Improved Clinical Abbreviation Expansion via Non-Sense-Based Approaches

Juyong Kim JUYONGK@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213

Linyuan Gong GONGLINYUAN@HOTMAIL.COM
Department of EECS, University of California, Berkeley, CA 94720

Justin Khim JKHIM@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213

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

Pradeep Ravikumar PRADEEPR@CS.CMU.EDU
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. [1]

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

1. Code available at https://github.com/dalgu90/abbr-exp-ml4h/

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, 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 $\mathbf{x} = [x_1; \quad ; x_N]$ 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 $\mathbf{a} = [x_{i_s}; \quad ; x_{i_e}] = \mathbf{x}^{i_s:i_e}$.

Next, we define the context. The left context is $\mathbf{c}_{\text{left}} = [x_1; \quad ; x_{i_s \ 1}]$, and the right context is $\mathbf{c}_{\text{right}} = [x_{i_e+1}; \quad ; x_N]$. In words, the left and right contexts are the parts of the text occurring before and after the abbreviation. Thus, the context is $\mathbf{c} = [\mathbf{c}_{\text{left}}; \mathbf{c}_{\text{right}}]$.

Now, we discuss the expansions. Let K be the number of candidate expansions of the abbreviation. Then for any k in $f1; : : : ; Kg$, we denote the k-th candidate expansion by the vector $\mathbf{y}_k = [y_{k;1}; \quad ; y_{k;M_k}]$ 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 $\mathbf{y}$, and this is a vector in V^M for some M. The gloss of the k-th candidate expansion is denoted by $\mathbf{g}_k$ in $V^{N'_k}$. We also denote the random variables corresponding to the input, output and context via $\mathbf{X}; \mathbf{Y}; \mathbf{C}$ respectively.

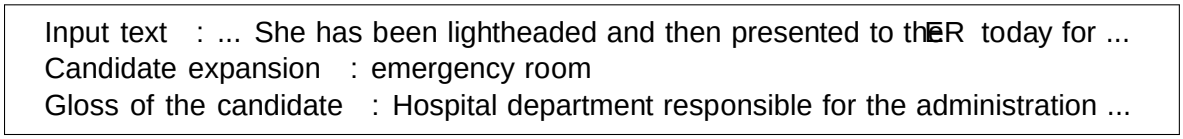
Example 1.
Input Text: "... as well as continued bleeding from the cervical os. This is consistent with an incomplete **AB**. The patient presents now for a suction D&C ..."
Expansion Candidates:
1. abortion 2. ankle-brachial
Vectors:
Input $\mathbf{x}$ = [:::; as; well; :::; **AB**; :::; &; C; :::]
Cand $\mathbf{y}_1$ = [abortion]
Cand $\mathbf{y}_2$ = [ankle; -; bra; ##chia; ##l]
True expansion $\mathbf{y}$ = [abortion]

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 $\mathbf{y}_k$, being in the place of the abbreviation, given the context $\mathbf{c}$: $\log \Pr(\mathbf{Y} = \mathbf{y}_k j \mathbf{c})$. 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 `[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 `[MASK]` tokens, as-



(a) Language Model Methods (b) Candidate Classification (c) Sentence Classification

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.

suming that the outputs at the masked positions are independent.

Thus, we can apply the pre-trained masked LM to abbreviation expansion, by replacing the abbreviation with [MASK]tokens and computing the log probability of each candidate expansion. We choose the expansion that gives the highest log probability, which we denote by $\hat{y}$, and define as:

$$\hat{y} = \arg\max_{y_k} \log \Pr(Y = y_k | c_{mask}; \theta)$$
$$= \arg\max_{y_k} \sum_{i=1}^{M_k} \log \Pr(Y_i = y_{k;i} | c_{mask}; \theta);$$

where c_{mask} is the context c with the [MASK] tokens, and θ is the set of BERT parameters.

Permutation Language Model. The masked LM approach has two drawbacks: (a) It fails to model dependencies between positions within the candidate. (b) It exposes the length of the candidate to the model since it replaces an abbreviation with the same number of [MASK]tokens as each candidate, making it difficult to compare candidates of different lengths.

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 ne-tuning.

3.3. Classi cation Based Methods

Another way of using a pre-trained language model to perform abbreviation expansion is to train a classi er 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 classi cation based approach ascandidate classi cation and sentence classi cation , respectively.

In candidate classi cation, the classi er takes the embeddings of candidate expansion contextualized by the context as input. In sentence classi cation, the classi er 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 ne-tuned BERT. Due to space limitations, we describe the details of these methods in Appendix A.

