporal Recognizer can extract five different temporal types and assign one of their values to each word of a tweet.

Our recognizer module in is built using a database of known events as *distant supervision* source. We assume tweets published around the time of a known event that mention a central entity are also likely to contain time entities referring to the event's date. For each event, such as [*Mercury*, *5/9/2016*], we gather all tweets that contain the central entity *Mercury* and are posted within 7 days of *5/9/2016*. We then label each tweet based on the event date in addition to the tweet's creation date. As both of the tweets Figure 2.1 are referencing the *Mercury* event, we will extract their corresponding sentence-level temporal tags from the different of the tweet's creation date and the event date *5/9/2016*. Hence, the sentence-level temporal tags for the tweet in Figure 2.1(a) are: TL=$future$, DOW=Mon, DOM=9,

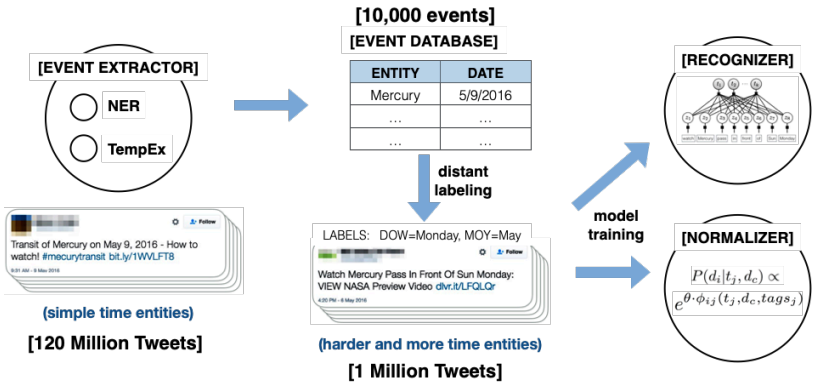


Figure 2.2: TweeTIME system diagram of model training.

MOY=May where for the tweet in Figure 2.1(b) the tags are: TL=$present$, DOM=9, and MOY=May.

2.1.1.1 Multiple-Instance Learning Temporal Tagging Model (MultiT)

Unlike supervised learning, where labeled instances are provided to the model, in multiple instance learning scenarios (Dietterich et al., 1997), the model is only provided with bags of instances labeled as either positive (where at least one instance is positive) or all negative. This is a close match to our problem setting, where we do not have the training label for each word, instead, we have labels for each sentence. We tagged each sentence with five different temporal properties according to the *distant supervision assumption*.

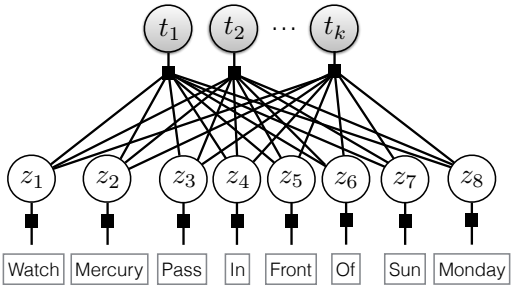


Figure 2.3: Multiple-Instance Learning Temporal Entity Tagger. It learns a word-level tagging model from the given sentence-level labels. In this example a sentence-level variable $t_a = 1$ indicates the temporal tag DOW=Mon must be present and $t_b = 1$ indicates that the target date is in the future (TL=$future$). The multiple instance learning assumption implies that at least one word must be tagged with each of these present temporal tags. For example, ideally after training, the model will learn to assign z_8 to tag a and z_1 to tag b.

We represent sentences and their labels using a graphical model that is divided into word-level and sentence-level variables (as shown in Figure 2.3). Unlike the standard supervised tagging problem, we never directly observe the words' tags ($\mathbf{z} = z_1, \ldots, z_n$) during learning. Instead, they are latent and we only observe the date of an event mentioned in the text, from

which we derive sentence-level binary variables $\mathbf{t} = t_1, \ldots, t_k$ corresponding to temporal tags for the sentence. Following previous work on multiple-instance learning (Hoffmann et al., 2011a; Xu et al., 2014), we model the connection between sentence-level labels and word-level tags using a set of deterministic-OR factors ϕ^{sent}.

The overall conditional probability of our model is defined as:

$$P(\mathbf{t}, \mathbf{z}|\mathbf{w}; \boldsymbol{\theta^r}) = \frac{1}{Z} \prod_{i=1}^{k} \phi^{sent}(t_i, \mathbf{z}) \times \prod_{j=1}^{n} \phi^{word}(z_j, w_j) = \frac{1}{Z} \prod_{i=1}^{k} \phi^{sent}(t_i, \mathbf{z}) \times \prod_{j=1}^{n} e^{\boldsymbol{\theta^r} \cdot \mathbf{f}(z_j, w_j)} \tag{2.1}$$

where $\mathbf{f}(z_j, w_j)$ is a feature vector and

$$\phi^{sent}(t_i, \mathbf{z}) = \begin{cases} 1 & \text{if } t_i = true \wedge \exists j : z_j = i \\ 1 & \text{if } t_i = false \wedge \forall j : z_j \neq i \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

We include a standard set of tagging features that includes word shape and identity in addition to prefixes and suffixes. To learn parameters $\boldsymbol{\theta^r}$ of the Temporal Tagger, we maximize the likelihood of the sentence-level heuristic labels conditioned on observed words over all tweets in the training corpus. Given a training instance **w** with label **t**, the gradient of the conditional log-likelihood with respect to the parameters is:

$$\nabla P(\mathbf{t}|\mathbf{w}) = \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{w}, \mathbf{t}; \boldsymbol{\theta^r}) \cdot \mathbf{f}(\mathbf{z}, \mathbf{w}) - \sum_{\mathbf{t},\mathbf{z}} P(\mathbf{t}, \mathbf{z}|\mathbf{w}; \boldsymbol{\theta^r}) \cdot \mathbf{f}(\mathbf{z}, \mathbf{w}) \tag{2.3}$$

This gradient is the difference of two conditional expectations over the feature vector **f**: a "clamped" expectation that is conditioned on the observed words and tags (**w**, **t**) and

a "free" expectation that is only conditioned on the words in the text, **w**, and ignores the sentence-level labels. To make the inference tractable, we use a Viterbi approximation that replaces the expectations with maximization. Because each sentence corresponds to more than one temporal tag, the maximization of the "clamped" maximization is somewhat challenging to compute. We use the approximate inference algorithm of Hoffmann et al. (2011a), that views inference as a weighted set cover problem, with worst case running time $(|T| \cdot |W|)$, where $|T|$ is the number of all possible temporal tag values and $|W|$ is the number of words in a sentence.

2.1.1.2 Missing Data Temporal Tagging Model (MiDaT)

While the multiple-instance learning assumption works well much of the time, it can easily be violated – there are many tweets that mention entities involved in an event but that never explicitly mention its date.

The missing data modeling approach to weakly supervised learning (Ritter et al., 2013) addresses this problem by relaxing the hard constraints of deterministic-OR factors, such as those described above, as soft constraints. Our missing-data model for weakly supervised tagging splits the sentence-level variables, t into two parts : m which represents whether a temporal tag is mentioned by at least one word of the tweet, and t' which represents whether a temporal tag can be derived from the event date. A set of pairwise potentials $\psi(m_j, t'_j)$ are introduced that encourage (but don't strictly require) agreement between m_j and t'_j, that is:

$$\psi(m_j, t'_j) = \begin{cases} \alpha_p, \text{ if } t'_j \neq m_j \\ \alpha_r, \text{ if } t'_j = m_j \end{cases} \tag{2.4}$$

Here, α_p (Penalty), and α_r (Reward) are parameters for the MiDaT model. α_p is the penalty for extracting a temporal tag that is not related to the event-date and α_r is the reward for extracting a tag that matches the date.

During learning, if the local classifier is very confident, it is possible for a word to be labeled with a tag that is not derived from the event-date, and also for a sentence-level tag to be ignored, although either case will be penalized by the agreement potentials, $\psi(m_j, t'_j)$, in the global objective. We use a local-search approach to inference that was empirically demonstrated to nearly always yield exact solutions (Ritter et al., 2013).

2.1.2 Temporal Entity Normalization with Log-Linear Model

The Temporal Normalizer is built using a log-linear model which takes the tags **t** produced by the Temporal Entity Recognizer (§2.1.1) as input and outputs one or more dates mentioned in a tweet. We formulate time normalization as a binary classification problem: given a tweet **w** published on date d^{pub}, we consider 22 candidate target dates (**w**, d_l^{cand}) such that $d_l^{cand} = d^{pub} + l$, where $l = -10, \ldots, -1, 0, +1, \ldots, +10$, limiting the possible date references that are considered within 10 days before or after the tweet creation date, in addition to $d_l^{cand} = null$ (the tweet does not mention a date). [6]

While our basic approach has the limitation, that it is only able to predict dates within ± 10 days of the target date, we found that in practice the majority of date references on social media fall within this window. Our approach is also able to score dates outside this range that are generated by traditional approaches to resolve time entities (details in §2.2.3.2).

[6] Although the temporal recognizer is trained with tweets from ± 7 days around the event date, we found that extending the candidate date range to ± 10 days for the temporal normalizer increased the performance of TweeTIME in the dev set.

The normalizer is similarly trained using the event database as distant supervision. The probability that a tweet mentions a candidate date is estimated using a log-linear model:

$$P(d^{cand}|\mathbf{w}, d^{pub}) \propto e^{\boldsymbol{\theta^n} \cdot \mathbf{g}(\mathbf{w}, d^{pub}, \mathbf{t})} \tag{2.5}$$

where $\boldsymbol{\theta^n}$ and $\mathbf{g}$ are the parameter and feature vector respectively in the Temporal Normalizer. For every tweet and candidate date pair ($\mathbf{w}$, d_l^{cand}), we extract the following set of features:

Temporal Tag Features that indicate whether the candidate date agrees with the temporal tags extracted by the Temporal Recognizer. Three cases can happen here: The recognizer can extract a tag that can not be derived from the candidate date; The recognizer can miss a tag derived from the candidate date; The recognizer can extract a tag that is derived from the candidate date.

Lexical Features that include two types of binary features from the tweet: 1) **Word Tag** features consist of conjunctions of words in the tweet and tags associated with the candidate date. We remove URLs, stop words and punctuation; 2) **Word POS** features that are the same as above, but include conjunctions of POS tags, words and temporal tags derived from the candidate date.

Time Difference Features are numerical features that indicate the distance between the creation date and the candidate date. They include difference of day ranges form -10 to 10 and the difference of week ranges from -2 to 2.

Feature Types	Example
Temporal Tag	TM^{cand}=$future \wedge future \in Tags$ DOW^{cand}=$Mon \wedge Mon \in Tags$ MOY^{cand}=$May \wedge May \in Tags$
Lexical	W="Watch" $\wedge$ TM^{cand}=$future$ W="Monday" $\wedge$ DOW^{cand}=Mon
Lexical & POS	W="Watch" $\wedge$ POS=$VB \wedge$ TM^{cand}=$future$ W="Monday" $\wedge$ POS=$NNP \wedge$ DOW^{cand}=Mon
Time Difference	DIFF-WEEK=$+1$ DIFF-DAY=$+3$

Table 2.2: Sample features extracted for the tweet shown in Figure 2.1(a) with creation date *5/6/2016*, candidate date *5/9/2016* and extracted temporal tags TL=$future$, DOW=Mon, and MOY=May.

2.2 Experiments

In the following sub-sections we present experimental results of time entity resolution from tweets using minimal supervision. We start by describing our dataset, and proceed to present our results, including a large-scale evaluation on heuristically-labeled data and an evaluation comparing against human judgements.

2.2.1 Data

We collected around 120 million tweets posted in a one year window starting from April 2011 to May 2012. These tweets were automatically annotated with named entities, POS tags and TempEx dates (Ritter et al., 2011b).

From the automatically-annotated corpus we extracted top $10,000$ events and their corresponding dates using the G^2 test, which measures the strength of association between

an entity and date using the log-likelihood ratio between a model in which the entity is conditioned on the date and a model of independence (Ritter et al., 2012).

For a given entity and date, the G^2 statistic is computed as follows:

$$G^2 = \sum_{x \in \{e, \neg e\}, y \in \{d, \neg d\}} O_{x,y} \times \ln \left(\frac{O_{x,y}}{E_{x,y}} \right) \tag{2.6}$$

Where, $O_{e,d}$ is the observed number of tweets containing a reference to event e and date d, $E_{e,d}$ is the expected number of tweets containing a both the entity and the date assuming a model of independence between the two.

Events extracted using this approach then consist of the highest-scoring entity-date pairs, for example [*Mercury, 5/9/2016*].

After automatically extracting the database of events, we next gather all tweets that mention an entity from the list that are also written within ± 7 days of the event. These tweets and the dates of the known events serve as labeled examples that are likely to mention a known date. We also include a set of pseudo-negative examples, that are unlikely to refer to any event, by gathering a random sample of tweets that do not mention any of the top $10,000$ events and where TempEx does not extract any date.

2.2.2 Large-Scale Heuristic Evaluation

We first evaluate our temporal entity recognizer model (§2.1.1), by testing how well it can predict the heuristically generated labels. As noted in previous work on distant supervision (Mintz et al., 2009a), this type of evaluation usually under-estimates precision, however it provides us with a useful intrinsic measure of performance.

In order to provide even coverage of months in the training and test set, we divide the twitter corpus into 3 subsets based on the mod-5 week of each tweet's creation date. To train system we use tweets that are created in $1st$, $2nd$ or $3rd$ weeks. To tune parameters of the MiDaT model we used tweets from $5th$ weeks, and to evaluate the performance of the trained model we used tweets from $4th$ weeks.

	P	**R**	**F_1**
MultiT	0.61	0.21	0.32
MiDaT	0.67	0.31	0.42

Table 2.3: Performance comparison of MultiT and MiDaT at predicting heuristically generated tags on the dev set.

The performance of the MiDaT model varies with the penalty and reward parameters. To find a (near) optimal setting of the values we performed a grid search on the dev set and found that a penalty of -25 and reward of 500 works best. A comparison of MultiT and MiDaT's performance at predicting heuristically generated labels is shown in Table 2.3. Precision and recall for several other values are shown in Table 2.4.

α_p	α_r	**P**	**R**	**F_1**
-100	1000	0.68	0.29	0.41
-50	1000	0.67	0.31	0.43
-10	1000	0.44	0.44	0.44
-20	1000	0.64	0.38	0.47
-25	1000	0.45	0.38	0.42
-25	500	0.64	0.38	0.48
-25	600	0.42	0.38	0.40
-25	400	0.43	0.39	0.42

Table 2.4: Performance comparison of MiDaT model with various parameters on the development set.

The word level tags predicted by the temporal recognizer are used as the input to the temporal normalizer, which predicts the referenced date from each tweet. The overall system's performance at predicting event dates on the automatically generated test set, compared against SUTime, is shown in Table 2.5.

	P	**R**	**F_1**
Dev set			
TweeTIME	0.93	0.69	0.79
SUTime	0.89	0.64	0.75
Test set			
TweeTIME	0.97	0.94	0.96
SUTime	0.85	0.75	0.80

Table 2.5: Performance comparison of TweeTIME and SUTime at predicting heuristically labeled normalized dates.

