Improved Clinical Abbreviation Expansion via Non-Sense-Based Approaches

Juyong Kim
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213

Linyuan Gong
Department of EECS, University of California, Berkeley, CA 94720

Justin Khim
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213

Jeremy C. Weiss
Heinz College of Information Systems and Public Policy, Carnegie Mellon University, Pittsburgh, PA 15213

Pradeep Ravikumar
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213

Editors: Emily Alsentzer®, Matthew B. A. McDermott®, Fabian Falck, Suproteem K. Sarkar, Subhrajit Roy‡, Stephanie L. Hyland‡

Abstract

Abbreviation expansion is an important problem in clinical natural language processing because abbreviations often occur in text notes in medical records, and expansions of these abbreviations are critical for downstream applications such as assistive diagnosis and insurance code review. Previous studies have treated abbreviation expansion as a special case of word sense disambiguation; however, abbreviation expansion is easier because we only need the character level expansion and not necessarily the full sense of the abbreviation. In particular, such character level expansions may naturally occur elsewhere in medical contexts. Accordingly, we consider two categories of methods for abbreviation expansion: (a) non-sense-based methods that use information solely at lexical levels using state-of-the-art language models, and (b) sense-based methods that also incorporate sense information, such as glosses, from knowledge bases, to simultaneously perform the two tasks of expansion and disambiguation of the abbreviation. We propose two language model based approaches, including a novel length-agnostic permutation language model, find non-sense methods to be more effective than sense-based methods, and achieve the state-of-the-art on three clinical datasets.

Keywords: Clinical NLP, abbreviation expansion

1. Introduction

Many tasks at the intersection of healthcare and machine learning, such as diagnosing patients, insurance coding for hospital procedures, and returning search results for medical terms, among others, often involves clinical notes expressed in natural language. However, clinical notes pose an idiosyncratic difficulty: abbreviations are commonly used
Improved Clinical Abbreviation Expansion via Non-Sense-Based Approaches

without reference to their meaning. For example, about 15% of PubMed queries include abbreviations (Islamaj Dogan et al., 2009), and about 15% of tokens in a clinical note are abbreviations (Xu et al., 2007). Thus, the task of determining the words of an abbreviation, called abbreviation expansion (AE), is helpful for any manual auditing of these notes, as well as for potential downstream systems to accomplish medical NLP tasks.

Abbreviation expansion poses two principal challenges: (a) abbreviations are not unique and may depend on the context, and (b) there is relatively little labeled data. As an illustration of the non-uniqueness of abbreviations, “AB” may refer to either “Abortion” or “Ankle-brachial.” Thus, practical approaches to abbreviation expansion must incorporate contextual information and function with little labeled data.

Most prior work on abbreviation expansion cast it as an instance of word sense disambiguation (WSD) and consequently have used models adapted from general WSD tasks for clinical abbreviation expansion. However, there is an important distinction between abbreviation expansion and WSD. Fundamentally, abbreviation expansion involves a character level expansion into a word or phrase, whereas WSD not only implicitly performs the character level expansion, but also assigns a unique “sense” from a given list of candidate senses. Like general vocabulary, medical terms have homonyms, so the word itself is not enough to determine the exact sense in many cases. For examples, the anatomic term pelvis may refer to hip bones or to regions of the kidney.

This difference has practical implications: WSD is a fundamentally more difficult problem since it not only implicitly performs a character level expansion as a first step but also an additional step of sense assignment. Moreover, the first step is potentially much simpler than the second since we have ample observations of character level expansions into words and phrases in the text corpus, whereas this is not true of word senses.

Thus, we propose two categories of approaches for abbreviation expansion. The first class is non-sense-based approaches, which use language models (LMs) that predict the probability of words in a specified location. Such an approach might not be applicable to WSD. The second class is sense-based approaches that use additional information on senses, such as the gloss, of various senses to perform WSD, which also results in an abbreviation expansion.

Additionally, we provide two language model-based approaches: one is based on the masked language model of BERT and the second is a novel adaptation of the permutation language model XLNet. Specifically, we call our adaptation the length-agnostic permutation language model, since it resolves an issue of exposing candidate expansion lengths.

Our non-sense-based approaches outperform the sense-based approaches, corroborating the intuition that abbreviation expansion may be easier than WSD. Moreover, non-sense based methods are easier to implement because they do not require additional knowledge. We examine three datasets: the MeSH index of MEDLINE dataset (MSH), and the Shared Annotated Resources/Conference and Labs of the Evaluation Forum dataset (ShARe/CLEF), and the University of Minnesota dataset (UMN). On these datasets, our method achieves state-of-the-art prediction accuracies.

The remainder of our paper is organized as follows. In Section 2, we provide details on related work. In Section 3, we describe our methods. In Section 4, we describe our data and experiments in detail, and we analyze the results in Section 5. Finally, we discuss directions for future research in Section 6. In the appendix, we provide de-
scriptions of additional methods, an experiment on partially-supervised and unsupervised training, and further details on our experimental setup.

2. Related Work

Here, we review three lines of related prior work: clinical abbreviation expansion, word sense disambiguation, and contextualized word embeddings.