Dataset	Abbr	Exp	Examples
MSH	203	410	37888
ShARe/CLEF	996	1058	7579
UMN	75	351	37500

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 preprocessing 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 Uni ed Medical Language System (UMLS) Concept Unique Identi ers (CUI) in the dataset; so we can collect the names and the glosses of the senses from the UMLS.[2]

ShARe/CLEF 2013 Task 2. The ShARe/CLEF eHealth challenge 2013 (Mowery et al., 2016) has three clinical NLP tasks de ned 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

2. For the UMN dataset, we do not use gloss since less than a half of the data are labeled with CUIs.

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[3]. 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, re ecting di erent 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", \a b", and \A. b" were converted into \a b" 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 for fair comparison with baselines.

UMN. The UMN dataset (Moon et al., 2012a) is collected from clinical notes from hospitals in the University of Minnesota a liated Fairview Health Services. Researchers manually annotated 500 instances for each of 75 abbreviations. Each abbreviation has an imbalanced distribution of expansions.

Some examples are annotated as MISTAKE, NAME, or UNSURE SENSE. We discarded them 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 di erence between the UMN dataset and the others is that labels are not represented as CUIs. [4] 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 ne-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

3. Some abbreviations are labeled as #CUI-less , and we treat all of these as a single expansion.

4. There is a sense inventory in the dataset, but it has CUIs for only 189 of the 337 labels.

length. We ne-tune BERT and the classi - cation layers with Adam optimizer (Kingma and Ba, 2014).

Since there is no biomedical version of XL-Net 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 super-vised and unsupervised permutation LM ap-proach on the UMN and the MSH dataset, with the same sentence truncation and opti-mizer as in BERT.

Neither the MSH nor the UMN dataset has an o cial split for training and evalu-ation. We divide the datasets into 10 folds randomly and ran hyperparameter search for each round of 10-fold cross-validation sepa-rately. We use macro-averaged accuracy as the performance measure and report the av-erage over 10 rounds. The details of the hy-perparameter search are in the supplemen-tary material.

For the ShARe/CLEF dataset, the train and test set are provided. Since the num-ber of training examples is small, we train the models with the best parameters cho-sen from the other datasets. We train each model ve times with random initialization and data shu ing, and we report the aver-age of the test performances. Micro-averaged accuracy is used as the performance measure since the distributions of abbreviations is im-balanced.

On all the datasets, we consider two base-lines: choosing a candidate expansion uni-formly at random (\Random") and choos-ing the most common expansion for a given abbreviation (\Majority"). These are most relevant for comparison against our unsuper-vised methods.

Method	Macro Acc(%)
Random	48.74
Majority	54.48
Non-sense-based Methods	
NB Jimeno-Yepes et al. (2011)	93.86
k-NN Sabbir et al. (2017)	94.34
SVM Jimeno-Yepes (2017)	95.97
Bi-LSTM Li et al. (2019)	96.71
Masked LM	95.89
Permutation LM	96.83
Candidate Classi cation	95.94
Sense-based Methods (Gloss)	
BLSTM Pesaranghader et al. (2019)	96.82
Sentence Classi cation	95.60

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

Method	Micro Acc(%)
Random	40.24
Majority	68.55
Non-sense-based Methods	
SVM + Pro le Wu et al. (2013)	71.9
Masked LM	76.56
Permutation LM	77.97
Candidate Classi cation	76.14
Sense-based Methods (Gloss)	
Sentence Classi cation	75.95

Table 3: Results for ShARe/CLEF 2013 Task 2, seen abbreviations only. Our super-vised and partially-supervised methods out-perform 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 prefer-able. The most important aspect of this question is whether non-sense approaches can provide comparable or superior perfor-

Method	Macro Acc(%)
Random	25.76
Majority	68.89
Non-sense-based Methods	
SVM Moon et al. (2012b)	92.75
SVM Wu et al. (2015)	95.79
Masked LM	98.39
Permutation LM	98.28
Candidate Classi cation	98.34

Table 4: Accuracy results for the UMN dataset. Our supervised masked LM performs best and achieves the state-of-the-art.

mance in a supervised setting, and our experimental results con rm that non-sense approaches are superior. A secondary question is how much our models perform on unseen abbreviations. Since our model uses a one-ts-all classi er, 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. Speci cally, 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 di cult 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 di erent 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 e ectively in

	Seen	Unseen	Total
Wu et al. (2013)	82.53 (2650/3211)	11.20 (63/563)	71.89 (2713/3774)
Masked LM	84.37 (2721/3225)	30.42 (167/549)	76.52 (2888/3774)

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 di erent because of the abbreviation preprocessing.