2.2.3 Evaluation Against Human Judgements

In addition to automatically evaluating our tagger on a large corpus of heuristically-labeled tweets, we also evaluate the performance of our tagging and date-resolution models on a random sample of tweets taken from a much later time period, that were manually annotated by the authors.

2.2.3.1 Evaluation of Word-Level Tags

To evaluate the performance of the MiDaT-tagger we randomly selected 50 tweets and labeled each word with its corresponding tag. Against this hand annotated test set, MiDaT achieves Precision=0.54, Recall=0.45 and F-value=0.49. A few examples of word-level tags predicted by MiDaT are shown in Table 2.6. We found that because the tags are learned as latent variables inferred by our model, they sometimes don't line up exactly with our

intuitions but still provide useful predictions, for example in Table 2.6, *Christmas* is labeled with the tag MOY=dec.

Tweets and their corresponding word tags (word^{tag})
Im^{NA} hella^{NA} excited^{future} for^{NA} tomorrow^{future}
Kick^{NA} off^{NA} the^{NA} New^{future} Year^{future} Right^{NA} @^{NA} #ClubLacura^{NA} #FRIDAY^{fri} !^{NA} HOSTED^{NA} BY^{NA} [[^{NA} DC^{NA} Young^{NA} Fly^{NA}]]^{NA}
@OxfordTownHall^{NA} Thks^{NA} for^{NA} a^{NA} top^{NA} night^{NA} at^{NA} our^{NA} Christmas^{dec} party^{NA} on^{NA} Fri!^{fri} Compliments^{NA} to^{NA} chef!^{NA} (Rose^{NA} melon^{NA} $cantaloupe^{NA}$:)^{NA}
Im^{NA} proud^{NA} to^{NA} say^{NA} that^{NA} I^{NA} breathed^{past} the^{NA} same^{NA} air^{NA} as^{NA} Harry^{NA} on^{NA} March^{mar} 21,^{21} 2015.^{NA} #KCA^{NA} #Vote1DUK^{NA}
C'mon^{present} let's^{present} jack^{NA} Tonight^{present} will^{NA} be^{present} a^{NA} night^{NA} to^{NA} remember.^{NA}

Table 2.6: Example MiDaT tagging output on the test set.

2.2.3.2 Evaluation of End-to-end Date Resolution

To evaluate the final performance of our system and compare against existing state-of-the art time resolvers, we randomly sampled 250 tweets from 2014-2016 and manually annotated them with normalized dates; note that this is a separate date range from our weakly-labeled training data which is taken from 2011-2012. We use 50 tweets as a development set and the remaining 200 as a final test set. We experimented with different feature sets on the development data. Feature ablation experiments of our normalizer model are presented in Table 2.8. The final performance of our system, compared against a range of state-of-the-art time resolvers is presented in Table 2.7. We see that TweeTIME outperforms SUTime, Tempex, HeidelTime (using its COLLOQUIAL mode, which is designed for SMS text) and UWTime. Brief descriptions of each system can be found in Section 2.3.

	P	R	F_1
Dev set			
TweeTIME	0.61	0.81	0.70
TweeTIME+SU	**0.67**	0.83	**0.74**
SUTime	0.51	**0.86**	0.64
TempEx	0.58	0.64	0.61
HeidelTime	0.57	0.63	0.60
UWTime	0.49	0.57	0.53
Test set			
TweeTIME	0.58	0.70	0.63
TweeTIME+SU	**0.62**	**0.76**	**0.68**
SUTime	0.54	0.64	0.58
TempEx	0.56	0.58	0.57
HeidelTime	0.43	0.52	0.47
UWTime	0.39	0.50	0.44

Table 2.7: Performance comparison of TweeTIME against state-of-the-art temporal taggers. TweeTIME+SU uses our proposed approach to system combination, re-scoring output from SUTime using extracted features and learned parameters from TweeTIME.

	P	R	F_1
TweeTIME	0.61	0.81	0.70
– Day Diff.	0.46	0.72	0.56
– Lexical&POS	0.48	0.80	0.60
– Week Diff.	0.49	0.85	0.62
– Lexical	0.50	0.88	0.64
– Temporal Tag	0.57	0.83	0.68

Table 2.8: Feature ablation of the Temporal Resolver by removing each individual feature group from the full set.

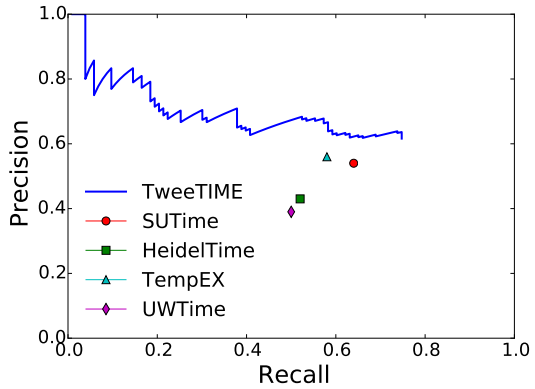


Figure 2.4: Precision and recall at resolving time entities compared against human judgements. TweeTIME achieves higher precision at comparable recall than other state-of-the-art systems.

2.2.3.3 System Combination with SUTime

As our basic TweeTIME system is designed to predict dates within ± 10 days of the creation date, it fails when a tweet refers to a date outside this range. To overcome this limitation we append the date predicted by SUTime in the list of candidate days. We then re-rank SUTime's predictions using our log-linear model, and include its output as a predicted date if the confidence of our normalizer is sufficiently high.

2.2.4 Error Analysis

We manually examined the system outputs and found 7 typical categories of errors (see examples in Table 2.9):

Spelling Variation: Twitter users are very creative in their use of spelling and abbreviations. For example, a large number of variations of the word *tomorrow* can be found in tweets, including *2morrow*, *2mrw*, *tmrw*, *2mrow* and so on. Previous temporal resolvers often fail in these cases, while TweeTIME significantly reduces such errors.

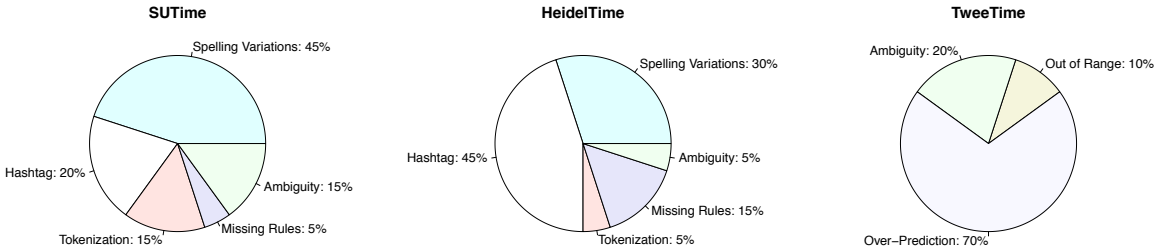


Figure 2.5: Error analyses for different temporal resolvers

Ambiguity: In many cases, temporal words like *Friday* in the tweet *"Is it Friday yet?"* may not refer to any specific event or date, but are often predicted incorrectly. Also included in this category are cases where the future and past are confused. For example, predicting the past Friday, when it is actually the coming Friday.

Missing Rule: Cases where specific temporal keywords, such as *April Fools*, are not covered by the rule-based systems.

Tokenization: Traditional systems tend to be very sensitive to incorrect tokenization and have trouble to handle entities such as *9th-december*, *May 9,2015* or *Jan1*. For the following Tweet:

> JUST IN Delhi high court asks state government to submit data on changes in pollution level since #OddEven rule came into effect on *Jan1*

TweeTIME is able to correctly extract *01/01/2016*, whereas HeidelTime, SUTime, TempEX and UWTime all failed to extract any dates.

Hashtag: Hashtags can carry temporal information, for example, #September11. Only our system that is adapted to social media can resolve these cases.

Out of Range: TweeTIME only predicts dates within 10 days before or after the tweet. Time entities referring to dates outside this range will not be predicted correctly. System combination with SUTime (Section 2.2.3.2) only partially addressed this problem.

Over-Prediction: Unlike rule-based systems, TweeTIME has a tendency to over-predict when there is no explicit time entity in the tweets, possibly because of the presence of present tense verbs. Such mistakes could also happen in some past tense verbs.

Because TweeTIME resolves time entities using a very different approach compared to traditional methods, its distribution of errors is quite distinct, as illustrated in Figure 2.5.

Error Category	Tweet	Gold Date	Predicted Date
Spelling	I cant believe *tmrw* is *fri*..the week flys by	2015-03-06	None (SUTime, HeidelTime)
Ambiguity	RT @Iyaimkatie: Is it Friday yet?????	None	2015-12-04 (TweeTime, SUTime, HeidelTime)
Missing Rule	#49ers #sanfrancisco 49ers fans should be oh so wary of *April Fools* pranks	2015-04-01	None (HeidelTime)
Tokenization	100000 - still waiting for that reply from *9th-december* lmao. you're pretty funny and chill	2015-12-09	None (SUTime, HeidelTime)
Hashtag	RT @arianatotally: Who listening to the *#SATURDAY* #Night w/ @AlexAngelo?I'm loving it.	2015-04-11	None (SUTime, HeidelTime)
Out of Range	RT @460km: In memory of Constable Christine Diotte @rcmpgrcpolice EOW: *March 12, 2002* #HeroesInLife #HerosEnVie	2002-03-12	2015-03-12 (TweeTime)
Over-Prediction	RT @tinatbh: January 2015: this will be my year December 2015: maybe not.	None	2015-12-08 (TweeTime)

Table 2.9: Representative Examples of System (SUTime, HeidelTime, TweeTIME) Errors.

2.3 Related Work

Temporal Resolvers primarily utilize either rule-based or probabilistic approaches. Notable rule-based systems such as TempEx (Mani and Wilson, 2000), SUTime (Chang and

Manning, 2012) and HeidelTime (Strötgen and Gertz, 2013) provide particularly competitive performance compared to the state-of-the-art machine learning methods. Probabilistic approaches use supervised classifiers trained on in-domain annotated data (Kolomiyets and Moens, 2010; Bethard, 2013a; Filannino et al., 2013) or hybrid with hand-engineered rules (UzZaman and Allen, 2010; Lee et al., 2014). UWTime (Lee et al., 2014) is one of the most recent and competitive systems and uses Combinatory Categorial Grammar (CCG).

Although the recent research challenge TempEval (UzZaman et al., 2013; Bethard and Savova, 2016) offers an evaluation in the clinical domain besides newswire, most participants used the provided annotated corpus to train supervised models in addition to employing hand-coded rules. Previous work on adapting temporal taggers primarily focus on scaling up to more languages. HeidelTime was extended to multilingual (Strötgen and Gertz, 2015), colloquial (SMS) and scientific texts (Strötgen and Gertz, 2012) using dictionaries and additional in-domain annotated data. One existing work used distant supervision (Angeli et al., 2012; Angeli and Uszkoreit, 2013), but for normalization only, assuming gold time mentions as input. They used an EM-style bootstrapping approach and a CKY parser. In contrast, our approach adapts temporal taggers to social media data using multiple instance learning technique, and for end-to-end task of detecting and normalizing time entities.

Distant Supervision has recently become popular in natural language processing. Much of the work has focused on the task of relation extraction (Craven and Kumlien, 1999; Bunescu and Mooney, 2007; Mintz et al., 2009b; Riedel et al., 2010; Hoffmann et al., 2011b; Nguyen and Moschitti, 2011; Surdeanu et al., 2012a; Xu et al., 2013; Ritter et al., 2013; Angeli et al., 2014). Recent work also shows exciting results on extracting named entities (Ritter et al., 2011b; Plank et al., 2014a), emotions (Purver and Battersby, 2012),

sentiment (Marchetti-Bowick and Chambers, 2012), as well as finding evidence in medical publications (Wallace et al., 2016). There are also works on noisy distant supervision for KB population and entity typing where researchers used multi-instance multi-label learning (Surdeanu et al., 2012b; Yaghoobzadeh et al., 2017; Murty et al., 2018) and custom losses (Ren et al., 2016; Abhishek et al., 2017). Distant supervision is also successfully applied on deep denoising models (Lin et al., 2016; Feng et al., 2017; Luo et al., 2017; Lei et al., 2018; Han et al., 2018). Our work is closely related to the joint word-sentence model that exploits multiple-instance learning for paraphrase identification (Xu et al., 2014) in Twitter.

2.4 Conclusion

In this chapter, we showed how to learn time entities from large amounts of unlabeled text, using a database of known events as distant supervision. We presented a method for learning a word-level temporal tags, using a minimally supervised model, from tweets that are heuristically labeled with only sentence-level labels. This approach was further extended to account for the case of missing tags, or temporal properties that are not explicitly mentioned in the text of a tweet. These temporal tags were then combined with a variety of other features in a novel date-resolver that predicts normalized dates referenced in a Tweet. By learning from large quantities of in-domain data, we were able to achieve 0.68 F1 score on the end-to-end time normalization task for social media data, significantly outperforming SUTime, TempEx, HeidelTime and UWTime on this challenging dataset for time normalization.

Chapter 3: Code and Named Entity Recognition in StackOverflow

Recently there has been significant interest in modeling human language together with computer code (Quirk et al., 2015; Iyer et al., 2016; Yin and Neubig, 2018), as more data becomes available on websites such as StackOverflow and GitHub. Software domain-specific NLP tools could help these research aspects. Such NLP tools would also facilitate a wide range of downstream applications including automated software knowledge bases extraction (Movshovitz-Attias and Cohen, 2015), automated quality measurements of software social network posts (Ravi et al., 2014), finding similar questions and more. However, there is a lack of NLP resources and techniques for identifying the software-related named entities within natural language texts.

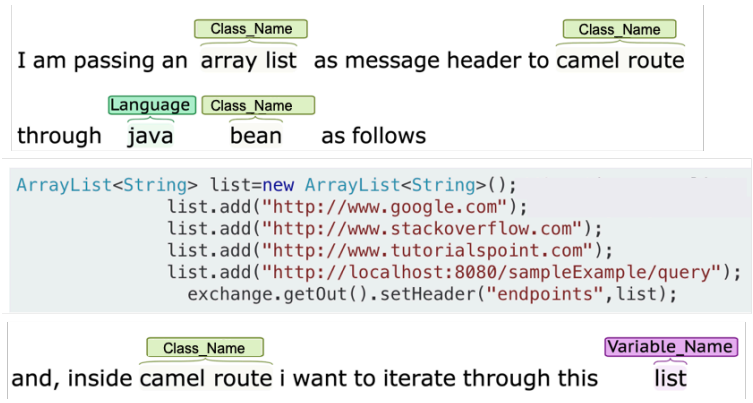


Figure 3.1: Examples of software-related named entities in a StackOverflow post.