2.1. Abbreviation Expansion

Clinical abbreviation expansion has been a topic of research for the past decade. Early work trained separate classifiers for each ambiguous abbreviation. Wu et al. (2015); Moon et al. (2012b); Sabbir et al. (2017) use Word2vec technique to learn word and concept embeddings and train SVM, Naive Bayes, decision trees, and k-nearest neighbors for each abbreviation. More recent works use contextualized word embeddings. Li et al. (2019); Jin et al. (2019) use BiLSTM to produce contextualized embeddings. Both use dedicated neural networks for each ambiguous abbreviation to predict the candidate expansions from the embedding. On the other hand, all of our approaches use a one-fits-all classifier that is applicable to unseen abbreviations and candidate expansions. One notable sense-based work is Pesaranghader et al. (2019), which however, it is a considerably more complex model: it uses pre-trained UMLS concept (CUI) sense embeddings, and given a context and candidate sense, uses a max pool over sense embeddings of each context word, computes cosine similarities to the candidate sense, and uses a BiLSTM over these cosine similarities, to extract contextualized features. It thus requires training a new embedding when a candidate is added. In contrast, we discuss a much simpler sense-based model architecture that performs comparably, and moreover can be applied to an unseen candidate expansion as long as the sense definition is available.

Some prior works augment the input features via knowledge bases (Moon et al., 2012b; Sabbir et al., 2017; Pesaranghader et al., 2019). Among our approaches, the sense-based methods use the sense definitions obtained from UMLS in fine-tuning and inference steps.

2.2. Word Sense Disambiguation

Word sense disambiguation is the task of determining the definition or sense of a given word. Prior work in WSD has attempted to use additional information, such as knowledge graphs or sense definitions, also known as gloss. Recent methods use gloss in neural network architectures (Luo et al., 2018a,b). These methods use non-contextualized and contextualized embeddings of the gloss to determine the word sense. GlossBERT (Huang et al., 2019) is a more recent example that concatenates the context and the gloss to determine the best sense.

2.3. Contextualized Word Embeddings

Contextualized word embeddings are increasingly common components of NLP tasks; recent important models include ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), and XLNet (Yang et al., 2019). We discuss the latter two in more detail.

BERT adapts the transformer encoder architecture (Vaswani et al., 2017) to generate contextualized embedding. BERT takes word embeddings, positional embeddings, and segment embeddings as input. BERT uses two unsupervised NLP tasks, the masked language model and next sentence prediction, to pre-train word embeddings. Variants of BERT are also made for specific settings by then training using domain-
specific data; NCBI-BERT (Peng et al., 2019) is a clinical version of BERT.

XLNet uses a generalized auto-regressive pre-training method that learns bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order. For both BERT and XLNet, \( \frac{1}{K} \) fraction of tokens are selected for prediction, where \( K = 6 \) in experiments.

3. Setup and Methods

In this section, we introduce our mathematical formulation of the problem and the methods we propose. Note that none of our methods involve training a separate classifier for each abbreviation. We visualize the methods with an examples in Figure 1.

3.1. Setup

In this subsection, we describe inputs, contexts, and embeddings, and we illustrate the notation in Example 1. Let \( V \) be a vocabulary, a set of subwords used by the tokenizer such as the BERT tokenizer. The vector \( x = [x_1, \ldots, x_N] \) in \( V^N \) is an input sentence, tokenized into a sequence of subwords. In the input sentence, the abbreviation spans from \( i_s \) to \( i_e \), since an abbreviation may be tokenized into multiple subwords. Thus, the abbreviation to be resolved is \( a = [x_{i_s}, \ldots, x_{i_e}] = x^{i_s:i_e} \).

Next, we define the context. The left context is \( c_{left} = [x_1, \ldots, x_{i_s-1}] \), and the right context is \( c_{right} = [x_{i_e+1}, \ldots, x_N] \). In words, the left and right contexts are the parts of the text occurring before and after the abbreviation. Thus, the context is \( c = [c_{left}, c_{right}] \).

Now, we discuss the expansions. Let \( K \) be the number of candidate expansions of the abbreviation. Then for any \( k \) in \( \{1, \ldots, K\} \), we denote the \( k \)-th candidate expansion by the vector \( y_k = [y_{k,1}, \ldots, y_{k,M_k}] \) in \( V^{M_k} \). Here, \( M_k \) is the length of the \( k \)-th candidate expansion and may vary with \( k \). The true expansion is denoted by \( y \), and this is a vector in \( V^M \) for some \( M \). The gloss of the \( k \)-th candidate expansion is denoted by \( g_k \) in \( V^{N_k} \). We also denote the random variables corresponding to the input, output and context via \( X, Y, C \) respectively.

3.2. Language Model Methods

In this subsection, we discuss our language model methods which are non-sense-based.

A functional characterization of the correct expansion of an abbreviation is that when the expansion is fit into the position of the abbreviation, it conveys the original meaning of the sentence correctly. This enables us to directly use a pre-trained language model to compute the probability of a candidate expansion \( y_k \), being in the place of the abbreviation, given the context \( c \):

\[
\log \Pr(Y = y_k | c).
\]

We consider two LM approaches: a masked LM approach and a permutation LM approach. A visualization is shown in Figure 3a.

**Masked Language Model.** BERT first replaces words with a number of masked tokens, denoted by \([\text{MASK}]\), and then uses the transformer encoder to predict the original tokens. During pre-training, the model maximizes the sum of the log probabilities of outputs at the positions of \([\text{MASK}]\) tokens, as-
○ Input text: ... She has been lightheaded and then presented to the **ER** today for ...
○ Candidate expansion: emergency room
○ Gloss of the candidate: Hospital department responsible for the administration ...

![Diagram](image)

Figure 1: Non-sense-based and sense-based methods for abbreviation expansion task. Methods (a) and (b) are non-sense-based methods, and method (c) is a sense-based method. Note that the classification layer in each method performs binary classification and is used for arbitrary abbreviations and candidate expansions.