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

Alan R Aronson. E ective mapping of biomedical text to the umls metathesaurus: the metamap program. In Proceedings of the AMIA Symposium, page 17. American Medical Informatics Association, 2001.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv.org, October 2018.

Alex Graves. Sequence transduction with recurrent neural networks, 2012.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge. arXiv.org, August 2019.

Rezarta Islamaj Dogan, G Craig Murray, Aurelie Neveol, and Zhiyong Lu. Understanding pubmed® user search behavior through log analysis. Database, 2009, 2009.

Antonio Jimeno-Yepes. Word embeddings and recurrent neural networks based on Long-Short Term Memory nodes in supervised biomedical word sense disambiguation. Journal of Biomedical Informatics, 73:137{147, 2017.

Antonio Jimeno-Yepes, Bridget T McInnes, and Alan R Aronson. Exploiting MeSH indexing in MEDLINE to generate a data set for word sense disambiguation. BMC Bioinformatics, 12(1):S1, 2011.

Qiao Jin, Jinling Liu, and Xinghua Lu. Deep contextualized biomedical abbreviation expansion. arXiv preprint arXiv:1906.03360, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Zhi Li, Fan Yang, and Yaoru Luo. Context Embedding Based on Bi-LSTM in Semi-Supervised Biomedical Word Sense Disambiguation. IEEE Access, 2019.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1402{1411, 2018a.

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

Meeting of the Association for Computational Inguistics (Long Papers), pages 2473{2482, 2018b.

Sungrim Moon, Serguei Pakhomov, and Genevieve Melton. Clinical abbreviation sense inventory, 2012a. URL http://purl.umn.edu/137703 .

Sungrim Moon, Serguei Pakhomov, and Genevieve B. Melton. Automated Disambiguation of Acronyms and Abbreviations in Clinical Texts - Window and Training Size Considerations.AMIA , 2012b.

Danielle L Mowery, Brett R South, Lee Christensen, Jianwei Leng, Laura-Maria Peltonen, Sanna Salanterä, Hanna Suominen, David Martinez, Sumithra Velupillai, Nœmie Elhadad, et al. Normalizing acronyms and abbreviations to aid patient understanding of clinical texts: Share/clef ehealth challenge 2013, task 2Journal of Biomedical Semantics, 7(1):43, 2016.

Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets. In ACL Workshop, June 2019.

Ahmad Pesaranghader, Stan Matwin, Marina Sokolova, and Ali Pesaranghader. deepBioWSD - e ective deep neural word sense disambiguation of biomedical text data. JAMIA , 26(5):438{446, 2019.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. arXiv preprint arXiv:1802.05365 , 2018.

AKM Sabbir, Antonio Jimeno-Yepes, and Ramakanth Kavuluru. Knowledge-based biomedical word sense disambiguation with neural concept embeddings. In 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE) , pages 163{170, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998{6008, 2017.

Yonghui Wu, Buzhou Tang, Min Jiang, Sungrim Moon, Joshua C Denny, and Hua Xu. Clinical Acronym/Abbreviation Normalization using a Hybrid Approach. CLEF , 2013.

Yonghui Wu, Jun Xu, Yaoyun Zhang, and Hua Xu. Clinical Abbreviation Disambiguation Using Neural Word Embeddings. In BioNLP@IJCNLP , pages 171{176, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics.

Hua Xu, Peter D Stetson, and Carol Friedman. A study of abbreviations in clinical notes. In AMIA annual symposium proceedings, volume 2007, page 821. American Medical Informatics Association, 2007.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.

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 classi cation 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 ...

(a) Language Model Methods (b) Candidate Classification (c) Sentence Classification

(d) Abbreviation comparison method (e) Gloss comparison method

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

$$e = [e_1; \quad ; e_N] \in R^{N \quad L}:$$

In all experiments except the permutation language model, we use BERT to obtain contextualized word embeddings. Using $BERT : V^N \to R^{N \quad L}$ 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

$$v_{abbr} = avg(e^{i_s:i_e})$$

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

$$e_{cand;k} = BERT([c_{left}; y_k; c_{right}]);$$
$$v_{cand;k} = avg(e^{i_s:i_s+M_k \quad 1}_{cand;k}):$$

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 $v_{sent} = avg(e)$.

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:

$$output_k = f(v_{cand;k}) + \quad ;$$

where f is a feed-forward neural network and $\quad$ 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:

$$\hat{k} = \arg\max_k output_k:$$

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 pretrained 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 ne-tune the BERT or XLNet model, we use margin loss of the log probability:

$$L(x; y; y_{neg};) = \max f + \log \Pr(Y = y_{neg} j c_{mask};) = j y_{neg} j - \log \Pr(Y = y j c_{mask};) = j y j ; 0 g;$$

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

A nal alternative is a partially supervised approach for the embedding methods in which we x the language model and train the classi er only. The key bene t 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 de nition, of each expansion.