In this chapter, we present a comprehensive study that investigates the unique challenges of named entity recognition in the social computer programming domain. These named entities are often ambiguous and have implicit reliance on the accompanied code snippets. For example, the word '*list*' commonly refers to a data structure, but can also be used as a variable name (as shown in post of Figure 3.1). In order to recognize these entities, we have introduced a software-related named entity recognizer, named SoftNER (Tabassum et al., 2020b) that utilizes an attention network to combine the local sentence-level context with corpus-level information extracted from the code snippets. Using our newly annotated corpus of 15,372 sentences in StackOverflow, we rigorously test our proposed SoftNER model, which outperforms BiLSTM-CRF model and fine-tuned BERT model for identifying 20 types of software-related named entities.

3.1 Annotated Software NER Corpus

In this section, we describe the construction of our StackOverflow NER corpus. We randomly selected 1,237 question-answer threads from StackOverflow 10-year archive (from September 2008 to March 2018) and manually annotated them with 20 types of entities. For each question, four answers were annotated, including the accepted answer, the most upvoted answer, as well as two randomly selected answers (if they exist). Table 3.1 shows the statistics of our corpus. 40% of the question-answer threads were double-annotated, which are used as the development and test sets in our experiments (§3.3). We also annotated 6,501 sentences from GitHub readme files and issue reports as additional evaluation data.

	Train	Dev	Test	Total
#questions	741	247	249	1,237
#answers	897	289	315	1,501
#sentences	9,315	2,942	3,115	15,372
#tokens	136,996	43,296	45,541	225,833
#entities	11,440	3,949	3,733	19,122

	per Question	per Answer
avg. #sentences	6.84	4.60
avg. #tokens	98.46	69.37
avg. #entities	7.62	5.11
avg. #tokens per sent.	14.38	15.08

Table 3.1: Statistics of our StackOverflow NER corpus. These counts exclude all the code blocks and output blocks (i.e., lines that appear within ⟨code⟩ and ⟨blockquote⟩ tags).

3.1.1 Annotation Schema

We defined and annotated 20 types of fine-grained entities, including 8 code-related entities and 12 natural language entities. The code entities include mentions of CLASS, VARIABLE, IN LINE CODE, FUNCTION, LIBRARY, VALUE, DATA TYPE, and HTML XML TAG. Whereas the natural language entities include mentions of APPLICATION, UI ELEMENT, LANGUAGE, DATA STRUCTURE, ALGORITHM, FILE TYPE, FILE NAME, VERSION, DEVICE, OS, WEBSITE, and USER NAME.

Our annotation guideline was developed through several pilots and further updated with notes to resolve difficult cases as the annotation progressed.[7] Each entity type was defined to encourage maximum span length (e.g., '*SGML parser*' instead of '*SGML*'). We annotated noun phrases without including modifiers (e.g., '*C*' instead of '*Plain C*'), except a few special cases (e.g., '*rich text*' as a common FILE TYPE). Table 3.2 represents the detail statics for each entity types with representative example phrases. On average, an

[7]Our annotation guideline is available at:
https://github.com/jeniyat/StackOverflowNER/.

entity contains about 1.5 tokens. While VARIABLE, FUNCTION and CLASS names mostly consist of only a single token, our annotators found that some are written as multiple tokens when mentioned in natural language text (e.g., '*array list*' for '*ArrayList*' in Figure 3.1). The annotators were asked to read relevant code blocks or software repositories to make a decision, if needed. Annotators also searched Google or Wikipedia to categorize unfamiliar cases.

Entity	Example	Count	avg. #word	avg. #char
CLASS	*ItemTemplate, PersistentGenericSet, actionManager*	2202	1.09 ± 0.32	10.08 ± 5.43
APPLICATION	*Visual Studio, Weka, Homebrew, FFmpeg, Sonar Scanner*	2019	1.23 ± 0.32	10.08 ± 5.43
VARIABLE	*key, value, pt, A*	1607	1.07 ± 0.28	7.58 ± 5.79
UI ELEMENT	*textbox, wizard, button*	1536	1.13 ± 0.38	6.44 ± 2.82
INLINE CODE	*-type d, alias.⟨alias⟩, adoqry.active := true*	1358	2.65 ± 2.74	15.40 ± 18.26
FUNCTION	*.toLowerCase(), GmailApp.search(), search*	1281	1.12 ± 0.62	11.07 ± 8.09
LANGUAGE	*python, java, c++*	1060	1.02 ± 0.15	4.55 ± 2.65
LIBRARY	*Pyspark, aws-java-sdk-core, OAuth*	1186	1.17 ± 0.48	7.10 ± 3.62
VALUE	*"hello world", '255.255.255.0', 30.5, True*	1188	1.64 ± 1.39	5.6 ± 6.03
DATA STRUCTURE	*tree, stack, linkedlist*	987	1.02 ± 0.14	5.26 ± 1.66
DATA TYPE	*string, integer, boolean, pointer*	634	1.05 ± 0.26	6.44 ± 1.97
FILE TYPE	*exe, dll, rich text*	556	1.01 ± 0.09	3.95 ± 1.66
FILE PATH	*Test.pdb, facebook_consumer.js, config/modules.config.php*	600	1.16 ± 0.67	13.35 ± 9.08
VERSION	*2003 R2, 4.2.6-200.fc22.x86_64, XP*	500	1.08 ± 0.39	3.76 ± 2.81
HTML XML TAG	*⟨div⟩, span, br*	277	1.07 ± 0.27	5.68 ± 3.81
DEVICE	*iPhone, Arduino MCU, 2d barcode scanner, nios cpu*	361	1.19 ± 0.46	6.09 ± 3.25
OPERATING SYSTEM	*Linux, iOS, Android*	303	1.10 ± 0.35	5.87 ± 2.14
WEBSITE	*stackexchange, GitHub, Codecourse, railstutorial.org*	182	1.10 ± 0.37	7.65 ± 3.60
USER NAME	*Chris M, @KamranFarzami, M.Deinum*	139	1.29 ± 0.53	7.63 ± 3.58
ALGORITHM	*A*-search, bubblesort, divide-and-conquer*	60	1.48 ± 0.59	8.45 ± 5.34

Table 3.2: Software-specific entity categories in the StackOverflow annotated corpus.

The annotators were asked to update, correct, or add annotations from the user provided ⟨code⟩ markdown tags. StackOverflow users can utilize ⟨code⟩ markdowns to highlight the code entities within the natural language sentences. However, in reality, many users do not enclose the code snippets within the ⟨code⟩ tags; and sometimes use them to highlight non-code elements, such as email addresses, user names, or natural language words. While creating the StackOverflow NER corpurs, we found that 59.73% of code-related entities are not marked by the StackOverflow users. Moreover, only 75.54% of the ⟨code⟩ enclosed

texts are actually code-related, while 10.12% used to are highlighting natural language texts. The rest of cases are referring to non-code entities, such as SOFTWARE NAMES and VERSIONS. While markdown tag could be a useful feature for entity segmentation (§3.2.1.3), we emphasize the importance of having a human annotated corpus for training and evaluating NLP tools in the software domain.

3.1.2 Annotation Agreement

Our corpus was annotated by four annotators who are college students majored in computer science. We used a web-based annotation tool, BRAT (Stenetorp et al., 2012a), and provided annotators with links to the original post on StackOverflow. Figure 3.2 shows a snippet of our annotation interface. For every iteration, each annotator was given 50 question-answer threads to annotate, 20 of which were double-annotated. An adjudicator then discussed disagreements with annotators, who also cross-checked the 30 single-annotated questions in each batch. The inter-annotator agreement is 0.62 before adjudication, measured by token-level Cohen's Kappa (Cohen, 1960).

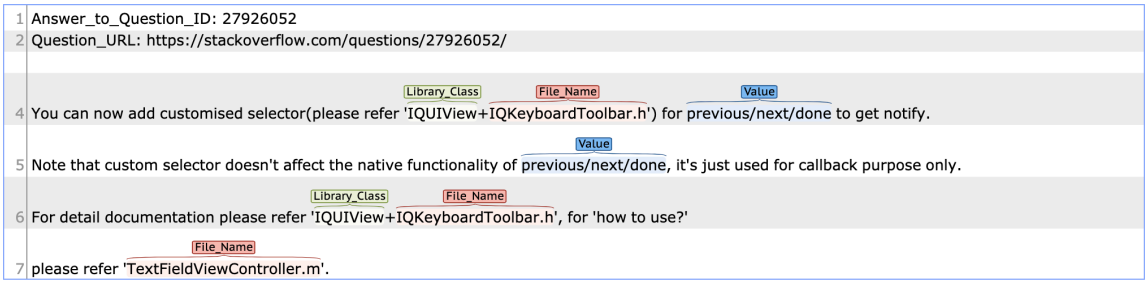


Figure 3.2: Our annotation interface.

3.1.3 Additional GitHub Data

To better understand the domain adaptability of our work, we further annotated the readme files and issue reports from 143 randomly sampled repositories in the GitHub dump (Gousios and Spinellis, 2012) from October 29, 2007 to December 31, 2017. We removed all the code blocks from the issue reports and readme files collected from these 143 repositories. The resulting GitHub NER dataset consists of 6,510 sentences and 10,963 entities of 20 types labeled by two in-house annotators. The inter-annotator agreement of this dataset is 0.68, measured by token-level Cohen's Kappa.

3.1.4 StackOverflow/GitHub Tokenization

We designed a new tokenizer, SOTOKENIZER, specifically for the social computer programming domain. StackOverflow and GitHub posts exhibit common features of web texts, including abbreviations, emoticons, URLs, ungrammatical sentences and spelling errors.

We found that tokenization is non-trivial as many code-related tokens are mistakenly split by the existing web-text tokenizers, including the CMU Twokenizer (Gimpel et al., 2011), Stanford TweetTokenizer (Manning et al., 2014), and NLTK Twitter Tokenizer (Bird et al., 2009):

txScope.Complete()	[*'txScope' '.' 'Complete' '(' ')'*]
std::condition_variable	[*'std' ':' ':' 'condition_variable'*]
math.h	[*'math' '.' 'h'*]
⟨*span*⟩	[*'⟨' 'span' '⟩'*]
a==b	[*'a' '=' '=' 'b'*]

Therefore, we implemented a new tokenizer, using Twokenizer[8] as the starting point and added additional regular expression rules to avoid splitting code-related tokens.

3.2 SoftNER: Named Entity Recognition Model for social software domain

The extraction of software-related named entities imposes significant challenges as it requires resolving a significant amount of unseen tokens, inherent polysemy, and salient reliance on context. Unlike news or biomedical data, spelling patterns and long-distance dependencies are more crucial in the software domain to resolve ambiguities and categorize unseen words. Taken in isolation, many tokens are highly ambiguous and can refer to either programming concepts or common English words, such as: '*go*', '*react*', '*spring*', '*while*', '*if*', '*select*'. To address these challenges, we design the SoftNER model that leverages sentential context to disambiguate and domain-specific character representations to handle rare words. Figure 3.3 shows the architecture of our model, which consists of primarily three components:

- An **input embedding layer** (§3.2.1) that extracts contextualized embeddings from the $\text{BERT}_{\text{base}}$ model and two new domain-specific embeddings for each word in the input sentence.
- A **embedding attention layer** (§3.2.2) that combines the three word embeddings using an attention network.
- A **linear-CRF** layer that predicts the entity type of each word using the attentive word representations from the previous layer.

[8] https://github.com/myleott/ark-twokenize-py

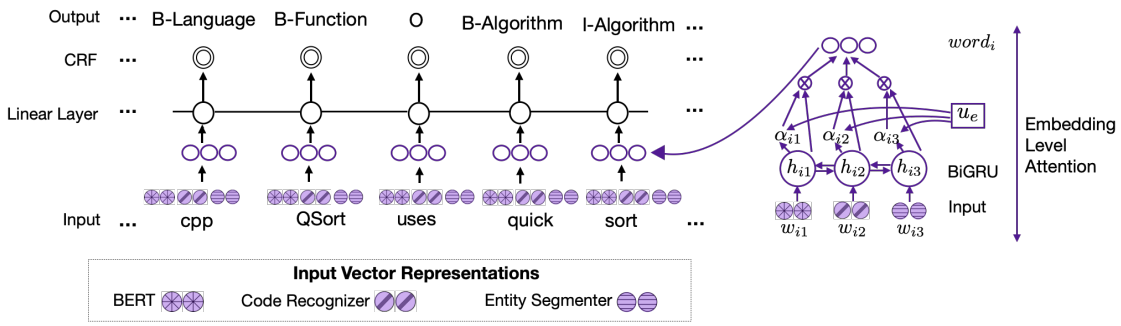


Figure 3.3: Our SoftNER model. It utilizes an attention network to combine the contextual word embeddings ($\text{BERT}_{\text{base}}$) with the domain-specific embeddings (Code Recognizer and Entity Segmenter). The detailed structure of the attention network is depicted on the right.

3.2.1 Input Embeddings

For each word in the input sentence, we extract in-domain BERT (Devlin et al., 2019) representations and two new domain-specific embeddings produced by (i) a **Code Recognizer**, which represents if a word can be part of a code entity regardless of context; and (ii) an **Entity Segmenter**, that predicts whether a word is part of any named entity in the given sentence. Each domain-specific embedding is created by passing a binary value, predicted by a network independent from the SoftNER model. We describe the two standalone auxiliary models that generate these domain-based vectors below.

3.2.1.1 In-domain Word Embeddings

Texts in the software engineering domain contain programming language tokens, such as variable names or code segments, interspersed with natural language words. This makes input representations pre-trained on general book or Wikipedia texts unsuitable for software domain. We pre-trained different in-domain word embeddings, including

BERT (BERTOverflow), ELMo (ELMoVerflow), and GloVe (GloVerflow) vectors on the StackOverflow 10-year archive[9] of 152 million sentences and 2.3 billion tokens (§3.2.3).

3.2.1.2 Context-independent Code Recognition

Humans with prior programming knowledge can easily recognize that '*list()*' is code, '*list*' can be either code or a common English word, whereas '*listing*' is more likely a non-code natural language token. We thus introduce a code recognition module to capture such prior probability of how likely a word can be a code token without considering any contextual information. It is worth noting that this standalone code recognition model is also useful for language-and-code research, such as retrieving code snippets based on natural language queries (Iyer et al., 2016; Giorgi and Bader, 2018; Yao et al., 2019).

Our code recognition model (**Code Recognizer**) is a binary classifier. It utilizes language model features and spelling patterns to predict whether a word is a code entity. The input features include unigram word and 6-gram character probabilities from two language models (LMs) that are trained on the Gigaword corpus (Napoles et al., 2012) and all the code-snippets in the StackOverflow 10-year archive respectively. We also pre-trained FastText (Joulin et al., 2016) word embeddings using these code-snippets, where a word vector is represented as a sum of its character ngrams. We first transform each ngram probability into a k-dimensional vector using Gaussian binning (Maddela and Xu, 2018), which has shown to improve the performance of neural models using numeric features (Sil et al., 2017; Liu et al., 2016; Maddela and Xu, 2018). We then feed the vectorized features into a linear layer, concatenate the output with FastText character-level embeddings, and pass them through another hidden layer with sigmoid activation. We predict the token as a code-entity if the output probability

[9] https://archive.org/details/stackexchange

is greater than 0.5. This binary prediction is then converted into a vector and used as an input to the SoftNER model.