The permutation language model of XLNet (Yang et al., 2019) maximizes the expected log-likelihood of a sequence with respect to all possible permutations of the factorization order. We can apply the permutation LM to solve the abbreviation expansion task by setting a factorization order so that the tokens of the candidate expansion appear from left to right. We can then obtain the log probability of a candidate expansion:

$$\log \Pr(Y = y_k | c; \theta) = \sum_{i=1}^{M_k} \log \Pr(Y_i = y_{k,i} | y_{k,<i}, c; \theta).$$

Here $y_{k,<i}$ is all tokens before the $i$-th token of the candidate.

This approach better models dependencies between positions within the candidate phrase than the masked LM does. However, the length of the candidate is still exposed to the XLNet via positional encodings. To hide the candidate length from the model, we design a new positional encoding scheme and a new pre-training task, the **Length-Agnostic Permutation Language Model**.
We assign two positions to each token in the sequence, **global position** and **local position**. In XLNet, a fraction of tokens are for prediction and others are not for prediction. We group a contiguous span of tokens for prediction as a single unit. For each token, its global position is the position of the unit that contains it, and its local position is its position within the unit. We apply relative positional encoding in XLNet with respect to these two positions and sum the embedding vectors. For abbreviation expansion, the model gets no information about the length of the candidate, which solves the issue. Figure 2 shows the input representation and the pre-training objective of the length-agnostic permutation LM.

The language model approaches can be performed in unsupervised or supervised way. Please see Appendix A.2.1 for the supervision used in fine-tuning.

### 3.3. Classification Based Methods

Another way of using a pre-trained language model to perform abbreviation expansion is to train a classifier that takes the contextualized embeddings of the candidates as input and outputs the probability of being a correct expansion. We call the non-sense-based and sense-based version of this classification based approach as **candidate classification** and **sentence classification**, respectively.

In candidate classification, the classifier takes the embeddings of candidate expansion contextualized by the context as input. In sentence classification, the classifier uses the joint sentence level representation of the context and the gloss. This idea was explored in GlossBERT (Huang et al., 2019), and we re-implement with biomedical fine-tuned BERT. Due to space limitations, we describe the details of these methods in Appendix A.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Abbr</th>
<th>Exp</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSH</td>
<td>203</td>
<td>410</td>
<td>37888</td>
</tr>
<tr>
<td>ShARe/CLEF</td>
<td>996</td>
<td>1058</td>
<td>7579</td>
</tr>
<tr>
<td>UMN</td>
<td>75</td>
<td>351</td>
<td>37500</td>
</tr>
</tbody>
</table>

Table 1: Summary statistics for the three datasets. For the ShARe/CLEF 2013 task 2 dataset, the number of unique abbreviations are counted after normalization.

### 4. Experiments

#### 4.1. Datasets

In this section, we describe three datasets on clinical abbreviation expansion that we use to evaluate our methods and our pre-processing procedure. Statistics of the three datasets are shown in Table 1.

**MSH.** The MSH WSD dataset (Jimeno-Yepes et al., 2011) is a biomedical WSD dataset collected from MEDLINE and manually annotated. The dataset consists of 37,888 instances of 203 ambiguous terms, 88 of which are abbreviations. Each term has 2 to 5 senses, each of which has at most 100 examples. The sense distribution is nearly uniform since most of the senses have 100 instances. All senses are represented as Unified Medical Language System (UMLS) Concept Unique Identifiers (CUI) in the dataset; so we can collect the names and the glosses of the senses from the UMLS.  

**ShARe/CLEF 2013 Task 2.** The ShARe/CLEF eHealth challenge 2013 (Mowery et al., 2016) has three clinical NLP tasks defined on clinical reports from MIMIC-II dataset, and task 2 is a normalization of acronyms and abbreviations. The training set and test set have annotations of 3,805 and 3,774 abbreviations extracted from 199 and 99 notes, respectively. Annotated abbreviations are labeled with CUIs.

---

2. For the UMN dataset, we do not use gloss since less than a half of the data are labeled with CUIs.
Improved Clinical Abbreviation Expansion via Non-Sense-Based Approaches

Figure 2: The length-agnostic permutation language model. Two positional encodings, global positions and local positions, are used to represent the locations of the tokens to be predicted while hiding the candidate length from the model. In the example, “the emergency room” and “acute abnormalities” are the tokens to be predicted.

expansions and their corresponding CUIs. For expansions for which the corresponding CUI does not have a UMLS gloss, we use the expansions themselves as the gloss.

The abbreviations in this dataset are more realistic in two respects. First, the abbreviations are not normalized, and thus there are many variants, reflecting different ways in which practitioners abbreviate the same phrase. So we did variation normalization in the same way as in Wu et al. (2013) to group the occurrences of abbreviations and to have better candidate expansions. Specifically, we removed special characters and lower cased all letters. For example, abbreviations “AFib”, “AFIB”, “afib”, and “A. fib” were converted into “afib” and the candidate expansions were shared among them.

Second, there are some test abbreviations that are unseen in the training set. For these test abbreviations, we made a set of candidate expansions (CUIs) using the table of acronyms and abbreviations (LRABR) in UMLS and MetaMap software (Aronson, 2001). Out of 549 unseen test examples, 382 examples are given candidate CUIs. We marked the remaining examples as incorrect during pre-processing. Besides, some examples are labeled GENERAL ENGLISH, where the abbreviation is a general term instead of a biomedical abbreviation. In such cases, we used the abbreviation itself as its expansion.