Sentence Classi cation 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.

Method	Macro Acc(%)
Random	48.74
Majority	54.48
NB Jimeno-Yepes et al. (2011)	93.86
k-NN Sabbir et al. (2017)	94.34
SVM Jimeno-Yepes (2017)	95.97
Bi-LSTM Li et al. (2019)	96.71
BLSTM Pesaranghader et al. (2019)	96.82
Unsupervised Methods	
Masked LM	70.41
Permutation LM	77.84
Supervised Methods	
Masked LM	95.89
Permutation LM	96.83
Candidate Classi cation	95.94
Abbreviation Comparison	95.90
Partially-supervised Methods	
Candidate Classi cation	94.07
Abbreviation Comparison	94.02
Sense-based Methods (Gloss)	
Sentence Classi cation	95.60
Gloss Comparison	95.20

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 classi cation approach in A.2, we do binary classi cation, but the input to the classi er is the sentence level feature v_{sent} which is computed from $e_{concat;k} = \text{BERT}([x; g_k])$. 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 classi cation method and of

Method	Micro Acc(%)
Random	40.24
Majority	68.55
SVM + Pro le Wu et al. (2013)	71.9
Unsupervised Methods	
Masked LM	69.26
Permutation LM	71.83
Supervised Methods	
Masked LM	76.56
Permutation LM	77.97
Candidate Classi cation	76.14
Abbreviation Comparison	76.53
Partially-supervised Methods	
Candidate Classi cation	75.77
Abbreviation Comparison	76.47
Sense-based Methods (Gloss)	
Sentence Classi cation	75.95
Gloss Comparison	74.51

Table 7: Results for ShARe/CLEF 2013 Task 2, seen abbreviations only. Our supervised and partially-supervised methods outperform pre-existing methods.

Method	Macro Acc(%)
Random	25.76
Majority	68.89
SVM Moon et al. (2012b)	92.75
SVM Wu et al. (2015)	95.79
Unsupervised Methods	
Masked LM	68.53
Permutation LM	78.29
Supervised Methods	
Masked LM	98.39
Permutation LM	98.28
Candidate Classi cation	98.34
Abbreviation Comparison	98.38
Partially-supervised Methods	
Candidate Classi cation	98.02
Abbreviation Comparison	98.15

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

the sense-based sentence classi cation approach. Together, we call these embedding comparison architectures, since they compare the embedding of the candidate with that of the abbreviation or the gloss.

Abbreviation Comparison Method For this method, we simply compare the contextualized embedding of the abbreviation and each of the expansion candidates. In supervised setting, we train the BERT model so that the contextualized embeddings of both the abbreviation and the candidate expansion are used to classify whether the candidate ts the context.

For the classi er, we concatenate the feature vectors (v_{abbr}, $v_{cand;k}$) and feed it into feed-forward neural network that outputs the log-probability:

$$\log \Pr(Y = y_k | x;) + = f(v_{abbr}; v_{cand;k}):$$

This approach can be viewed as Siamese neural network since we use same network twice to get embeddings. For training the sentence classi cation 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 x the language model and train the classi er only. This speeds up the training since we do not

Method	MSH	SC13	UMN
Majority	54.58	68.55	68.89
U Masked LM	70.41	69.26	68.53
U Permutation LM	77.84	71.83	78.29
S Masked LM	95.89	76.56	98.39
S Permutation LM	96.83	77.97	98.28
PS Cand Classi cation	94.07	75.77	98.02
S Cand Classi cation	95.94	76.14	98.34

Table 9: Accuracy results of the ablation study of ne-tuning. Our unsupervised methods outperforms simple baselines, and our partially-supervised methods with perform slightly worse than supervised methdos. U : unsupervised, S: supervised, PS: partially-supervised, SC13: ShARe/CLEF 2013 Task 2.

require the gradient update of BERT and the embeddings only need to be computed once.

In supervised methods, we have the size of hidden layers same as the dimension of the word embeddings. For the partially-supervised methods, we increase the hidden layer size by a factor of 5 to increase the expressiveness of the classi er.

We give the full results, including the partially-supervised methods and the embedding comparison methods, in Table 6, Table 7, and Table 8 for the MSH, ShARe/CLEF, and UMN datasets respectively.

Appendix C. Comparison of Supervised, Partially-Supervised, and Unsupervised Methods.

