Encoder based Attention Mechanism for Word Sense Disambiguation

Lalchand Pandia *
lcpandia@gmail.com

Harish Karnick*
Inter-disciplinary Program in Cognitive Science
Indian Institute of Technology, Kanpur
hk@iitk.ac.in

Abstract

In this work we present a BERT(Devlin et al., 2018) based approach for Word Sense Disambiguation(WSD). Unlike previous works which use feature based language representations for WSD, we fine tune an existing BERT based representation model to construct our model.

We have evaluated our approach on the standard line, hard, serve (**?**) and interest datasets(Bruce and Wiebe, 1994) and obtained new state-of-the-art micro-averaged F1 scores.

To handle the extreme sample size imbalance in the dataset we have used a weighted version of the loss function while constructing the model and found that even senses with very few training examples can be correctly classified. We obtained better macro-averaged F1 scores(Manning et al., 2008) for interest, hard and line datasets as compared to the standard unweighted loss function.

1 Introduction

The problem of Word Sense Disambiguation (WSD) is the task of finding what a word means when it is used in a particular context. It has been a central problem in the field of computational linguistics, and in particular for Machine Translation (MT) and Information Retrieval (IR) applications. Previous approaches to WSD relied heavily on hand-crafted features and ignored order of words in a sentence. Recent approaches like ELMo(Peters et al., 2018) use a biLSTM combined with Language Modelling(LM) on all-words WSD tasks. These approaches are feature-based and do not pay much attention to fine-tuning pre-trained language representations.
The contributions of this paper are as follows:

- We demonstrate how a fine-tuning approach based on a BERT pre-trained model can improve performance on WSD tasks.
- We show that introducing class specific weights per sense can further help to disambiguate senses where very few examples of the sense are present in the dataset.

2 Related Work

Earlier approaches to WSD relied on hand-crafted features and applied classical learning algorithms((Ng and Lee, 1996);(**?**);(Bruce and Wiebe, 1994). These methods ignored order of words in a sentence.
Recent developments in word embeddings have inspired several work which uses word embeddings to correctly disambiguate senses(Rothe and Schütze, 2015). But these approaches also rely heavily on hand-crafted features. Other works tries to alleviate this problem by learning context dependent representations(Melamud et al., 2016)and achieve satisfactory results.
Other recent work on Language Modelling(LM) learn word representations of the word to disambiguate as function of the entire input sentence(Peters et al., 2018)using a bidirectional LM and use it in WSD.

3 Model

A sentence S is a sequence of words $(w_1, w_2, ..., w_N)$ and we can formulate WSD (word sense disambiguation) as the task of assigning the appropriate sense(s) to all or some of the words in the sentence, i.e., to find a mapping A from words to senses, such that $A(i) \subseteq Senses_D(w_i)$, where $Senses_D(w_i)$ is the set of senses encoded in a dictionary D for word w_i, and $A(i)$ is that subset of senses of w_i which are appropriate in the context T. Usually,

* Work performed while at CSE, Indian Institute of Technology, Kanpur

T consists of the sentence(s) within which the word w_i, whose senses are to be determined, is embedded.

There are two variants of WSD:

- Lexical Sample WSD: The system is required to disambiguate a restricted set of target words usually occurring one per sentence.
- All-words WSD: The system is expected to disambiguate all open-class words in a text (i.e., nouns, verbs, adjectives, and adverbs).

We will be focusing on **Lexical Sample WSD**. Specifically given a sentence $(w_1, w_2, ..., w_N)$ and a target word w_T, i.e. the word to disambiguate, we wish to compute a probability distribution over the possible senses corresponding to that word. Let $|V|$ denote the size of the vocabulary.
In this work, we use BERT (Bidirectional Encoder Representations from Transformers(Devlin et al., 2018)), a multi-layer bidirectional Transformer for WSD. Specifically, we try to encode the context of a target word x_t into a hidden state which is multiplied with the output layer to obtain the logits. The logits are then fed into the Softmax function to obtain probabilities for each sense of the target word.

3.1 Input Representation

The input representation is same as adopted in (Devlin et al., 2018)

- The first token of every sentence is always the special token [CLS]. We separate each sentence with a special token [SEP].
- We use WordPiece embeddings(Wu et al., 2016)
- We use learned positional embeddings with supported sequence lengths upto 512 tokens.

3.2 Model Architecture

The basic model architecture is described below.

3.2.1 Input Layer

The input sentence is converted to input-ids. Each token is then fed to *Word-piece embeddings* and element-wise addition of *position-wise embeddings* of every token is done.

3.2.2 BERT layer

We use the BERT pre-trained model which is a multi-layer bidirectional Transformer encoder. It consists of:

- L, number of Transformer blocks
- H, hidden size
- A, number of self-attention heads

The feed forward size is set to 4H. We specifically used $BERT_{BASE}$: L=12, H=768, A=12.