One key difference between the UMN dataset and the others is that labels are not represented as CUIs. Using sense-based approaches would require developing a knowledge-base; thus, we only present results for non-sense based approaches.

4.2. Training and Evaluation
We initialize BERT with the pre-trained NCBI-BERT model (12-layer, uncased) for fine-tuning. We truncate sentences so that their lengths do not exceed 128, the maximum sequence length of NCBI-BERT, and the lengths of the left and right context are as similar as possible. For the approaches using two segments as input, we truncate the second segment of each instance so that its length does not exceed half of the maximum

...
length. We fine-tune BERT and the classification layers with Adam optimizer (Kingma and Ba, 2014).

Since there is no biomedical version of XLNet available so far, we pre-train XLNet on biomedical corpora, PubMed abstracts and MIMIC-III clinical notes. We use the same pre-training settings as NCBI-BERT with only a few exceptions. Detailed settings are shown in the supplementary material. Pre-training XLNet takes 18 days with 4 NVIDIA Tesla V100 GPUs. We conduct the supervised and unsupervised permutation LM approach on the UMN and the MSH dataset, with the same sentence truncation and optimizer as in BERT.

Neither the MSH nor the UMN dataset has an official split for training and evaluation. We divide the datasets into 10 folds randomly and ran hyperparameter search for each round of 10-fold cross-validation separately. We use macro-averaged accuracy as the performance measure and report the average over 10 rounds. The details of the hyperparameter search are in the supplementary material.

For the ShARe/CLEF dataset, the train and test set are provided. Since the number of training examples is small, we train the models with the best parameters chosen from the other datasets. We train each model five times with random initialization and data shuffling, and we report the average of the test performances. Micro-averaged accuracy is used as the performance measure since the distributions of abbreviations is imbalanced.

On all the datasets, we consider two baselines: choosing a candidate expansion uniformly at random (“Random”) and choosing the most common expansion for a given abbreviation (“Majority”). These are most relevant for comparison against our unsupervised methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>48.74</td>
</tr>
<tr>
<td>Majority</td>
<td>54.48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-sense-based Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NB Jimeno-Yepes et al. (2011)</td>
<td>93.86</td>
</tr>
<tr>
<td>k-NN Sabbir et al. (2017)</td>
<td>94.34</td>
</tr>
<tr>
<td>SVM Jimeno-Yepes (2017)</td>
<td>95.97</td>
</tr>
<tr>
<td>Bi-LSTM Li et al. (2019)</td>
<td>96.71</td>
</tr>
<tr>
<td>Masked LM</td>
<td>95.89</td>
</tr>
<tr>
<td>Permutation LM</td>
<td><strong>96.83</strong></td>
</tr>
<tr>
<td>Candidate Classification</td>
<td>95.94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sense-based Methods (Gloss)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BLSTM Pesaranghader et al. (2019)</td>
<td>96.82</td>
</tr>
<tr>
<td>Sentence Classification</td>
<td>95.60</td>
</tr>
</tbody>
</table>

Table 2: Accuracy results for the MSH dataset. Our methods perform nearly as well as the state-of-the-art with less specialized approaches.

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>40.24</td>
</tr>
<tr>
<td>Majority</td>
<td>68.55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-sense-based Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM + Profile Wu et al. (2013)</td>
<td>71.9</td>
</tr>
<tr>
<td>Masked LM</td>
<td>76.56</td>
</tr>
<tr>
<td>Permutation LM</td>
<td><strong>77.97</strong></td>
</tr>
<tr>
<td>Candidate Classification</td>
<td>76.14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sense-based Methods (Gloss)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence Classification</td>
<td>75.95</td>
</tr>
</tbody>
</table>

Table 3: Results for ShARe/CLEF 2013 Task 2, seen abbreviations only. Our supervised and partially-supervised methods outperform pre-existing methods.

5. Results and Analysis

We report our results and those of prior work for the MSH, ShARe/CLEF, and UMN datasets in Table 2, Table 3, and Table 4, respectively. Recall that our key question is whether non-sense based methods are preferable. The most important aspect of this question is whether non-sense approaches can provide comparable or superior perfor-
Table 4: Accuracy results for the UMN dataset. Our supervised masked LM performs best and achieves the state-of-the-art.

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>25.76</td>
</tr>
<tr>
<td>Majority</td>
<td>68.89</td>
</tr>
<tr>
<td>Non-sense-based Methods</td>
<td></td>
</tr>
<tr>
<td>SVM Moon et al. (2012b)</td>
<td>92.75</td>
</tr>
<tr>
<td>SVM Wu et al. (2015)</td>
<td>95.79</td>
</tr>
<tr>
<td>Masked LM</td>
<td>98.39</td>
</tr>
<tr>
<td>Permutation LM</td>
<td>98.28</td>
</tr>
<tr>
<td>Candidate Classification</td>
<td>98.34</td>
</tr>
</tbody>
</table>

Performance in a supervised setting, and our experimental results confirm that non-sense approaches are superior. A secondary question is how much our models perform on unseen abbreviations. Since our model uses a one-fits-all classifier, it performs much better on unseen test abbreviations than previous approaches. We detail the results of these experiments.