3.2.1.3 Entity Segmentation

The segmentation task refers to identifying entity spans without assigning entity category. Entity segmentation is simpler and less error-prone than entity recognition as it does not require a fine-grained classification of the entity types. In fact, a segmentation model (**Entity Segmenter**) trained on our annotated StackOverflow corpus can achieve 90.41% precision on the dev set (details in §3.3.4), predicting whether each token is a part of entity in the given sentence. Our segmentation model fine-tunes the in-domain BERT after concatenating it with two hand-crafted features:

- **Word Frequency** represents the word occurrence count in the training set. As many code tokens are defined by individual users, they occur much less frequently than normal English words. In fact, code and non-code tokens have an average frequency of 1.47 and 7.41 respectively in our corpus. Moreover, ambiguous token that can be either code or non-code entities, such as '*windows*', have a much higher average frequency of 92.57. To leverage this observation, we include word frequency as a feature, converting the scalar value into a k-dimensional vector by Gaussian binning (Maddela and Xu, 2018).

- **Code Markdown** indicates whether the given token appears inside a ⟨code⟩ markdown tag in the StackOverflow post. It is worth noting that ⟨code⟩ tags are noisy as users do not always enclose inline code in a ⟨code⟩ tag or sometimes use the tag to highlight non-code texts (details in §3.1.1). Nevertheless, we find it helpful to

include the markdown information as a feature as it improves the performance of our segmentation model.

This inclusion of these hand-crafted features is influenced by Wu et al. (2018), where word-shapes and POS tags were shown to improve the performance of sequence tagging models.

3.2.2 Embedding-Level Attention

For each input word w_i in the input sentence, we have three embeddings: BERT (w_{i1}), Code Recognizer (w_{i2}), and Entity Segmenter (w_{i3}). We introduce the embedding-level attention α_{it} ($t \in \{1, 2, 3\}$), which captures each embedding's contribution towards the meaning of the word, to combine them together. To compute α_{it}, we pass the input embeddings through a bidirectional GRU and generate their corresponding hidden representations $h_{it} = \overleftrightarrow{GRU}(w_{it})$. These vectors are then passed through a non-linear layer, which outputs $u_{it} = tanh(W_e h_{it} + b_e)$. We introduce an embedding-level context vector u_e, which is randomly initialized and updated during the training process. This context vector is combined with the hidden embedding representation using a softmax function to extract weight of the embeddings: $\alpha_{it} = \frac{exp({u_{it}}^T u_e)}{\sum_t exp({u_{it}}^T u_e)}$. Finally, we create the word vector by a weighted sum of all the information from different embeddings as $word_i = \sum_t \alpha_{it} h_{it}$. The aggregated word vector $word_i$ is then fed into a linear-CRF layer, which predicts the entity category for each word based the BIO tagging schema.

3.2.3 Implementation Details

We use PyTorch framework to implement our proposed SoftNER model and its two auxiliary components, namely code recognition and entity segmentation. The input to the SoftNER model include 850-dimensional vectors extracted from both the code recognizer and the entity segmenter.

We pre-trained $\text{BERT}_{\text{base}}$, ELMo and GloVe vectors on 152 million sentences from the StackOverflow, excluding sentences from the 1,237 posts in our annotated corpus. The pre-training of the 768-dimensional $\text{BERT}_{\text{base}}$ model with 64,000 WordPiece vocabulary took 7 days on a Google TPU. The pre-training of 1024-dimensional ELMo vectors took 46 days on 3 NVIDIA Titan X Pascal GPUs. The pre-training of 300-dimensional GloVe embeddings (Pennington et al., 2014) with a frequency cut-off of 5 took 8 hours on a server with 32 CPU cores and 386 GB memory.

We train the SoftNER model and the two auxiliary models separately. Our segmentation model follows the simple BERT fine-tuning architecture except for the input, where BERT embeddings are concatenated with 100-dimensional code markdown and 10-dimensional word frequency features. We set the number of bins k to 10 for Gaussian vectorization. Our code recognition model is a feedforward network with two hidden layers and a single output node with sigmoid activation.

3.3 Experiments

In this section, we show that our SoftNER model outperforms all the previous NER approaches on the StackOverflow and GitHub data. We also discuss the factors pivotal to

	P	R	F_1
Test set			
Feature-based CRF	71.77	39.70	51.12
BiLSTM-CRF (ELMoVerflow)	73.03	64.82	68.68
Attentive BiLSTM-CRF (ELMoVerflow)	78.22	78.59	78.41
Fine-tuned BERT	77.02	45.92	57.54
Fine-tuned BERTOverflow	68.77	67.47	68.12
SoftNER (BERTOverflow)	**78.42**	**79.79**	**79.10**
Dev set			
Feature-based CRF	66.85	46.19	54.64
BiLSTM-CRF (ELMoVerflow)	74.44	68.71	71.46
Attentive BiLSTM-CRF (ELMoVerflow)	79.43	80.00	79.72
Fine-tuned BERT	79.57	46.42	58.64
Fine-tuned BERTOverflow	72.11	70.51	71.30
SoftNER (BERTOverflow)	**78.81**	**81.72**	**80.24**

Table 3.3: Evaluation on the *dev* and *test* sets of the StackOverflow NER corpus. Our SoftNER model outperforms the existing approaches.

the performance of our model, namely pre-trained in-domain BERT embeddings and two domain-specific auxiliary tasks.

3.3.1 Data

We train and evaluate our SoftNER model on the StackOverflow NER corpus of 9,352 train, 2,942 development and 3,115 test sentences we constructed in §3.1. We use the same data for our segmentation model but replace all the entity tags with an I-ENTITY tag. For the code recognition model, we created a new lexicon of 6,000 unique tokens randomly sampled from the training set of the StackOverflow NER corpus. Each token was labelled independently without context as CODE, AMBIGUOUS or NON-CODE by two annotators with computer science background. The inter-annotator agreement was 0.89, measured by Cohen's Kappa. After discarding disagreements, we divided the remaining 5,312 tokens into 4,312 train and 1,000 test instances. Then, we merged AMBIGUOUS and NON-CODE

categories to facilitate binary classification. We name this dataset of 5312 individual tokens as SOLEXICON.

3.3.2 Evaluation Against the Baseline Models

Table 3.3 shows the precision (**P**), recall (**R**) and $\mathbf{F}_1$ score comparison of different models evaluated on the StackOverflow NER corpus. Our SoftNER model outperforms the existing NER approaches in all the three metrics. In the following subsection we describe the architecture of these baseline models.

3.3.2.1 Feature-based Linear CRF

We implemented a CRF baseline model using CRFsuite[10] to extract the software entities.

	P	**R**	$\mathbf{F}_1$
Feature-based CRF	66.85	46.19	54.64
– Context Features	68.91	43.58	53.39
– Markdown Feature	70.64	40.15	51.20
– Rule and Gazetteer Features	69.71	40.66	51.36

Table 3.4: Feature based CRF performance with varying input features on *dev* data.

This model uses standard orthographic, contextual and gazetteer features. It also includes the code markdown tags (§3.2.1.3) and a set of regular expression features. The regular expressions are developed to recognize specific categories of code-related entities. Feature ablation experiments on this CRF model are presented in Table 3.4. One noticeable distinction from the named entity recognizer in many other domains is that the *contextual features are not as helpful in feature-based CRFs for classifying software entities*. This is

[10]http://www.chokkan.org/software/crfsuite/

because, in the StackOverflow NER corpus a significant number of neighbouring words are shared among different software entities. As an example, the bigram *'in the'* frequently appears as the left context of the following types: APPLICATION, CLASS, FUNCTION, FILE TYPE, UI ELEMENT, LIBRARY, DATA STRUCTURE and LANGUAGE. We found non-contextual features that consider the spelling features of tokens within entities are extremely important, as the entities in StackOverflow are either strings that follow certain rules (e.g. FILE NAME) or computer terminologies (e.g. PROGRAMMING LANGUAGE) or proper nouns (e.g. APPLICATION).

3.3.2.2 BiLSTM-CRF with ELMoVerflow

We implemented a baseline model with in-domain ELMo embeddings (**ELMoVerflow**; details in §3.2.3). This architecture is used as the state-of-the-art baseline named-entity recognition models in various domains (Lample et al., 2016; Kulkarni et al., 2018; Dai et al., 2019).

3.3.2.3 Attentive BiLSTM CRF with ELMoVerflow

We propose a baseline Attentive NER model that utilizes a BiLSTM-CRF network to predict the entity type of each word from its weighted representations. The weighted word representations are extracted by a multi-level attention network, similar to Yang et al.(2016), that combines the contextualized ELMo embeddings with the code recognizer (§3.2.1.2) and segmenter vector (§3.3.2.3.1). These three input embeddings are merged together in the first attention layer and then their corresponding weights are calculated using the second layer. Although such multi-level attention is not commonly used in NER, we found it empirically helpful for the software domain (see Table 3.5).

	P	R	F_1
Attentive BiLSTM-CRF	**79.43**	**80.00**	**79.72**
– Multi-level Attention	77.68	78.08	77.88
– Code Recognizer	77.18	77.76	77.47
– Entity Segmenter	74.82	75.32	75.07

Table 3.5: Ablation study of Attentive-NER on the *dev* set of StackOverflow NER corpus.

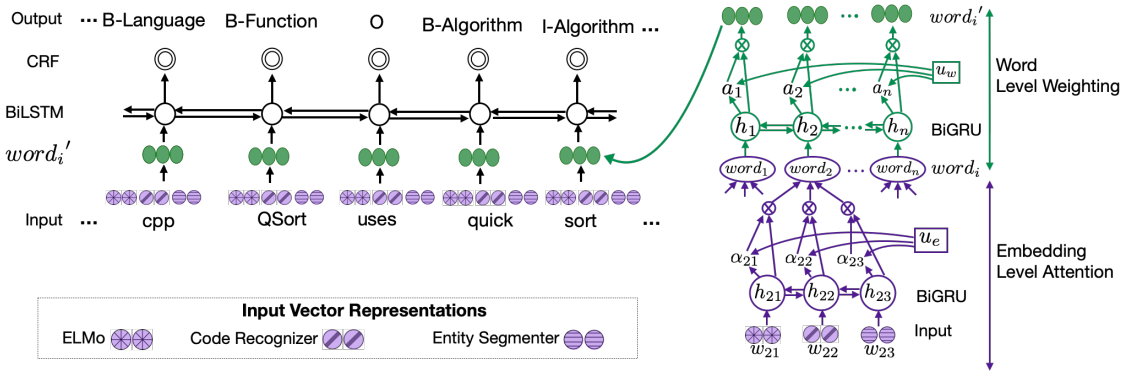


Figure 3.4: Our Attentive BiLSTM CRF model. It utilizes an attention network to combine the contextual word embeddings (ELMo) with the domain-specific embeddings (Code Recognizer and Entity Segmenter). The detailed structure of the attention network is depicted on the right.

Embedding-Level Attention uses three embeddings, ELMo (w_{i1}), Code Recognizer (w_{i2}), and Entity Segmenter (w_{i3}), for each word w_i in the input sentence. The embedding-level attention α_{it} ($t \in \{1, 2, 3\}$) to captures each embedding's contribution towards the meaning of the word. To compute α_{it}, it pass the input embeddings through a bidirectional GRU and generate their corresponding hidden representations $h_{it} = \overleftrightarrow{GRU}(w_{it})$. These vectors are then passed through a non-linear layer, which outputs $u_{it} = tanh(W_e h_{it} + b_e)$. It uses an embedding-level context vector, u_e, which is learned during the training process. This context vector is combined with the hidden embedding representation using a softmax function to extract weight of the embeddings, $\alpha_{it} = \frac{exp({u_{it}}^T u_e)}{\sum_t exp({u_{it}}^T u_e)}$. Finally, the word

vector is created by a weighted sum of all the information from different embeddings as $word_i = \sum_t \alpha_{it} h_{it}$.

Weighted Word Representation uses a word-level weighting factor α_i to emphasize the importance of each word w_i for the NER task. Similar to the embedding-level attention, it calculates α_i from the weighted word vectors $word_i$. A bidirectional GRU is used to encode the summarized information from neighbouring words and thus it get $h_i = \overleftrightarrow{GRU}(word_i)$. This is then passed through a hidden layer which outputs $u_i = tanh(W_w h_i + b_w)$. Then the normalized weight for each word vector is extracted by $\alpha_i = \frac{exp({u_i}^T u_w)}{\sum_t exp({u_i}^T u_w)}$, where u_w is another word-level context vector that is learned during training. The final weighted word representation is computed by $word'_i = \alpha_i h_i$.

Subsequently, the aggregated word vector $word'_i$ is fed into a BiLSTM-CRF network, which predicts the entity category for each word. The complete architecture of the Attentive BiLSTM CRF model is illustrated in Figure 3.4. Compared to BiLSTM-CRF, our proposed Attentive BiLSTM-CRF demonstrates a 9.7 increase in F_1 on the *test* set (Table 3.3) and reduces the segmentation errors and entity type errors by 80.33% 23.34% respectively.

3.3.2.3.1 Entity Segmentation with ELMoVerflow

The Attentive-NER tagger utilizes the outputs from an auxiliary segmentation module which consists of a BiLSTM encoder and a CRF decoder. This model concatenates ELMo embeddings with two hand-crafted features- word frequency and code markdown (§3.2.1.3).

The segmentation model follows the same architecture and training setup as the Attentive-NER model except for the input, where ELMo embeddings are concatenated with 100-dimensional code markdown and 10-dimensional word frequency features. The binary

output from this entity segmenter model is later passed as through an embedding layer and used as one of the auxiliary inputs of the Attentive NER model.

Table 3.6 shows the performance of this segmentation model with ELMoVerflow on the *dev* set. This model achieves an F_1 score of 84.3 and an accuracy of 97.4%. The ablation study in Table 3.6 depicts the importance of the hand-crafted frequency and markdown features for this segmenter model by providing an increment of 1.2 and 2.1 points in the F_1 score respectively.

	P	**R**	**F_1**
Entity Segmentation (ELMoVerflow)	**86.80**	**81.86**	**84.26**
– Word Frequency	84.61	81.53	83.04
– Code Markdown	82.49	81.83	82.16

Table 3.6: Ablation study of our segmentation model with ELMoVerflow on the *dev* set of the StackOverflow NER corpus.

3.3.2.4 Fine-tuned out-of-domain BERT

We fine-tune the original $BERT_{base}$ cased checkpoint[11] on our annotated corpus to extract the software entities.

3.3.2.5 Fine-tuned in-domain BERT

We pre-trained $BERT_{base}$ (**BERTOverflow**; details in §3.2.3) model on StackOverflow archive and then fine-tune it on our annotated corpus.