Table 9 contains the results for supervised and unsupervised language models and supervised and partially supervised candidate classi cation approaches, which utilize LMs. First, we observe that the supervised methods perform the best, although the partially-supervised candidate classi cation approach is not much worse. Thus, ne-tuning the language model during training gives better performance, but the performance reduction with a lack of such ne-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 dif- cult 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 di erence 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 di erent 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 e cacy 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 di erent dataset improves the performance modestly. However, the improvement is not as substantial as the supervised

Method (Masked LM)	Fine-tune	Macro Acc(%)
Unsupervised		68.53
Transfer	MSH	75.68
Supervised	UMN	98.39

Table 10: Transfer learning result for the UMN dataset. Transfer learning is helpful, but not as e ective as supervised learning.

method. We attribute this to the small overlap of between the two datasets; only 8 out of 75 abbreviations in the UMN data are in the MSH data.

Appendix E. Pre-training Biomedical XLNet

To ensure that the performance of our di erent 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).

Corpus	Words	Domain
PubMed abstract	4,470M	Biomedical
MIMIC-III	556M	Clinical

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.

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

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.

Hyper-param	**UMN / MSH**	**SC13T2**
Margin	{0.1, 0.5, 2.0, 5.0}	2.0
LengthNorm	1.0	1.0
Learning rate	{2e-5, 5e-5}	2e-5
Batch size	16	16
Train epochs	5	5

Table 13: Supervised masked LM approach

Hyper-param	**UMN / MSH**
Margin	{0.1, 0.5, 2.0, 5.0}
LengthNorm	0.6
Learning rate	{3e-5, 4e-5}
Adam settings	(0.9, 0.98, 1e-6)
Batch size	32
Train epochs	5
Weight decay	0.1

Table 14: Supervised permutation LM Approach

Hyper-param	**UMN / MSH**	**SC13T2**
# Layers	{2, 3, 4}	3
Hidden size	768	768
Learning rate	{2e-5, 5e-5}	2e-5
Batch size	32	32
Train epochs	5 / 10	10

Table 15: Supervised candidate embedding approach

Hyper-param	**UMN / MSH**	**SC13T2**
# Layers	{2, 3, 4}	3
Hidden size	768	768
Loss function	xent	xent
Learning rate	{2e-5, 5e-5}	2e-5
Batch size	16	16
Train epochs	3 / 5	5

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

Hyper-param	**UMN / MSH**	**SC13T2**
# Layers	3	Not chosen
Hidden size	768	
Loss function	margin	
Margin	{0.1, 0.2, 0.5, 1.0}	
Learning rate	{2e-5, 5e-5}	
Batch size	16	
Train epochs	4 / 5	

Table 17: Supervised Siamese network approach (margin loss)

Hyper-param	**UMN / MSH**	**SC13T2**
# Layers	{2, 4, 6}	4
Hidden size	3840	3840
Learning rate	{2e-5, 5e-5}	5e-5
Batch size	32	32
Train epochs	5 / 10	10

Table 18: Partially-supervised candidate embedding approach

Hyper-param	UMN / MSH	SC13T2
# Layers	{2, 4, 6}	Not chosen
Hidden size	3840	
Loss function	xent	
Learning rate	{2e-5, 5e-5}	
Batch size	16	
Train epochs	5 / 8	

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

Hyper-param	UMN / MSH	SC13T2
# Layers	4	4
Hidden size	3840	3840
Loss function	margin	margin
Margin	{0.1, 0.2, 0.5, 1.0}	0.2
Learning rate	{2e-5, 5e-5}	5e-5
Batch size	16	16
Train epochs	5 / 8	8

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

Hyper-param	UMN / MSH	SC13T2
Weak supervision	{Y, N}	N
Sentence feature	{[CLS], avg}	avg
# Layers	3	3
Hidden size	768	768
Learning rate	{2e-5, 5e-5}	2e-5
Batch size	32	32
Train epochs	10	10

Table 21: Semantic sentence embedding approach

Hyper-param	UMN / MSH	SC13T2
Sentence feature	{[CLS], avg}	Not chosen
# Layers	{3, 4}	
Hidden size	768	
Loss function	xent	
Learning rate	{2e-5, 5e-5}	
Batch size	16	
Train epochs	5	

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

Hyper-param	UMN / MSH	SC13T2
Sentence feature	{[CLS], avg}	avg
# Layers	3	3
Hidden size	768	768
Loss function	margin	margin
Margin	{0.2, 0.5}	0.2
Learning rate	{2e-5, 5e-5}	5e-5
Batch size	16	16
Train epochs	5	10

Table 23: Semantic Siamese network approach (margin loss)