**Sense vs Non-Sense in Supervised Settings.** For the MSH dataset, our permutation LM outperforms both our basic sense-based approaches and other recent approaches. Our permutation LM outperforms the sense-based approaches by over 1%, which is considerable given the already high accuracy and is comparable to recent approaches. Specifically, the permutation LM narrowly outperforms Bi-LSTM and BLSTM, which were the previous best non-sense and sense based approaches respectively (Li et al., 2019; Pesaranghader et al., 2019). For the ShARE/CLEF dataset, our permutation LM approach outperforms our sense-based approaches by over 2%. Finally, we note that our results with non-sense based approaches constitute the new state-of-the-art on both datasets.

We can further illustrate the utility of non-sense based approaches by considering the UMN dataset. As noted earlier, the UMN dataset has no gloss; so our sense-based methods are not used. However, our non-sense based approaches achieve over 98% accuracy, establishing the new state-of-the-art.

**Performance on Unseen Abbreviations**

To test our model in a difficult setting, we analyze the performance on unseen test abbreviations of the ShARE/CLEF dataset and compare with Wu et al. (2013). As described in 4.1, we use the LRABR table of UMLS to get the expansion candidates for the unseen test abbreviations, and the best candidate is chosen by the trained model. In Wu et al. (2013), unseen abbreviations are disambiguated by a WSD method built on a private clinical corpus and UMLS Terminology Services API. Note that the LRABR table is also used to build the WSD method.

Table 5 shows the micro accuracy on the seen and unseen test abbreviations of the ShARE/CLEF dataset. Note that the number of unseen test examples can be different because of the abbreviation preprocessing. For the test examples with unseen abbreviations, our method achieves accuracy almost three times higher than the competitor. We attribute this to the universality of our model that can be applicable to all abbreviations given the list of candidate expansions. Note that only 382 out of the 549 unseen test examples are covered by LRABR and more exhaustive knowledge base would lead higher accuracy.

**6. Discussion**

Our experiments show that non-sense based methods can obtain state-of-the-art performance in clinical abbreviation expansion. Additionally, a key feature is that non-sense methods are easier to use than sense-based approaches since they do not require a knowledge base. However, a number of problems still need to be solved for abbreviation expansion to be performed more effectively in
Table 5: Accuracies on unseen abbreviations for ShARe/CLEF 2013 Task 2. The numbers in parentheses are the number correct and the number of samples. Seen/Unseen: test examples with seen/un-seen abbreviation. Note that the number of unseen test examples can be different because of the abbreviation preprocessing.

<table>
<thead>
<tr>
<th>Method</th>
<th>Seen</th>
<th>Unseen</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wu et al. (2013)</td>
<td>82.53</td>
<td>11.20</td>
<td>71.89</td>
</tr>
<tr>
<td>Masked LM</td>
<td>84.37</td>
<td>30.42</td>
<td>76.52</td>
</tr>
</tbody>
</table>

practice, regardless of the method. Two important directions are better methods for choosing candidate expansions and for handling abbreviation variants. Such advances may greatly improve performance on downstream tasks.

References


Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. Incorporating glosses into neural word sense disambiguation. In Proceedings of the 56th Annual...
Appendix A. Additional Methods

In this section, we describe several variants of non-sense-based and sense-based methods for abbreviation expansion. These methods either perform binary classification with the contextualized embeddings of the candidates or compare the embeddings of the abbreviation with that of the candidates. Please see 3 for visualization.
○ **Input text**: ... She has been lightheaded and then presented to the **ER** today for ...
○ **Candidate expansion**: emergency room
○ **Gloss of the candidate**: Hospital department responsible for the administration ...

- **Figure 3**: Non-sense-based and sense-based methods for abbreviation expansion task. Methods (a), (b) and (d) are non-sense-based methods, and method (c) and (e) are sense-based methods. Note that the classification layer in each method performs binary classification and is used for arbitrary abbreviations and candidate expansions.
A.1. Contextualized Embeddings

In this subsection, we define contextualized word embeddings. A contextualized embedding with dimension $L$ of the vector $x \in V^N$ is a vector

$$e = [e_1, \cdots, e_N] \in \mathbb{R}^{N \times L}.$$  

In all experiments except the permutation language model, we use BERT to obtain contextualized word embeddings. Using BERT: $V^N \to \mathbb{R}^{N \times L}$ to stand for the mapping given by BERT, we write $e = BERT(x)$.

In our methods, we use a feature vector representation of the abbreviation and the candidate expansions that we obtain by averaging the corresponding contextualized word embeddings. Given the input sentence, the feature vector of the abbreviation $v_{abbr}$ is

$$v_{abbr} = \text{avg}(e_{i_{abbr}}),$$

where $e = BERT(x)$ and $\text{avg}(v_1, \cdots, v_n) = \frac{1}{n} \sum_{i=1}^{n} v_i$. To get the feature vector of $k$-th candidate expansion $v_{cand,k}$, we replace the abbreviation with the candidate and repeat the procedure:

$$e_{cand,k} = BERT([c_{left}, y_k, c_{right}]),$$  

$$v_{cand,k} = \text{avg}(e_{i_{cand,k}}^{i_{cand,k}} + M_k - 1).$$

In sense-based methods, we use the sentence level feature vector $v_{sent}$ of BERT. This is obtained by using the contextualized vector of the first [CLS] token $v_{sent} = e_1$ or taking the average of all the contextualized vectors $v_{sent} = \text{avg}(e)$.

A.2. Non-sense-based Methods

Candidate Classification Approach.

Our main embedding approach is to train a classifier that determines if a candidate is consistent with the context. Specifically, we take the BERT embeddings of an expansion candidate $v_{cand,k}$, which captures the coherence of the candidate with the context of the abbreviation, and perform binary classification:

$$\text{output}_k = f(v_{cand,k}) + \gamma,$$

where $f$ is a feed-forward neural network and $\gamma$ is a scaling parameter. We train the network to output the probability of each candidate being correct independently.