[11] https://github.com/google-research/BERT

3.3.3 In-domain vs. out-of-domain Word Embeddings

Table 3.7 shows the performance comparison between in-domain and out-of-domain word embeddings. We consider off-the-shelf BERT (Devlin et al., 2019), ELMo (Peters et al., 2018) and GloVe (Pennington et al., 2014) vectors trained on newswire and web texts as out-of-domain embeddings. When using the BiLSTM-CRF model (Lample et al., 2016; Kulkarni et al., 2018; Dai et al., 2019), we observe a large increase of 13.64 F_1 score when employing in-domain ELMo (ELMoVerflow) representations over in-domain GloVe (GloVeOverflow), and an increase of 15.71 F_1 score over out-of-domain ELMo. We found that fine-tuning out-of-domain BERT (Devlin et al., 2019) outperforms the out-of-domain ELMo (Table 3.7), although it underperforms in-domain ELMo (ELMoVerflow) by 12.92 F_1 score and in-domain BERt (BERTOverflow) by 12.76 F_1 score (Table 3.3). Similarly, in-domain ELMo outperforms the out-of-domain fine-tuned BERT by 10.67 F_1 score on Github data (Table 3.12; more details in §3.4).

It is worth noting that, the performance improvements from contextual word embeddings are more pronounced on our software domain than on newswire and biomedical domains. Original ELMo and BERT outperform GloVe by 2.06 and 2.12 points in F_1 respectively on CoNLL 2003 NER task of newswire data (Peters et al., 2018; Devlin et al., 2019). For biomedical domain, in-domain ELMo outperforms out-of-domain ELMo by only 1.33 points in F_1 on the BC2GM dataset (Sheikhshabbafghi et al., 2018).

We hypothesized that the performance gains from the in-domain contextual embeddings are largely aided by the model's ability to handle ambiguous and unseen tokens. The increase in performance is especially notable (41% $\rightarrow$ 70% accuracy) for unseen tokens, which constitute 38% of the tokens inside gold entity spans in our dataset. This experiment also

	P	R	F_1
out-of-domain Word Embeddings			
GloVe (newswire+Wiki+Web)	61.71	49.08	54.67
ELMo (newswire+Wiki)	67.66	47.41	55.75
Fine-tuned BERT (book+Wiki)	45.92	77.02	57.54
In-Domain Word Embeddings			
GloVeOverflow	66.28	51.28	57.82
ELMoVerflow	**74.44**	68.71	**71.46**
Fine-tuned BERTOverflow	72.11	**70.51**	71.30

Table 3.7: Performance of fine-tuned BERT model, BiLSTM-CRF model with GloVe and ELMo embeddings on the *dev* set of our StackOverflow NER corpus. Contextualized word representations show a clear benefit when trained on the in-domain StackOverflow data.

demonstrates that our annotated NER corpus provides an attractive test-bed for measuring the adaptability of different contextual word representations.

3.3.4 Evaluation of Auxiliary Systems

The domain-specific vectors produced by the **Code Recognizer** and the **Entity Segmenter** are also crucial for the overall performance of our SoftNER model. Table 3.8 shows an ablation study. Removing code recognizer vectors and entity segmenter vectors results in a drop of 2.19 and 3.69 in F_1 scores respectively. If we replace embedding-level attention with a simple concatenation of embeddings, the performance also drop by 2.81 F_1. In addition, we evaluate the effectiveness of our two domain-specific auxiliary systems on their respective tasks.

	P	R	F_1
SoftNER	**78.81**	**81.72**	**80.24**
– Embedding Attention	75.83	79.09	77.43
– Code Recognizer	78.76	77.35	78.05
– Entity Segmenter	77.82	75.32	76.55

Table 3.8: Ablation study of SoftNER on the *dev* set of StackOverflow NER corpus.

3.3.4.0.1 Code Recognition:

Table 3.9 compares the performance of our code recognition model with other baselines on the SLEXICON *test* set (§3.3.1), which consists of 1,000 random words from the *train* set of StackOverflow NER corpus classified as either a code or a non-code token. The baselines include: (i) a Most Frequent Label baseline, which assigns the most frequent label according to the human annotation in SOLEXICON *train* set; and (ii) a frequency baseline, which learns a threshold over token frequency in the *train* set of StackOverflow NER corpus using a decision tree classifier. Our model outperforms both baselines in terms of F_1 score. Although the most frequent label baseline achieves better precision than our model, it performs poorly on unseen tokens resulting in a large drop in recall and F_1 score. The ablation experiments show that the FastText word embeddings along with the character and word-level features are crucial for the code recognition model.

3.3.4.0.2 Entity Segmentation:

Table 3.10 shows the performance of our segmentation model on the *dev* set of our StackOverflow corpus, where the entity tags are replaced by an I-ENTITY tag. Our model achieves an F_1 score of 88.09 and with 90.41% precision and 85.89% recall. Incorporating word

	P	R	F_1
Token Frequency	33.33	2.25	4.22
Most Frequent Label	**82.21**	58.59	68.42
Our Code Recognition Model	**78.43**	83.33	**80.80**
– Character ngram LMs	64.13	**84.51**	72.90
– Word ngram LMs	67.98	72.96	70.38
– FastText Embeddings	76.12	81.69	78.81

Table 3.9: Evaluation results and feature ablation of our code recognition model on SOLEXICON *test* set of 1000 manually labeled unique tokens, which are sampled from the *train* set of StackOverflow NER corpus.

frequency and code markdown feature increases the F_1 score by 1.57 and 2.66 points respectively. The low 10.5 F_1 score of Stanford NER tagger (Manning et al., 2014), which is trained on newswire text, demonstrates the importance of domain-specific tools for the software engineering domain.

	P	R	F_1
Stanford NER Tagger	63.02	5.74	10.52
Our Entity Segmentation Model	**90.41**	**85.89**	**88.09**
– Word Frequency	88.32	84.79	86.52
– Code Markdown	86.23	84.64	85.43

Table 3.10: Evaluation of our segmentation model on the *dev* set of the StackOverflow NER corpus.

3.3.5 Web Demo

We have developed a web based API, which facilitates running our complete pipeline in a straight forward manner as shown in Figure 3.5. The representative illustration shows an example in python, but in practice, the SoftNER is capable of extraction of named entities from any software-domain texts independent of the programming-language .

Figure 3.5: Web API for extracting the software related named entities extraction, which is accessible at http://sdl898-ritter2.cse.ohio-state.edu:8923/

The average run time for this web demo is 22.91 seconds (averaged over 25 different runs). Our NER module uses 3 different input representations (§3.2.1) and we found that majority of the time is consumed by the extraction of contextual word representations with an average time of 15.78 seconds. Extraction of the predictions from the code recognizer takes an average time of 0.09 seconds. Combining the inputs with the embedding level attention takes an average time of 4.18 seconds, whereas the CRF tag extraction layer takes an average time of 1.03 seconds. The rest of the time is spent to process the input texts and then convert their outputs into brat format which facilitates the web rendering. We think the overall run time of this demo can be reduced if we could utilize a faster contextualized

embedding extraction system, probably by using multiple GPUs. We leave the run time reduction of this web demo as our future work.

3.3.6 Error Analysis

Based on our manual inspection, the incorrect predictions made by NER systems on StackOverflow data can be largely classified into the following two categories (see examples in Table 3.11):

- **Segmentation Mismatch** refers to the cases where model predicts the boundary of entities incorrectly. Our SoftNER model reduces such segmentation errors by 89.36% compared to the fine-tuned BERTOverflow baseline.

Segmentation Mismatch	**Gold:** installing on Windows Server 2012 R2. (Operating_System, Version) **Predicted:** installing on Windows Server 2012 R2. (Operating_System, Version)
Entity-Type Mismatch	**Gold:** I used ShareKit which was a breeze to add. (Library) **Predicted:** I used ShareKit which was a breeze to add. (Application)

Table 3.11: Representative examples of system errors.

- **Entity-Type Mismatch** refers to the errors where a code entity (e.g., names of variables) is predicted as a non-code entity (e.g., names of devices), and vice-versa. Our SoftNER model reduces such entity type errors by 13.54% compared to the fine-tuned BERTOverflow baseline.

As illustrated in Figure 3.6, our SoftNER model reduced the errors in both categories by incorporating the auxiliary outputs from segmenter and code recognizer model.

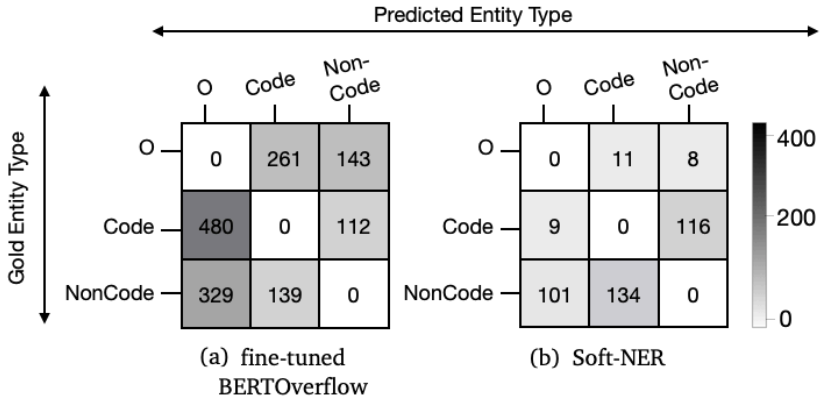


Figure 3.6: Comparison of errors made by the fine-tuned BERTOverflow baseline and our SoftNER model on the *dev* set of the StackOverflow NER corpus. In the heatmap, darker cell color corresponds to higher error counts. Our SoftNER model reduces errors in all the categories.

3.4 Domain Adaptation to GitHub data

To understand the domain adaptability of our StackOverflow based SoftNER, we evaluate its performance on readme files and issue reports from 143 randomly sampled repositories in the GitHub dump (Gousios and Spinellis, 2012). We also trained ELMo embeddings (ELMoGithub) on 4 million sentences from randomly sampled 5,000 GitHub repositories.

Table 3.12 shows that the performance of our SoftNER model using StackOverflow ELMo embeddings is similar to the top performing BiLSTM-CRF model using GitHub ELMo embeddings with a difference of only 1.61 points in F_1. We also did not observe any significant gain after adding the code recognizer and segmenter vectors to the Github ELMo embeddings. We think one likely explanation is that GitHub data contains less code-related tokens when compared to StackOverflow. The percentage of code-related entity tokens is 63.20% in GitHub and 77.21% in StackOverflow. Overall, we observe a drop of our SoftNER tagger from 79.10 F_1 on StackOverflow (Table 3.3) to 61.08 F_1 on GitHub data

(Table 3.12) in F_1 due to domain mismatch. However, we believe that our NER tagger still achieves sufficient performance to be useful for applications on GitHub.[12] We leave investigation of semi-supervised learning and other domain adaptation approaches for future work.

	P	**R**	**F_1**
Feature-Based CRF	43.16	35.71	39.09
BiLSTM-CRF (ELMoGitHub)	**64.53**	**60.96**	**62.69**
Attentive BiLSTM-CRF (ELMoVerflow)	62.05	59.20	60.59
Attentive BiLSTM-CRF (ELMoGitHub)	63.29	60.89	62.07
Fine-tuned out-of-domain BERT	56.59	48.13	52.02
Fine-tuned BERTOverflow	61.71	58.75	60.19
SoftNER (BERTOverflow)	61.92	60.26	61.08

Table 3.12: Evaluation on the GitHub NER dataset of readme files and issue posts. *All the models are trained on our StackOverflow NER corpus.* Our SoftNER model performs close to BiLSTM-CRF model trained on the GitHub ELMo embeddings.

3.5 Related Work

The CoNLL 2003 dataset (Sang and De Meulder, 2003) is a widely used benchmark for named entity recognition, which contains annotated newswire text from the Reuters RCV1 corpus. State-of-the-art approaches on this dataset (Baevski et al., 2019) use a bidirectional LSTM (Lample et al., 2016; Ma and Hovy, 2016) with conditional random field (Collobert et al., 2011) and contextualized word representations (McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019).

Named entity recognition has been explored for new domains and languages, such as social media (Finin et al., 2010; Ritter et al., 2011a; Plank et al., 2014b; Derczynski et al.,

[12] As a reference, the state-of-the-art performance for 10-class Twitter NER is 70.69 F_1 (Zhang et al., 2018).

2015; Limsopatham and Collier, 2016; Aguilar et al., 2017), biomedical texts (Collier and Kim, 2004; Greenberg et al., 2018; Kulkarni et al., 2018), multilingual texts (Benajiba et al., 2008; Xie et al., 2018) and code-switched corpora (Aguilar et al., 2018; Ball and Garrette, 2018). Various methods have been investigated for handling rare entities, for example incorporating external context (Long et al., 2017) or approaches that make use of distant supervision (Choi et al., 2018; Yang et al., 2018; Onoe and Durrett, 2019).

There has been relatively little prior work on named entity recognition in the software engineering domain. Ye et al. (2016) annotated 4,646 sentences from StackOverflow with five named entity types (Programming Language, Platform, API, Tool-Library-Framework and Software Standard). The authors used a traditional feature-based CRF to recognize these entities. In contrast, we present a much larger annotated corpus consisting of 15,372 sentences labeled with 20 fine-grained entity types. We also develop a novel attention based neural NER model to extract those fine-grained entities.

3.6 Conclusion

In this chapter, we presented the task of named entity recognition in the social computer programming domain. We put out our newly developed NER corpus of 15,372 sentences from StackOverflow and 6,510 sentences from GitHub annotated with 20 fine-grained named entities and demonstrate that this new corpus is an ideal benchmark dataset for contextual word representations, as there are many challenging ambiguities that often require long-distance context to resolve. We also propose a novel attention based model, named SoftNER, that outperforms the state-of-the-art NER models on this dataset. Furthermore, we investigated the important sub-task of code recognition. Our code recognition model

captures additional spelling information beyond then contextual word representations and consistently helps to improve the NER performance. We believe our corpus, StackOverflow-specific BERT embeddings and named entity tagger will be useful for various language-and-code tasks, such as code retrieval, software knowledge base extraction and automated question-answering.

Chapter 4: Entity and Relation Information Retrieval From the Wet Lab Protocols

Recent research has begun to apply human language technologies to extract structured representations of procedures from natural language protocols (Kuniyoshi et al., 2020; Vaucher et al., 2020; Kulkarni et al., 2018; Soldatova et al., 2014; Vasilev et al., 2011; Ananthanarayanan and Thies, 2010). Extraction of named entities and relations from these protocols is an important first step towards machine reading systems that can interpret the meaning of these noisy human generated instructions. However, performance of state-of-the-art tools for extracting named entity and relations from wet lab protocols still lags behind well edited text genres (Jiang et al., 2020). This motivates the need for continued research to create new datasets and tools for this noisy text genre.

In this chapter, we describe the development and findings of a shared task on named entity and relation extraction from the noisy wet lab protocols, which was held at the 6-th Workshop on Noisy User-generated Text (WNUT 2020) and attracted 15 participating teams. We present the design details of the task including enhanced training and development datasets in addition to the newly annotated test data. We also briefly summarize the systems developed by the participant teams, and then conclude with an analysis of the results.