3.2.3 Softmax Layer

The final hidden state corresponding to [CLS] is used as the aggregate sequence representation. This layer consists of $U \in R^{S \times H}$, S is the number of senses for the target word. $s(T) = Uh_T + b$

$y(T) = softmax(s(T)) = \frac{e^{s(T)}}{\Sigma_{i \in S} e^{s(i)}}$ The final sense of the word is predicted by taking argmax over all the probabilities and the corresponding sense index is predicted. For fine tuning, all of the parameters of BERT and U are jointly optimized using Adam optimizer(Kingma and Ba, 2015)

3.3 Loss function

While we found that the fine-tuned procedure as described in the BERT paper works well for balanced datasets, we also saw that simply using the unweighted cross-entropy function was not giving enough gains when dealing with highly imbalanced datasets.
We used a weighted version of the cross-entropy loss function to fine tune our model.

$$L = -\Sigma_{i \in I} \Sigma_{j \in S(w_i)} W^C_{i,j} t_{i,j} log y_j(i) \quad (1)$$

S denotes the total number of senses for word w_T. $W^C_{i,j}$ denotes the weight of j^{th} sense in training example i
The weights W^C are treated as model hyperparameters for different datasets.

4 Experiments and Results

Maximum sequence length	128
Batch size	32
Learning rate	2e-5
Number of epochs	3

Table 1: Fine-tuned hyperparameters

The hyper-parameters used in our experiments are summarized in Table 1 We initially tried with sequence lengths of 100 and 200. But sequence length of 128 worked better. We also tried by setting number of epochs to 1 but 3 worked better.
To explore the effect of weighted loss function, we varied class weights and found different class weights for different datasets. Initially we used weights that were inversely proportional to the number of examples of each sense in the dataset. But this hurt performance on the better represented senses. This lead us to moderate the weights on the sparsely represented senses downwards. We think a more principled way to choose weights based on the softmax probability during training is likely to be better. However, this is experiment is yet to be done.

Sense	class weights
interest6	1.89
interest5	4.69
interest1	6.32
interest4	14.09
interest3	39.46
interest2	69

Table 2: class weights for interest dataset

Sense	class weights
hard1	1.26
hard2	8.52
hard3	10.87

Table 3: class weights for hard dataset

Sense	class weights
cord	11.26
division	10.4
formation	12.59
phone	9.69
product	1.8
text	10.3

Table 4: class weights for line dataset

Sense	class weights
serve10	2.42
serve6	10.23
serve2	5.04
serve12	3.44

Table 5: class weights for serve dataset

The class-weights used for different datasets are summarized in Table 2, Table 3, Table 4, Table 5

We report 4-fold cross validated results on all the datasets.

Dataset	unweighted	weighted
Interest	**95.85**	95.47
Line	96.83	**97.43**
Serve	**98.85**	98.72
Hard	**96.3**	96.28

Table 6: Micro averaged F1 scores with unweighted and weighted cross-entropy loss function

Dataset	unweighted	weighted
Interest	76.78	**84.79**
Line	95.4	**96.21**
Serve	**98.64**	98.43
Hard	92.07	**92.13**

Table 7: Macro averaged F1 scores with unweighted and weighted cross-entropy loss function

The previous best result on *interest* dataset was achieved by Naive Bayes Ensemble(Pedersen, 2000) and it achieved a micro F1-score of 89. The same approach also achieved best result on *line* dataset with micro F1-score of 88.
The previous best micro F1 scores on *serve* and *hard* dataset were 78 and 83 respectively and were achieved by Naive Bayes classifier with bag-of-words context feature set(**?**).
As can be seen from Table 6 that we have surpassed the previous state-of-the-art results on all the 4 datasets by significant margins.
The macro-averaged F1 scores with the unweighted cross-entropy loss and weighted cross-entropy loss functions are given in Table 7. The results suggest that our weighted approach is better suited to handle class imbalance by paying attention to all the senses by enhancing the loss on sparsely present senses.

5 Conclusion

We presented a fine-tuning based approach to WSD that uses a weighted loss function to disambiguate highly imbalanced datasets. We demonstrated the effect of weights with macro-averaged F1 scores. Currently the weights are fixed hyper-parameters but we believe that a more principled way of learning these weights can greatly benefit our approach.

Our approach has significantly improved the best performance till now for both micro and macro averaged F1 scores on the standard Lexical Sample WSD datasets.

References

Rebecca Bruce and Janyce Wiebe. 1994. Word-sense disambiguation using decomposable models. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 139–146, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 40–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ted Pedersen. 2000. A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. *CoRR*, cs.CL/0005006.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.