In the test phase, given $K$ candidate expansions, we choose the candidate with the highest log probability:

$$\hat{k} = \arg \max_k \text{output}_k.$$

Figure 3b shows an example of the candidate classification approach.

Additionally, we can compare the contextualized embedding of the abbreviation and each of the expansion candidates. Since this is a minor architecture variant that produces similar results, we discuss this further in the Appendix.

A.2.1. Supervised, Partially-supervised, and Unsupervised Approaches

Additionally, our non-sense based LM approaches can be used in an unsupervised or partially supervised manner. In the former, LMs are trained only on generic text corpora, as is typical in large-scale NLP, and then used directly for abbreviation expansion. In the latter, partially supervised methods train a classifier on top of the LM, which is far more computationally efficient.

Since language models are already pre-trained with a large corpus, an interesting observation is that the LM approaches explained above can perform abbreviation expansion without further fine-tuning. However, they can also be fine-tuned so that correct expansions can have high probability.
To fine-tune the BERT or XLNet model, we use margin loss of the log probability:

\[
\mathcal{L}(x, y, y_{\text{neg}}; \theta) = \max\{\beta + \log \Pr(Y = y_{\text{neg}}|c_{\text{mask}}; \theta)/|y_{\text{neg}}|^{\alpha} - \log \Pr(Y = y|c_{\text{mask}}; \theta)/|y|^{\alpha}, 0\},
\]

where \(y_{\text{neg}}\) is an incorrect expansion, \(\beta\) is the hyper-parameter for margin, and \(\alpha\) is the hyper-parameter for length normalization to reduce the bias towards short sequences, following common practice in neural machine translation Graves (2012).

A final alternative is a partially supervised approach for the embedding methods in which we fix the language model and train the classifier only. The key benefit is that this speeds up the training since we do not require the gradient update of BERT and the embeddings only need to be computed once.

While the main focus of our paper is on the non-sense versus sense-based dichotomy, we also give the results for this interesting ablation study.

### A.3. Sense-based Methods

So far we have discussed approaches to expand abbreviations using non-sense-based methods. Alternatively, we can perform WSD to solve the problem of expanding abbreviations as a by-product. This allows us to use external information about the candidate expansions; here, we focus on knowledge-based approaches where we have access to the gloss, or definition, of each expansion.

**Sentence Classification Approach.** With glosses, we can use BERT to internally compute the coherence between the input text and the gloss of each candidate expansion. This is the key idea of GlossBERT Huang et al. (2019).

We feed the concatenation of the input text and the gloss of a candidate into BERT.

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>48.74</td>
</tr>
<tr>
<td>Majority</td>
<td>54.48</td>
</tr>
<tr>
<td>NB Jimeno-Yepes et al. (2011)</td>
<td>93.86</td>
</tr>
<tr>
<td>k-NN Sabbir et al. (2017)</td>
<td>94.34</td>
</tr>
<tr>
<td>SVM Jimeno-Yepes (2017)</td>
<td>95.97</td>
</tr>
<tr>
<td>Bi-LSTM Li et al. (2019)</td>
<td>96.71</td>
</tr>
<tr>
<td>BLSTM Pesaranghader et al. (2019)</td>
<td>96.82</td>
</tr>
</tbody>
</table>

Table 6: Accuracy results for the MSH dataset. Our methods perform nearly as well as the state-of-the-art with less specialized approaches.

As with the candidate classification approach in A.2, we do binary classification, but the input to the classifier is the sentence level feature \(e_{\text{sent}}\) which is computed from \(e_{\text{concat}, k} = \text{BERT}([x, g_k])\). The method is visualized in Figure 3c.

Additionally, we can compare the contextualized embedding of each of the expansion candidates with its gloss; this is a minor architecture variant that we discuss in the following.

### A.4. Embedding Comparison

**Architectures**

We also tried another variant of the non-sense candidate classification method and of
<table>
<thead>
<tr>
<th>Method</th>
<th>Micro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>40.24</td>
</tr>
<tr>
<td>Majority</td>
<td>68.55</td>
</tr>
<tr>
<td>SVM + Profile Wu et al. (2013)</td>
<td>71.9</td>
</tr>
</tbody>
</table>

**Unsupervised Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masked LM</td>
<td>69.26</td>
</tr>
<tr>
<td>Permutation LM</td>
<td>71.83</td>
</tr>
</tbody>
</table>

**Supervised Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masked LM</td>
<td>76.56</td>
</tr>
<tr>
<td>Permutation LM</td>
<td><strong>77.97</strong></td>
</tr>
<tr>
<td>Candidate Classification</td>
<td>76.14</td>
</tr>
<tr>
<td>Abbreviation Comparison</td>
<td>76.53</td>
</tr>
</tbody>
</table>

**Partially-supervised Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate Classification</td>
<td>75.77</td>
</tr>
<tr>
<td>Abbreviation Comparison</td>
<td>76.47</td>
</tr>
</tbody>
</table>

**Sense-based Methods (Gloss)**

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence Classification</td>
<td>75.95</td>
</tr>
<tr>
<td>Gloss Comparison</td>
<td>74.51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>25.76</td>
</tr>
<tr>
<td>Majority</td>
<td>68.89</td>
</tr>
<tr>
<td>SVM Moon et al. (2012b)</td>
<td>92.75</td>
</tr>
<tr>
<td>SVM Wu et al. (2015)</td>
<td>95.79</td>
</tr>
</tbody>
</table>