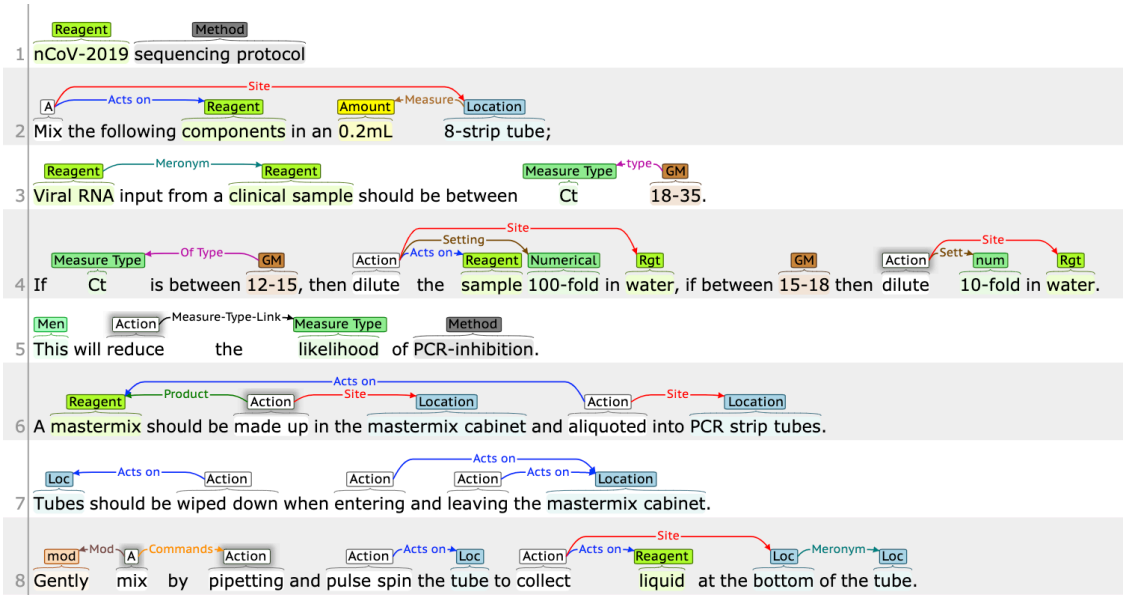


Figure 4.1: Examples of named entities and relations in a wet lab protocol

4.1 Wet Lab Protocols

Wet lab protocols consist of the guidelines from different lab procedures which involve chemicals, drugs, or other materials in liquid solutions or volatile phases. The protocols contain a sequence of steps that are followed to perform a desired task. These protocols also include general guidelines or warnings about the materials being used. The publicly available archive of `protocol.io` contains such guidelines of wet lab experiments, written by researchers and lab technicians around the world. This protocol archive covers a large spectrum of experimental procedures including neurology, epigenetics, metabolomics, stem cell biology, etc. Figure 4.1 shows a representative wet lab protocol.

The wet lab protocols, written by users from all over the worlds, contain domain specific jargon as well as numerous nonstandard spellings, abbreviations, unreliable capitalization. Such diverse and noisy style of user created protocols imposed crucial challenges for the

	Train	Dev	Test-18	Test-20	Total
#protocols	370	122	123	111	726
#sentences	8444	2839	2813	3562	17658
#tokens	107038	36106	36597	51688	231429
#entities	48197	15972	16490	104654	185313
#relations	32158	10812	11242	70591	124803

	per Protocol	per Sentence
avg. #sentences	24.32	-
avg. #tokens	318.77	13.11
avg. #entities	255.25	10.49
avg. #relation	171.90	7.07

Table 4.1: Statistics of the Wet Lab Protocol corpus.

entity and relation extraction systems. Hence, off-the-shelf named entity recognition and relation extraction tools, tuned for well edited texts, suffer a severe performance degradation when applied to noisy protocol texts (Kulkarni et al., 2018).

To address these challenges, there has been an increasing body of work on adapting entity and relation extraction recognition tools for noisy wet lab texts (Jiang et al., 2020; Luan et al., 2019; Kulkarni et al., 2018). However, different research groups have used different evaluation setups (e.g., training / test splits) making it challenging to perform direct comparisons across systems. By organizing a shared evaluation, we hope to help establish a common evaluation methodology (for at least one dataset) and also promote research and development of NLP tools for user generated wet-lab text genres.

4.1.1 Annotated Corpus

Our annotated wet lab corpus includes 726 experimental protocols from the 8-year archive of ProtocolIO (April 2012 to March 2020). These protocols are manually annotated

with 15 types of relations among the 18 entity types[13]. The fine-grained entities can be broadly classified into 5[14] categories: ACTION, CONSTITUENTS, QUANTIFIERS, SPECIFIERS, and MODIFIERS. The CONSTITUENTS category includes mentions of REAGENT, LOCATION, DEVICE, MENTION, and SEAL. The QUANTIFIERS category includes mentions of AMOUNT, CONCENTRATION, SIZE, TIME, TEMPERATURE, PH, SPEED, GENERIC-MEASURE and NUMERICAL. The SPECIFIERS category includes mentions of MODIFIER, MEASURE-TYPE and METHOD. The ACTION entity refers to the phrases denoting tasks that are performed to complete a step in the protocol. The mentions of these entities contain different types of relations, including– SITE, SETTING, CREATES, MEASURE-TYPE-LINK, CO-REFERENCE-LINK, MOD-LINK, COUNT, MERONYM, USING, MEASURE, COMMANDS, OF-TYPE, OR, PRODUCT, and ACTS-ON.

4.1.1.1 Train and Development data

The training and development dataset for our task was taken from previous work on wet lab corpus (Kulkarni et al., 2018) that consists of from the 623 protocols. We excluded the eight duplicate protocols from this dataset and then re-annotated the 615 unique protocols in BRAT (Stenetorp et al., 2012b). This re-annotation process aided us to add the previously missing 20,613 missing entities along with 10,824 previously missing relations and also to facilitate removing the inconsistent annotations. The updated corpus statics is provided in Table 3.1. This full dataset (Train, Dev, Test-18) was provided to the participants at the beginning of the task and they were allowed to use any of part of this dataset to train their final model.

[13]Our annotated corpus is available at: https://github.com/jeniyat/WNUT_2020_NER.

[14]The fiver broader entity categories are created by grouping together the entities with respect to their role in the experiments.

4.1.2 Test Data Annotation

For this shared task we added 111 new protocols (Test-20) which were used to evaluate the submitted models. Test-20 dataset consists of 100 randomly sampled general protocols and 11 manually selected covid-related protocols from protocols.io. This 111 protocols were double annotated by three annotators using a web-based annotation tool, BRAT (Stenetorp et al., 2012b). Figure 4.1 presents a screenshot of our annotation interface. We also provided the annotators a set of guidelines containing the entity and relation type definitions. The annotation task was split in multiple iterations. In each iteration, an annotator was given a set of 10 protocols. An adjudicator then went through all the entity and relation annotations in these protocols and resolved the disagreements. Before adjudication, the inter-annotator agreement is 0.75 , measured by Cohen's Kappa (Cohen, 1960).

4.1.3 Baseline Model

We provided the participants baseline model for both of the subtasks. The baseline model for named entity recognition task utilized a feature-based CRF tagger developed using the CRF-Suite[15] with a standard set of contextual, lexical and gazetteer features. The baseline relation extraction system employed a feature-based logistic regression model developed using the Scikit-Learn[16] with a standard set of contextual, lexical and gazetter features.

4.1.4 NER Systems

Thirteen teams (Table 4.2) participated in the named entity recognition sub-task. A wide variety of approaches were taken to tackle this task. Table 4.3 summarizes the word

[15] http://www.chokkan.org/software/crfsuite/

[16] https://scikit-learn.org/

Team Name	Affiliation
B-NLP	Bosch Center for Artificial Intelligence
Big Green	Dartmouth College
BIO-BIO	Harbin Institute of technology, Shenzhen
BITEM	University of Applied Sciences and Arts of Western Switzerland, Swiss Institute of Bioinformatics, University of Geneva
DSC-IITISM	IIT(ISM) Dhanbad
Fancy Man	University of Manchester, Xian Jiaotong University, East China University of Science and Technology, Zhejiang University
IBS	IBS Software Pvt. Ltd, NTNU
Kabir	Microsoft
KaushikAcharya	Philips
mahab	Amirkabir University of Technology
mgsohrab	National Institute of Advanced Industrial Science and Technology
PublishInCovid19	Flipkart Private Limited
SudeshnaTCS	TCS Research & Innovation Lab
IITKGP	IIT, Kharagpur

Table 4.2: Team Name and affiliation of the participant.

Team	Word Representation	Features	Approach
BITEM	BERT, BioBERT, RoBERTa, XLNet	-	Ensemble of Transformers
PublishInCovid19	PubMedBERT	-	Ensemble of BiLSTM-CRFs
Fancy Man	BERT	-	BERT fine tuning
mahab	BERT	Lexical Features	BERT fine tuning
mgsohrab	SciBERT	Lexical Features	SciBERT fine tuning
SudeshnaTCS	XLNet	Rules	XLNet fine tuning
IITKGP	BioBERT	-	BioBERT fine tuning
B-NLP	SciBERT, word2vec	-	Biaffine Classifier
BIO-BIO	BioBERT	-	BiLSTM-CRF
DSC-IITISM	GLoVe, CamemBERT, Flair	-	BiLSTM-CRF
Kabir	GLoVe, ELMo, BERT, Flair	Gazetteers	RNN-CRF
IBS	-	Gazetteers, POS Tagger	Ensemble of CRFs
KaushikAcharya	-	POS Tagger, Dependency Parser	CRF

Table 4.3: Summary of NER systems designed by each team.

representations, features and the machine learning approaches taken by each team. Majority of the teams (11 out of 13) utilized contextual word representations. Four teams combined the contextual word representations with global word vectors. Only two teams did not use any type of word representations and relied entirely on hand-engineered features and a CRF taggers. The best performing teams utilized a combination of contextual word representation

with ensemble of learning. Below we provide a brief description of the approach taken by each team.

B-NLP (Lange et al., 2020) modeled the NER as a parsing task and uses a biaffine classifier. The second classifier of their system used the predictions from the first classifier and then updates the labels of the predicted entities. Both of the classifiers utilized word2vec (Mikolov et al., 2013) and SciBERT (Lee et al., 2019) word representations.

BIO-BIO (Kecheng et al., 2020) implemented a BiLSTM-CRF tagger that utilized BioBERT (Lee et al., 2020) word representation.

BITEM (Knafou et al., 2020) developed a voting based ensemble classifier that consists of 14 transformer models that utilized 7 differrent word representations including BERT (Devlin et al., 2019), ClinicalBERT (Huang et al., 2019), PubMedBERT(base) (Gu et al., 2020), BioBERT (Lee et al., 2020), RoBERTa (Liu et al., 2019), Biomed-RoBERTa(base) (Gururangan et al., 2020) and XLNet (Yang et al., 2019).

DSC-IITISM (Gupta et al., 2020) developed a BiLSTM-CRF model that utilized a concatenation of CamemBERT(base) (Martin et al., 2020), Flair(PubMed) (Akbik et al., 2018), and GloVe(en) (Pennington et al., 2014) word representations.

Fancy Man (Zeng et al., 2020) fine-tuned the BERT(base) (Devlin et al., 2019) model with an additional linear layer.

IBS (Sikdar et al., 2020) system used an ensemble classifier consists of 4 feature based on CRF taggers.

Kabir (Khan, 2020) system used a RNN-CRF model that utilized a concatenation of Flair(PubMed) (Akbik et al., 2018) and ELMo(PubMed) (Peters et al., 2018) word representation.

KaushikAcharya (Acharya, 2020) system used a linear CRF with hand-crafted features.

mahab (Pour and Farinnia, 2020) system fine-tuned the BERT(base) (Devlin et al., 2019) sequence tagging model.

mgsohrab (Sohrab et al., 2020) fine-tuned the SciBERT (Beltagy et al., 2019) WLP corpus.

PublishInCovid19 (Singh and Wadhawan, 2020) used a structured ensemble classifier (Nguyen and Guo, 2007) consisting of 11 BiLSTM-CRF taggers, that utilized the PubMedBERT (Gu et al., 2020) word representation.

SudeshnaTCS (Jana, 2020) fine-tuned the XLNet (Yang et al., 2019) on WLP corpus.

IITKGP (Kaushal and Vaidhya, 2020) fine-tuned the Bio-BERT (Lee et al., 2020) WLP corpus.

4.1.5 Relation Extraction Systems

Two teams (Table 4.2) participated in the relation extraction sub-task. Both of the teams followed fine-tuning of contextual word representation and did not use any hand-crafted features. Table 4.4 summarizes the word representations and the machine learning approaches followed by each team. Below we provide a brief description of the model developed by taken by each team.

Big Green (Miller and Vosoughi, 2020) considered the protocols as a knowledge graph, in which relationships between entities are edges in the knowledge graph. They trained a BERT (Devlin et al., 2019) based system to classify edge presence and type between two entities, given entity text, label, and local context.

mgsohrab (Sohrab et al., 2020) utilized PubMedBERT (Gu et al., 2020) as input to the relation extraction model that enumerates all possible pairs of arguments using deep exhaustive span representation approach.

Team	Word Representation	Features	Approach
mgsohrab	PubMedBERT	-	PubMedBERT fine-tuning
Big Green	BERT	-	BERT fine-tuning
Baseline	-	Gazetteers, Lexical, Contextual	Logistic Regression

Table 4.4: Summary of relation extraction systems designed by each team.

4.2 Evaluation

In this section, we present the performance of each participating systems along with a description of the errors made by the model types.

4.2.1 NER Errors Analysis

Table 4.5 shows the comparison of precision (**P**), recall (**R**) and $\mathbf{F}_1$ score among different teams, evaluated on the *Test-20* corpus. Here, the exact match refers to the case where predicted entity type and boundary are exactly same as the gold entity type and boundary. Whereas, the partial match refers to the case where the predicted entity type is same as the gold entity type and predicted entity boundary has some overlap with the gold entity boundary.

We observe that ensemble models with contextual word representations outperforms all other approaches by achieving 77.99 F_1 score in exact match (Team:BITEM) and 81.75 F_1 score in partial match (Team:PublishInCovid19).

In Figure 4.2, we present an error analysis of different NER systems.

	P	**R**	**F_1**
Exact Match			
BITEM	**84.73**	72.25	**77.99**
PublishInCovid19	81.36	**74.12**	77.57
Fancy Man	76.21	71.76	73.92
mahab	50.19	52.96	51.54
mgsohrab	83.69	70.62	76.60
SudeshnaTCS	74.99	71.43	73.16
IITKGP	77.00	72.93	74.91
B-NLP	77.95	63.93	70.25
BIO-BIO	78.49	71.06	74.59
DSC-IITISM	64.20	57.07	60.42
Kabir	78.79	72.20	75.35
IBS	74.26	62.55	67.90
KaushikAcharya	73.68	63.98	68.48
Baseline	70.06	61.91	65.73
Ensemble	78.35	75.39	76.84

	P	**R**	**F_1**
Partial Match			
BITEM	**88.72**	75.66	81.67
PublishInCovid19	85.74	**78.11**	**81.75**
Fancy Man	81.15	76.41	78.71
mahab	55.09	58.14	56.57
mgsohrab	87.95	74.22	80.50
SudeshnaTCS	79.73	75.95	77.80
IITKGP	81.76	77.43	79.54
B-NLP	84.85	69.59	76.46
BIO-BIO	83.16	75.29	79.03
DSC-IITISM	68.52	60.90	64.49
Kabir	83.73	76.73	80.08
IBS	79.72	67.15	72.89
KaushikAcharya	79.31	68.87	73.73
Baseline	75.66	66.85	70.98
Ensemble	84.16	79.15	81.58

Table 4.5: Results on extraction of 20 Named Entity types from the *Test-20* dataset. **Exact Match** reports the performance when the predicted entity type is same as the gold entity and the predicted entity boundary is the exact same as the gold entity boundary. **Partial Match** reports the performance when the predicted entity type is same as the gold entity and the predicted entity boundary has some overlap with gold entity boundary.