**Unsupervised Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masked LM</td>
<td>68.53</td>
</tr>
<tr>
<td>Permutation LM</td>
<td>78.29</td>
</tr>
</tbody>
</table>

**Supervised Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masked LM</td>
<td><strong>98.39</strong></td>
</tr>
<tr>
<td>Permutation LM</td>
<td>98.28</td>
</tr>
<tr>
<td>Candidate Classification</td>
<td>98.34</td>
</tr>
<tr>
<td>Abbreviation Comparison</td>
<td>98.38</td>
</tr>
</tbody>
</table>

**Partially-supervised Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidate Classification</td>
<td>98.02</td>
</tr>
<tr>
<td>Abbreviation Comparison</td>
<td>98.15</td>
</tr>
</tbody>
</table>

Table 8: Accuracy results for the UMN dataset. Our supervised masked LM performs best and achieves the state-of-the-art.

This approach can be viewed as Siamese neural network since we use same network twice to get embeddings. For training the sentence classification approach and the abbreviation comparison approach, we use binary cross-entropy loss. A visual representation is given in Figure 3d.

**Gloss Comparison Method**

Along the same lines as the abbreviation comparison method, we can compare the feature vector of the abbreviation $v_{abbr}$ and the sentence-level feature $v_{sent}$ of the gloss of each candidate obtained from $e_{gloss,k} = BERT(g_k)$. To compute the probability, we concatenate these vectors and feed them into a feed-forward neural network. Figure 3e shows a diagram of the method.

**Appendix B. Partially-supervised Training**

In the embedding methods, we can fix the language model and train the classifier only. This speeds up the training since we do not

Table 9 contains the results for supervised and unsupervised language models and supervised and partially supervised candidate classification approaches, which utilize LMs. First, we observe that the supervised methods perform the best, although the partially-supervised candidate classification approach is not much worse. Thus, fine-tuning the language model during training gives better performance, but the performance reduction with a lack of such fine-tuning may be tolerable in some cases because training is faster. Second, our unsupervised methods have much worse performance than the partially-supervised or supervised methods, but our unsupervised permutation LM method greatly outperforms the baselines. The improvements is the least in the ShARe/CLEF, about 3%, but majority baseline is fairly difficult given the class imbalance. In fact, our unsupervised permutation LM performs nearly as well as the previous state-of-the-art method Wu et al. (2013), with a micro accuracy difference of 0.07%. Considering that unsupervised LM methods do not use any training data for the task at hand, such improvements are outstanding.

Appendix D. Transfer Learning
Since our approaches train a single model that can be used for all abbreviations, we can perform evaluation on a dataset different than training. This transfer learning procedure is prevalent in WSD literature, where training is done with a large dataset and evaluation is done with smaller ones. To study the efficacy of transfer learning and also compare against unsupervised and supervised learning, we chose the masked LM approach. We trained the model with the MSH dataset and evaluated the model on the UMN dataset. The experiment was repeated 5 times and the average of the macro-averaged test accuracies is reported.

Table 10 shows the results of the transfer learning experiment. Training the model with a different dataset improves the performance modestly. However, the improvement is not as substantial as the supervised
Table 10: Transfer learning result for the UMN dataset. Transfer learning is helpful, but not as effective as supervised learning.

<table>
<thead>
<tr>
<th>Method</th>
<th>Fine-tune</th>
<th>Macro Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsupervised</td>
<td>×</td>
<td>68.53</td>
</tr>
<tr>
<td>Transfer</td>
<td>MSH</td>
<td>75.68</td>
</tr>
<tr>
<td>Supervised</td>
<td>UMN</td>
<td>98.39</td>
</tr>
</tbody>
</table>

Appendix E. Pre-training

Biomedical XLNet

To ensure that the performance of our different approaches is comparable, we used the same pre-training dataset as NCBI-BERT Peng et al. (2019). We preprocessed the training data by lower-casing applying Moses tokenizer\(^5\) and lower-casing all characters. Table 11 shows the statistics of our text corpora for pre-training our Length-Agnostic XLNet (the number of words is counted after tokenization).

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Words</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>PubMed abstract</td>
<td>4,470M</td>
<td>Biomedical</td>
</tr>
<tr>
<td>MIMIC-III</td>
<td>556M</td>
<td>Clinical</td>
</tr>
</tbody>
</table>

Table 11: Corpora for pre-training.

Our XLNet Yang et al. (2019) implementation is based on the source code released by the authors\(^6\). Following them, we built sentencepiece encoding on the corpora. Because XLNet has more parameters given the same number of layers and hidden dimensions as NCBI-BERT, we reduced the hidden dimensions of feedforward layers from 3,072 to 2,048 so that our XLNet is of the similar size as NCBI-BERT. As a result, the total number of parameters of our XLNet is 116M, which is similar to NCBI-BERT. Table 12 shows all the hyperparameters for pre-training.

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># layers</td>
<td>12</td>
</tr>
<tr>
<td>Hidden size</td>
<td>768</td>
</tr>
<tr>
<td>Feedforward size</td>
<td>2048</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>49152</td>
</tr>
<tr>
<td># attention heads</td>
<td>12</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>Attention dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1e-4</td>
</tr>
<tr>
<td>Adam settings</td>
<td>(0.9, 0.98, 1e-6)</td>
</tr>
<tr>
<td>Batch size (tokens)</td>
<td>131072</td>
</tr>
<tr>
<td>Sequence length</td>
<td>128</td>
</tr>
<tr>
<td># predicting tokens</td>
<td>20</td>
</tr>
<tr>
<td>Training steps</td>
<td>500000</td>
</tr>
<tr>
<td>Warmup steps</td>
<td>10000</td>
</tr>
</tbody>
</table>

Table 12: Hyperparameter settings for pre-training Length-Agnostic XLNet.

Appendix F. Hyper-parameter Search

For the UMN and the MSH dataset, there is no split of the dataset for training and evaluation. We divided the datasets into 10 folds by splitting the instances per abbreviation uniformly at random over the folds. For each round of 10-fold cross-validation, we set aside 1 fold (10%) for testing and used the remaining folds (90%) for hyperparameter search. Training on 8 folds and evaluating on 1 fold, we chose the hyperparameter with the best macro-averaged accuracy. After we found the best hyperparameter, we re-ran training

\(^5\) https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl
\(^6\) https://github.com/zihangdai/xlnet
with all the 9 training folds. We ran this process of hyperparameter search and testing for all 10 rounds, and report the average of the macro-averaged accuracy as the final performance of the method.

Below are the lists of the hyper-parameters searched during 10-fold cross-validation for the UMN and the MSH dataset with the hyper-parameters chosen to train the models on ShARe/CLEF 2013 Task 2 (SC13T2) dataset. For each of the Siamese network approaches, we tried cross-entropy loss and margin loss. We chose the best hyper-parameters from all configurations of both losses.

### Table 13: Supervised masked LM approach

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Margin β</td>
<td>{0.1, 0.5, 2.0, 5.0}</td>
</tr>
<tr>
<td>LengthNorm α</td>
<td>1.0</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>5</td>
</tr>
</tbody>
</table>

### Table 14: Supervised permutation LM Approach

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Margin β</td>
<td>{0.1, 0.5, 2.0, 5.0}</td>
</tr>
<tr>
<td>LengthNorm α</td>
<td>0.6</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{3e-5, 4e-5}</td>
</tr>
<tr>
<td>Adam settings</td>
<td>(0.9, 0.98, 1e-6)</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
</tr>
<tr>
<td>Train epochs</td>
<td>5</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Table 15: Supervised candidate embedding approach

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>{2, 3, 4}</td>
</tr>
<tr>
<td>Hidden size</td>
<td>768</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>3 / 5</td>
</tr>
</tbody>
</table>

### Table 16: Supervised Siamese network approach (cross-entropy loss)

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>{2, 3, 4}</td>
</tr>
<tr>
<td>Hidden size</td>
<td>768</td>
</tr>
<tr>
<td>Loss function</td>
<td>xent</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>3 / 5</td>
</tr>
</tbody>
</table>

### Table 17: Supervised Siamese network approach (margin loss)

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>3</td>
</tr>
<tr>
<td>Hidden size</td>
<td>768</td>
</tr>
<tr>
<td>Loss function</td>
<td>margin</td>
</tr>
<tr>
<td>Margin β</td>
<td>{0.1, 0.2, 0.5, 1.0}</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>4 / 5</td>
</tr>
</tbody>
</table>

### Table 18: Partially-supervised candidate embedding approach

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>{2, 4, 6}</td>
</tr>
<tr>
<td>Hidden size</td>
<td>3840</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
</tr>
<tr>
<td>Train epochs</td>
<td>5 / 10</td>
</tr>
</tbody>
</table>
## Hyper-param UMN / MSH SC13T2

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>{2, 4, 6}</td>
</tr>
<tr>
<td>Hidden size</td>
<td>3840</td>
</tr>
<tr>
<td>Loss function</td>
<td>xent</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>5 / 8</td>
</tr>
</tbody>
</table>

Table 19: Partially-supervised Siamese network approach (cross-entropy loss)

## Hyper-param UMN / MSH SC13T2

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>4</td>
</tr>
<tr>
<td>Hidden size</td>
<td>3840</td>
</tr>
<tr>
<td>Loss function</td>
<td>margin</td>
</tr>
<tr>
<td>Margin $\beta$</td>
<td>{0.1, 0.2, 0.5, 1.0}</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>5 / 8</td>
</tr>
</tbody>
</table>

Table 20: Partially-supervised Siamese network approach (margin loss)

## Hyper-param UMN / MSH SC13T2

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Layers</td>
<td>{3, 4}</td>
</tr>
<tr>
<td>Hidden size</td>
<td>768</td>
</tr>
<tr>
<td>Loss function</td>
<td>xent</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 22: Semantic Siamese network approach (cross-entropy loss)

## Hyper-param UMN / MSH SC13T2

<table>
<thead>
<tr>
<th>Hyper-param</th>
<th>UMN / MSH SC13T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weak supervision</td>
<td>{Y, N}</td>
</tr>
<tr>
<td>Sentence feature</td>
<td>{[CLS], avg}</td>
</tr>
<tr>
<td># Layers</td>
<td>3</td>
</tr>
<tr>
<td>Hidden size</td>
<td>768</td>
</tr>
<tr>
<td>Loss function</td>
<td>margin</td>
</tr>
<tr>
<td>Margin $\beta$</td>
<td>{0.2, 0.5}</td>
</tr>
<tr>
<td>Learning rate</td>
<td>{2e-5, 5e-5}</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
</tr>
<tr>
<td>Train epochs</td>
<td>5 / 10</td>
</tr>
</tbody>
</table>

Table 23: Semantic Siamese network approach (margin loss)