Analysis of the errors these different NER model prediction demonstrate that, the BERT based models make less mistakes in false positive and incorrect type errors compared to the traditional neural networks and feature based models. We also observed that, these BERT models suffer from higher false negatives errors compared to the other approaches.

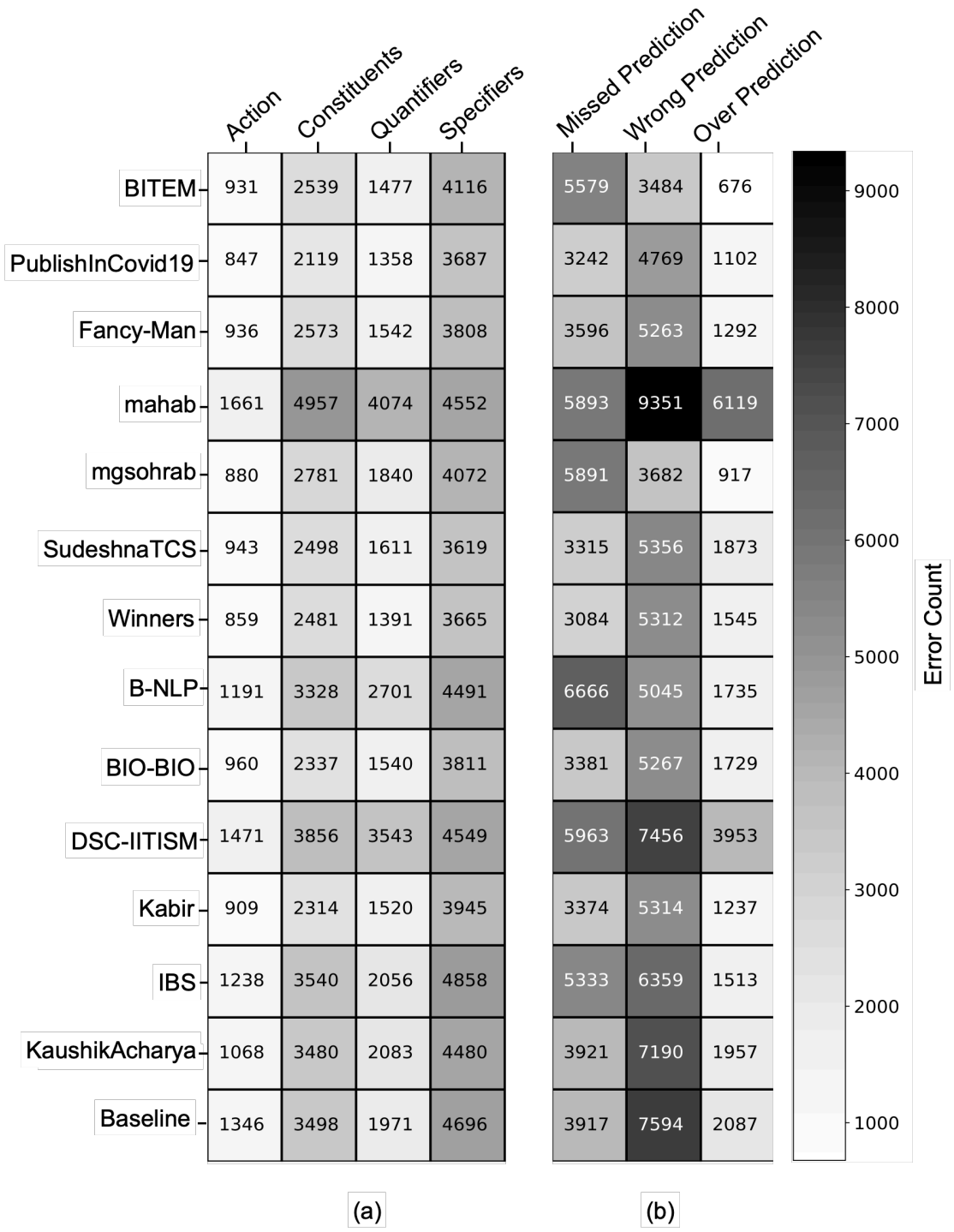


Figure 4.2: Summary of incorrectly classified entity tokens by each submitted systems.

To combine the advantages of these different approaches, we made an majority voting based ensemble classifier. Our ensemble NER tagger utilizes the predictions of all the submitted systems and then it assigns each word the most frequently predicted tag. This ensemble classifier performs better than all the single fine-tuned BERT models and it outperformed the traditional neural and feature based models by achieving 76.84 F_1 (Table 4.5). However, our ensemble NER tagger performed 1.15 F_1 below the neural ensemble models (Team:BITEM, PublishInCovid19). We would like to note that, we did not have access to the participant model's predictions on development and training set. Hence, it was not possible for us to fine-tune our ensemble classifier on the entity recognition task.

4.2.2 RE Errors Analysis

Table 4.6 shows the comparison of precision (**P**), recall (**R**) and $\mathbf{F}_1$ score among the participating teams, evaluated on the *Test-20* corpus. Both of the teams utilized the gold entities and then predicted the relations among these entities by fine-tuning contextual word representations. We observed that fine-tuning of domain related PubMedBERT provides significantly higher performance compared to the general BERT fine-tuning. While examining the relation predictions from both of these systems, we found that system with PubMedBERT fine-tuning (Team:mgsohrab) resulted in significantly less amount of errors in every category (Figure 4.3).

	P	**R**	$\mathbf{F}_1$
mgsohrab	**80.86**	80.07	**80.46**
Big Green	45.42	**86.54**	59.57
Baseline	80.10	66.21	72.50
Ensemble	82.65	80.32	81.32

Table 4.6: Results on extraction of 31 relation types from the *Test-20* dataset.

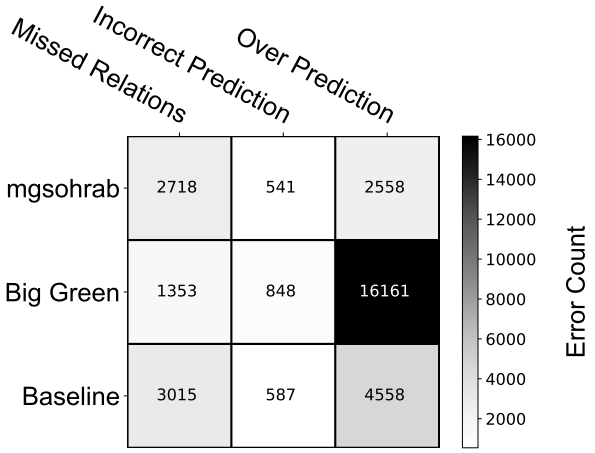


Figure 4.3: Summary of incorrectly predicted relations in each submitted systems.

The error analysis over different participant predictions revealed that the general domain BERT has less false negative errors compared to the domain-related BERT. However, the domain related Pubmed-BERT models have significantly less number of false positive and incorrect type errors compared to the general domain BERT. To combine the advantages of these different approaches, we made an ensemble classifier from the prediction of the submitted systems, where we assign the most frequently predicted relation for each entity pair. This ensemble classifier outperforms the winner system by achieving 81.32 F_1 score.

4.3 Related Work

The task of information extraction from wet lab protocols is closely related to the event trigger extraction task. The event trigger task has been studied extensively, mostly using ACE data (Doddington et al., 2004) and the BioNLP data (Nédellec et al., 2013).

Broadly, there are two ways to classify various event trigger detection models: (1) *Rule-based* methods using pattern matching and regular expression to identify triggers (Vlachos et al., 2009) and (2) *Machine Learning based* methods focusing on generation of high-end hand-crafted features to be used in classification models like SVMs or maxent classifiers (Pyysalo et al., 2012). Kernel based learning methods have also been utilized with embedded features from the syntactic and semantic contexts to identify and extract the biomedical event entities (Zhou et al., 2014). In order to counteract highly sparse representations, different neural models were proposed. These neural models utilized the dependency based word embeddings with feed forward neural networks (Wang et al., 2016b), CNNs (Wang et al., 2016a) and Bidirectional RNNs (Rahul et al., 2017).

Previous work has experimented on datasets of well-edited biomedical publications with a small number of entity types. For example, the JNLPBA corpus (Kim et al., 2004) with 5 entity types (`CELL LINE`, `CELL TYPE`, `DNA`, `RNA`, and `PROTEIN`) and the BC2GM corpus (Hirschman et al., 2005) with a single entity class for genes/proteins. In contrast, our dataset addresses the challenges of recognizing 20 fine-grained named entities along with 17 types of relations from the user-created wet lab protocols.

4.4 Conclusion

In this chapter, we presented a shared task for wet lab protocols, consisting of two sub-tasks: named entity recognition and relation extraction from the wet lab protocols. We described the task setup and datasets details, and also outlined the approaches taken by the participating systems. The shared task included larger and improvised dataset compared to the prior literature (Kulkarni et al., 2018). This improvised dataset enables us to draw

stronger conclusions about the true potential of different approaches. It also facilitates us in analyzing the results of the participating systems, which aids us in suggesting potential research directions for both future shared tasks and noisy text processing in user generated lab protocols.

Chapter 5: Conclusion and Future Work

In this thesis, we explored the challenges and opportunities that arises while extracting information from the user generated web texts. We showed that off-the-shelf NLP tools are insufficient for handling the noisy and uniquely-styled user generated social media texts. Hence, we developed a set of NLP tools[17] which have been tuned to extract the entity information from different social domains including Twitter, StackOverflow and ProtocolIO. From our experiments on these different web domains, we found that in-domain contextualized word representations are crucial for the information extraction models. In fact, we found that the performance of the general domain BERT models drop significantly when we move to a new domain, even if the novel words from that domain are present in the training data for the fine-tuning BERT model. This reveals the lack of generalizibility of the pre-trained transformer models in special domains and emphasizes the importance of domain specific or related contextual word representations.

Moving forward, there are many unsolved and fundamental challenges to be addressed towards the construction of scalable natural language understanding systems for user generated texts. Below we describe several possible directions for future work to extract structured information from noisy text of different social domains.

[17] github.com/jeniyat/

5.1 BiLinear Model for Time Entity Extraction

In Chapter 2, we presented a temporal entity extraction method from tweets that utilizes a combination of lexical features with numerical date features in a multiple instance learning setup. The performance of this model could be further improved by employing a domain related contextual word representation. In future, we would like to experiment with a bi-linear model (Toutanova et al., 2016) to combine the sentence level contextualized tweet representation from BERTweet (Nguyen et al., 2020) with the numerical date features. The bi-linear model is expected to restrict the word embedding dimension so that it would be equal to the dimension of the date representations. As the time expression in Tweets are highly context dependent, we believe such contextualized temporal resolver would be able to extract the time entities with higher precision.

5.2 Corpus Enrichment for Software Entity Extraction

In Chapter 3, we presented software domain NER models, capable of named entities from the code-mixed text of the social software networks. The performance of these NER models can be further improved with additional training data from similar software domain texts. In this section, we propose a method of collecting large scale entity tagged data without human annotation efforts (§5.2.1) as well as the process of creating a perturbed test data (§5.2.2) which can help us to understand the model's linguistic understandability.

5.2.1 Tag-Definition Supervision

An NER corpus can be enhanced with the usage of large knowledge base by increasing the annotation coverage through the inclusion of diversified entities. For software domain

named entities, such knowledge base enhancement can be provided by the StackOverflow tagset[18]. We have the following observation from the tag definitions:

> The first sentence of a StackOverflow tag definition often states the entity's type via an 'is a' relation.

For example, consider the definitions of 'python' and 'array' from the StackOverflow tagset: (i) *Python is a multi-paradigm, dynamically typed, multipurpose programming* LANGUAGE (ii) *An array is an ordered linear* DATA STRUCTURE. Both of these definitions contain the most commonly used entity type for the corresponding tag. The full StackOverflow archive contains 62,316 tags and many of them contains such 'is a' relation inside their definitions. The entity types extracted from these 'is a' relation, combined with the tag name, can be utilized to substantially increase the size of our training data as well as the entity vocabulary. This would be helpful to further improve the performance of the software domain NER models.

5.2.2 Contrast Set Creation

Contrast set refers to a collection of manually perturbed test set instances created in order to address the unintentional systematic gaps (e.g., annotation artifacts) in an NLP test corpora (Gardner et al., 2020). We propose to create a contrast set for StackOverflow NER corpus by manually perturbing the words in selected sentences in order to create a harder set of test examples. Table 5.1 contains different representative example of the contrast set creation.

[18] https://stackoverflow.com/tags

In our test set, we have 2730 sentences that contain ambiguous entities. We can perturb the words from these sentences by changing the types of these ambiguous entities as depicted in Example 1 and 2.

Our test data contains sentences from different programming languages. We can perturb these sentences by combining the part of the sentences from these different languages as shown in Example 3 and 4.

Another possible way of creating perturbed sentences would to replace the entity tokens with their synonyms that are not commonly used in the software domain as shown in Example 5 and 6.

Example 1	Say , I have a vector *name* str[VARIABLE] , which contain many strings like : Say , I have a vector *of* str[DATA STRUCTURE] , which contain many strings like:
Example 2	Each *ivec4*[CLASS] represents a quarter of the screen Each *table*[UI-ELEMENT] represents a quarter of the screen
Example 3	*Use the java*[LANGUAGE] *AirplaneModEnabler*[CLASS] where the handler[CLASS] looks for EVENT_SERVICE[VARIABLE] to toggle the state of the checkbox[UI-ELEMENT] and to display the summary. Get SQL[UI-ELEMENT] statement generated by django[LIBRARY] *with the python*[LANGUAGE] *smanager*[LIBRARY] *library*. *Get SQL*[LANGUAGE] *statement generated by django*[LIBRARY] *where the handler*[CLASS] *looks for EVENT_SERVICE*[VARIABLE] *to toggle the state of the checkbox*[UI-ELEMENT] *and to display the summary.*
Example 4	*This should be defined* as static const char*[DATA-TYPE] *in the header*[UI-ELEMENT] and then added to ./abc.conf[FILE-NAME]: I have a MySQL[LANGUAGE] in the backend *but most of the tables*[DATA-STRUCTURE] *are auto-generated*. *I have a MySQL*[LANGUAGE] *in the backend as static const char**[DATA-TYPE] *added to ./abc.conf*[FILE-NAME] *:*
Example 5	However input is still a *text*[FILE-TYPE] file which has .xls[FILE-TYPE] *extension* However input is still a *passage*[FILE-TYPE] file which has .xls[FILE-TYPE] *at the end*
Example 6	JSP[CLASS] is deprecated in JSF[LIBRARY] *version* 2.0[VERSION]. JSP[CLASS] is deprecated in JSF[LIBRARY] *rendition* 2.0[VERSION].

Table 5.1: Contrast set creation. In each example, the original sentences are shown in black and the manually perturbed sentences are shown in brown.

5.3 Joint Entity and Relation Extraction from ProtocolIO

In Chapter 4, we presented a shared task for extracting named entities and relations from the noisy ProtocolIO texts. All the participants of this task employed separate models for entity and relation extraction. Here the relation extraction models assumed the presence of gold entities and utilized the features from those labeled entities. However, in a more realistic setting, the "gold" entity labels are not available for the target sentence. Hence, as a future work, we would like to explore models which are capable of making joint prediction of entity and relations from this wet lab corpus. A possible method would be to utilize a combination of domain related contextual embeddings (i.e., PubMedBERT (Gu et al., 2020), SciBERT (Beltagy et al., 2019)) to create a shared span representation and share them with both NER and RE models. Next, these shared span representations will be updated utilizing the scoring functions of the NER and RE task by applying span graph propagation methods (Luan et al., 2019).

Bibliography

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Kaushik Acharya. 2020. KaushikAcharya at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named Entity Recognition on Code-Switched Data: Overview of the CALCS 2018 Shared Task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*.

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López-Monroy, and Thamar Solorio. 2017. A Multi-task Approach for Named Entity Recognition in Social Media Data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text (WNUT)*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*.

Omar Alonso, Michael Gertz, and Ricardo Baeza-Yates. 2007. On the value of temporal information in information retrieval. In *ACM SIGIR Forum*.

Shirani Amirreza, Bowen Xu Xu, David Lo, Thamar Solorio, and Amin Alipour. 2019. Question relatedness on stack overflow: The task, dataset, and corpus-inspired models. In *AAAI Reasoning for Complex Question Answering Workshop (AAAI 2019)*.

Vaishnavi Ananthanarayanan and William Thies. 2010. Biocoder: A programming language for standardizing and automating biology protocols. *Journal of biological engineering*.

Gabor Angeli, Christopher D Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Gabor Angeli and Jakob Uszkoreit. 2013. Language-independent discriminative parsing of temporal expressions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention 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)*.

Kelsey Ball and Dan Garrette. 2018. Part-of-Speech Tagging for Code-Switched, Transliterated Texts without Explicit Language Identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Maxwell Bates, Aaron J Berliner, Joe Lachoff, Paul R Jaschke, and Eli S Groban. 2017. Wet lab accelerator: a web-based application democratizing laboratory automation for synthetic biology. *ACS synthetic biology*.

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. SciBERT: Pretrained Contextualized Embeddings for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yassine Benajiba, Mona Diab, and Paolo Rosso. 2008. Arabic Named Entity Recognition using Optimized Feature Sets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Steven Bethard. 2013a. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*.

Steven Bethard. 2013b. A synchronous context free grammar for time normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Steven Bethard and Guergana Savova. 2016. SemEval-2016 Task 12: Clinical TempEval. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval)*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to extract relations from the Web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nathanael Chambers. 2013. NavyTime: Event and time ordering from raw text. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval)*.

Angel X Chang and Christopher D Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*.

Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. 2016. Expectation-regulated neural model for event mention extraction. *Proccedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Technologies (NAACL)*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the Bio-entity Recognition Task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research (JMLR)*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB)*.

Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2019. Using Similarity Measures to Select Pretraining Data for NER. *arXiv preprint arXiv:1904.00585*.

Leon Derczynski and Robert J Gaizauskas. 2013. Temporal signals help label temporal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke Van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of Named Entity Recognition and Linking for Tweets. *Information Processing & Management*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective deep memory networks for distant supervised relation extraction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*.

Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. ManTIME: Temporal expression identification and normalization in the TempEval-3 challenge. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating Named Entities in Twitter Data with Crowdsourcing. In *Proceedings of the Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hanna Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating nlp models via contrast sets. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

John M Giorgi and Gary Bader. 2018. Transfer Learning for Biomedical Named Entity Recognition with Neural Networks. *bioRxiv*.

G. Gousios and D. Spinellis. 2012. GHTorrent: Github's Data from a Firehose. In *Proceedings of the 9th IEEEConference on Mining Software Repositories (MSR)*.

Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal Likelihood Training of BiLSTM-CRF for Biomedical Named Entity Recognition from Disjoint Label Sets. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *arXiv preprint arXiv:2007.15779*.

Saket Gupta, Aman Sinha, and Rohit Agarwal. 2020. DSC-IITISM at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distant supervision for relation extraction via instance-level adversarial training.

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. 2005. Overview of biocreative: critical assessment of information extraction for biology.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011a. Knowledge-based weak supervision for information extraction of overlapping relations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011b. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing Source Code using a Neural Attention Model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sudeshna Jana. 2020. SudeshnaTCS at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2011. Overview of the tac 2011 knowledge base population track. In *Proceedings of the Fourth Text Analysis Conference (TAC)*.

Zhengbao Jiang, Wei Xu, Jun Araki, and Graham Neubig. 2020. Generalizing natural language analysis through span-relation representations. In *Proceedings of the 2020 Conference of the Association for Computational Linguistics*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing Text Classification Models. *arXiv preprint arXiv:1612.03651*.

Nattiya Kanhabua, Sara Romano, Avaré Stewart, and Wolfgang Nejdl. 2012. Supporting temporal analytics for health-related events in microblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM)*.

Ayush Kaushal and Tejas Vaidhya. 2020. IITKGP at WNUT 2020 Shared Task-1: Domain specific BERT representation for Named Entity Recognition of lab protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Zhan Kecheng, Xiong Ying, Peng Hao, Yao, and LiQing Yao. 2020. BIO-BIO at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Kabir Khan. 2020. kabir at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*.

Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, Andrew Sparkes, Kenneth E Whelan, and Amanda Clare. 2009. The automation of science. *Science*.

Julien Knafou, Nona Naderi, Jenny Copara, Douglas Teodoro, and Patrick Ruch. 2020. BITEM at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Oleksandr Kolomiyets and Marie-Francine Moens. 2010. KUL: Recognition and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. 2018. An Annotated Corpus for Machine Reading of Instructions in Wet Lab Protocols. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Fusataka Kuniyoshi, Kohei Makino, Jun Ozawa, and Makoto Miwa. 2020. Annotating and extracting synthesis process of all-solid-state batteries from scientific literature. *arXiv:2002.07339*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 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 (NAACL-HLT).

Lukas Lange, Xiang Dai, Heike Adel, and Jannik Strötgen. 2020. B-NLP at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kai Lei, Daoyuan Chen, Yaliang Li, Nan Du, Min Yang, Wei Fan, and Ying Shen. 2018. Cooperative denoising for distantly supervised relation extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Nut Limsopatham and Nigel Collier. 2016. Bidirectional LSTM for Named Entity Recognition in Twitter Messages. In *Proceedings of 2016 the Workshop on Noisy User-generated Text (WNUT)*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Xiao Ling and Daniel S Weld. 2010. Temporal information extraction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*.

Dan Liu, Wei Lin, Shiliang Zhang, Si Wei, and Hui Jiang. 2016. Neural Networks Models for Entity Discovery and Linking. *arXiv preprint arXiv:1611.03558*.

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

Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2017. World Knowledge for Reading Comprehension: Rare Entity Prediction with Hierarchical LSTMs Using External Descriptions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Xuezhe Ma 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)*.

Mounica Maddela and Wei Xu. 2018. A Word-Complexity Lexicon and A Neural Readability Ranking Model for Lexical Simplification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL)*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 2014 Association for Computational Linguistics System Demonstrations (ACL)*.

Micol Marchetti-Bowick and Nathanael Chambers. 2012. Learning for microblogs with distant supervision: Political forecasting with Twitter. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Proceedings of the 2017 Conference on Neural Information Processing Systems (NeurIPS)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Chris Miller and Soroush Vosoughi. 2020. Big Green at WNUT 2020 Shared Task-1: Relation Extraction as Contextualized Sequence Classification. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009a. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the Association of Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009b. Distant supervision for relation extraction without labeled data. In *Proceedigns of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL)*.

Dana Movshovitz-Attias and William W Cohen. 2015. KB-LDA: Jointly Learning a Knowledge Base of Hierarchy, Relations, and Facts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. 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*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-Scale Knowledge Extraction*.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*. Association for Computational Linguistics Sofia, Bulgaria.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th international conference on Machine learning*.

Truc-Vien T. Nguyen and Alessandro Moschitti. 2011. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yasumasa Onoe and Greg Durrett. 2019. Learning to Denoise Distantly-Labeled Data for Entity Typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NACL)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the of Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Barbara Plank, Dirk Hovy, Ryan McDonald, and Anders Søgaard. 2014a. Adapting taggers to twitter with not-so-distant supervision. In *Proceedings of the COLING 2014*.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014b. Learning part-of-speech taggers with inter-annotator agreement loss. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Mohammad Mahdi Abdollah Pour and Parsa Farinnia. 2020. mahab at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Sampo Pyysalo, Tomoko Ohta, Makoto Miwa, Han-Cheol Cho, Jun'ichi Tsujii, and Sophia Ananiadou. 2012. Event extraction across multiple levels of biological organization. *Bioinformatics*.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Patchigolla VSS Rahul, Sunil Kumar Sahu, and Ashish Anand. 2017. Biomedical event trigger identification using bidirectional recurrent neural network based models. *arXiv preprint arXiv:1705.09516*.

Sujith Ravi, Bo Pang, Vibhor Rastogi, and Ravi Kumar. 2014. Great Question! Question Quality in Community Q&A. In *Eighth International AAAI Conference on Weblogs and Social Media*.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. AFET: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedigns of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011a. Named Entity Recognition in Tweets: An Experimental Study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alan Ritter, Mausam, Sam Clark, and Oren Etzioni. 2011b. Named entity recognition in Tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*.

Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. 2015. Weakly supervised extraction of computer security events from Twitter. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics (TACL)*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*.

H Andrew Schwartz, Greg Park, Maarten Sap, Evan Weingarten, Johannes Eichstaedt, Margaret Kern, Jonah Berger, Martin Seligman, and Lyle Ungar. 2015. Extracting human temporal orientation in Facebook language. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Golnar Sheikhshabbafghi, Inanc Birol, and Anoop Sarkar. 2018. In-domain Context-aware Token Embeddings Improve Biomedical Named Entity Recognition. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*.

Utpal Kumar Sikdar, Bjorn Gamback, and M Krishana Kumar. 2020. IBS at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2017. Neural Cross-Lingual Entity Linking. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Janvijay Singh and Anshul Wadhawan. 2020. PublishInCovid19 at WNUT 2020 Shared Task-1: Entity Recognition in Wet Lab Protocols using Structured Learning Ensemble and Contextualised Embeddings. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Mohammad Golam Sohrab, Khoa Duong, Makoto Miwa, and Hiroya Takamura. 2020. mgsohrab at WNUT 2020 Shared Task-1: Name Entity and Relation Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Larisa N Soldatova, Daniel Nadis, Ross D King, Piyali S Basu, Emma Haddi, Véronique Baumlé, Nigel J Saunders, Wolfgang Marwan, and Brian B Rudkin. 2014. Exact2: the semantics of biomedical protocols. *BMC bioinformatics*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012a. brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012b. brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*.

Jannik Strötgen and Michael Gertz. 2015. A baseline temporal tagger for all languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012a. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012b. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Jeniya Tabassum, Sydney Lee, Wei Xu, and Alan Ritter. 2020a. WNUT-2020 Task 1 Overview: Extracting Entities and Relations from Wet Lab Protocols. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. 2020b. Code and named entity recognition in stackoverflow. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeniya Tabassum, Alan Ritter, and Wei Xu. 2016. A minimally supervised method for recognizing and normalizing time expressions in twitter. In *Proceedings of The 2016 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Naushad UzZaman and James F Allen. 2010. TRIPS and TRIOS system for Tempeval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TEMPEVAL-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval)*.

Viktor Vasilev, Chenkai Liu, Traci Haddock, Swapnil Bhatia, Aaron Adler, Fusun Yaman, Jacob Beal, Jonathan Babb, Ron Weiss, Douglas Densmore, et al. 2011. A software stack for specification and robotic execution of protocols for synthetic biological engineering. In *3rd international workshop on bio-design automation*.

Alain C Vaucher, Federico Zipoli, Joppe Geluykens, Vishnu H Nair, Philippe Schwaller, and Teodoro Laino. 2020. Automated extraction of chemical synthesis actions from experimental procedures. *ChemRxiv*.

Andreas Vlachos, Paula Buttery, Diarmuid O Séaghdha, and Ted Briscoe. 2009. Biomedical event extraction without training data. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. Association for Computational Linguistics.

Byron C Wallace, Joël Kuiper, Aakash Sharma, Mingxi Brian Zhu, and Iain J Marshall. 2016. Extracting PICO sentences from clinical trial reports using supervised distant supervision. *Journal of Machine Learning Research (JMLR)*.

Jian Wang, Honglei Li, Yuan An, Hongfei Lin, and Zhihao Yang. 2016a. Biomedical event trigger detection based on convolutional neural network. *International Journal of Data Mining and Bioinformatics*.

Jian Wang, Jianhai Zhang, Yuan An, Hongfei Lin, Zhihao Yang, Yijia Zhang, and Yuanyuan Sun. 2016b. Biomedical event trigger detection by dependency-based word embedding. *BMC medical genomics*.

Minghao Wu, Fei Liu, and Trevor Cohn. 2018. Evaluating the Utility of Hand-crafted Features in Sequence Labelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A Smith, and Jaime Carbonell. 2018. Neural Cross-Lingual Named Entity Recognition with Minimal Resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wei Xu, Raphael Hoffmann, Zhao Le, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics (TACL)*.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly Supervised NER with Partial Annotation Learning and Reinforcement Learning. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Ziyu Yao, Jayavardhan Reddy Peddamail, and Huan Sun. 2019. CoaCor: Code Annotation for Code Retrieval with Reinforcement Learning. In *Proceedings of the World Wide Web Conference (WWW)*.

Deheng Ye, Zhenchang Xing, Chee Yong Foo, Zi Qun Ang, Jing Li, and Nachiket Kapre. 2016. Software-specific Named Entity Recognition in Software Engineering Social Content. In *Proceedings of the 2016 IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A Transition-based Neural Abstract Syntax Parser for Semantic Parsing and Code Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*.

Qingcheng Zeng, Haoding Meng, Xiaoyang Fang, and Zhexin Liang. 2020. Fancy Man Launches Zippo at WNUT 2020 Shared Task-1: Name Entity Extraction from Wet Lab Protocol. In *Proceedings of EMNLP 2020 Workshop on Noisy User-generated Text (WNUT)*.

Qi Zhang, Jinlan Fu, Xiaoyu Liu, and Xuanjing Huang. 2018. Adaptive Co-attention Network for Named Entity Recognition in Tweets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Deyu Zhou, Dayou Zhong, and Yulan He. 2014. Event trigger identification for biomedical events extraction using domain knowledge. *Bioinformatics*.