Lotus at SemEval-2021 Task 2: Combination of BERT and Paraphrasing for English Word Sense Disambiguation

Niloofar Ranjbar
Department of Computer Engineering
Jam School of Engineering
Persian Gulf University
nranjbar@pgu.ac.ir

Hossein Zeinali
Department of Computer Engineering
Amirkabir University of Technology
hzeinali@aut.ac.ir

Abstract

In this paper, we describe our proposed methods for the multilingual word-in-Context disambiguation task in SemEval-2021. In this task, systems should determine whether a word that occurs in two different sentences is used with the same meaning or not. We proposed several methods using a pre-trained BERT model. In two of them, we paraphrased sentences and add them as input to the BERT, and in one of them, we used WordNet to add some extra lexical information. We evaluated our proposed methods on test data in SemEval-2021 task 2.

1 Introduction

Measuring semantic similarity is a task that is important in many applications such as summarization, plagiarism detection, etc. To measure the semantic similarity between two words or sentences, the words' meaning in their contexts should be understood. In "Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation" of SemEval 2021 (Martelli et al., 2021), systems should determine whether a word that occurs in two different sentences is used with the same meaning or not. The considered languages in the Word-in-Context Disambiguation task of the evaluation are Arabic, Chinese, English, French, and Russian (Martelli et al., 2021). Due to our internal limitation, we only evaluated the proposed methods on the English dataset.

In most cases, humans can understand the correct meaning of each word by paying attention to the context of that word. This was the main reason for proposing the attention mechanism for machine translation (Vaswani et al., 2017). Because attention plays an important role in BERT topology (Devlin et al., 2019), we believe it should work for the WSD task. However, in some cases, the two sentences may be about the same subject, while the target word has a different meaning. In these cases, find out that whether the target word has the same meaning in both sentences or not is difficult. To overcome this challenge, we need to add some extra information about the contexts to the input of the BERT.

We proposed four main methods for this task. In the first method, we use the BERT model as a language representation model in which the inputs are the two sentences that come in a row and are separated by a "[SEP]" token. Furthermore, the target word is surrounded by a "[TGT]" token in both sentences. Then, the first output of the last layer is used as a classifier to determine whether the word has the same meaning in the two sentences or not. In the second one, we added some information about the target word from the WordNet[1] to the end of each input and used the same strategy as the first method. In the third and fourth methods, we used the paraphrases of sentences as additional inputs to the BERT.

The remainder of the paper is organized as follows: In section 2, we provide a short literature review of the WSD task and mention some related works that used BERT for this task. After that, in section 3, we introduce our methods in consequent sections: the first method that only uses the BERT model is explained in Section 3.1. The second method that uses the BERT model and some additional information from WordNet is explained in Section 3.2. The third and fourth methods that use the BERT model and add paraphrased sentences as an additional input to the model are explained in Section 3.3. Then in Section 4, the used datasets in our experiments are specified. Finally, the results and discussion are presented in Section 5.

[1] https://wordnet.princeton.edu/

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 724–729
Bangkok, Thailand (online), August 5–6, 2021. ©2021 Association for Computational Linguistics

2 Background

2.1 Task Setup

The dataset provided by MCL-WiC[2] addresses both multilingual and cross-lingual aspects, covers all part-of-speeches, and also a high number of domains and genres. The task of the challenge is a binary classification where systems should specify whether the target word has the same meaning in both sentences (T for true) or different meaning (F for false). The sentences can be from the same language (multilingual dataset) or across different languages (cross-lingual dataset) (Martelli et al., 2021).

In the following you can find an example of sentence pair in English from the multilingual dataset[3]:

- In that context of coordination and integration, Bolivia holds a key play in any process of infrastructure development.
- A musical play on the same subject was also staged in Kathmandu for three days.

In this example, the target word "play" should be tagged as F (False) since it is used in two distinct meanings.

We evaluated our methods on the English part of the multilingual dataset. We used the MCL-WiC training set, containing 8000 sentence pairs in English as the training data, and the MCL-WiC development set containing 1000 sentence pairs in English as the validation set for our system.

2.2 Related works

There are several published works that used BERT for WSD task (Wiedemann et al., 2019; Du et al., 2019; Yap et al., 2020; Guo et al., 2020) and also some other papers that combined BERT and WordNet for this task (Levine et al., 2020; Peters et al., 2019).

For example, (Levine et al., 2020) proposed a method named SenseBERT in which the model is pre-trained to predict the masked words and their WordNet super-senses. They proposed a mechanism in which they pay attention to the words at the sense level. Therefore, they achieved a lexical-semantic level language model. SenseBERT achieved 72.1 % accuracy on the "Word in Context" task which is a state-of-the-art result on this task.

(Peters et al., 2019) proposed a general method in which multiple knowledge bases are embedded into large-scale models. By doing that and using structured knowledge, they enhanced their representations. First, they use an entity linker to retrieve relevant entity embeddings for each knowledge base (KB). After that, they use word-to-entity attention to update contextual word representations. They trained entity linkers and self-supervised language modeling together in an end-to-end multi-task setting in which a large amount of raw text is combined with a small amount of entity linking supervision. By merging WordNet and a subset of Wikipedia into BERT, KnowBert shows improvement in several tasks such as word sense disambiguation. This method achieved a good result on the "Word in Context" task with 70.9 % accuracy.

"GlossBERT" is another work that was proposed in (Huang et al., 2019). The contexts and glosses of the target words were put together and considered as inputs to the BERT. Three BERT-based models were proposed for WSD. The SemCor3.0 training corpus was used to fine-tune the pre-trained BERT model and finally, the models were evaluated on several English WSD benchmark datasets. The experimental results show that the proposed method achieved new state-of-the-art results on the WSD task.

3 System Overview

In this section, we explain our proposed methods. The first method that relies only on the BERT model is explained in Section 3.1. This method is further improved in 3.2 by adding some extra information extracted from the WordNet to the BERT's input. Finally, in section 3.3 we explain an augmentation method by generating the paraphrases of both sentences and add them as an extra input to the BERT.

3.1 BERT Method

In this method, we fine-tune a BERT model using the TensorFlow-models PIP package as explained in fine-tuning-BERT Tutorial. To do this, we use the pre-trained BERT encoder (large-uncased-BERT) from TensorFlow Hub. As mentioned in the tutorial, to fine-tune a pre-trained model, exactly the same tokenization, vocabulary, and index mapping with the model should be used. So we

[2]Multilingual and Cross-lingual Word-in-Context Disambiguation task

[3]https://github.com/SapienzaNLP/mcl-wic/blob/master/SemEval-2021_MCL-WiC_all-datasets.zip

rebuild and use the tokenizer that was used by the base model.

Next, we preprocess the provided data and prepare it as the input of the BERT model. For every pair of sentences first, we add a "[TGT]" token before and after the target word in both sentences. After that, we add a "[SEP]" (Separator) token at the end of both sentences. Then we encode them separately by the tokenizer. The tokenizer itself changes all words to lowercase and separates the unknown words. We also encode and add a "[CLS]" token at the first position of each example to be able to do a classification task. Examples are constructed by concatenation of two sentences. Note that we had to add "[TGT]" as a token into the tokenizer vocabulary. Here is an example of a pair of sentences:

> *[CLS] In that context of coordination and integration, Bolivia holds a key [TGT] play [TGT] in any process of infrastructure development. [SEP] A musical [TGT] play [TGT] on the same subject was also staged in Kathmandu for three days.[SEP]*

The output of the tokenization process is named input word ids. For the above example, we will have:

> *[101 1999 2008 6123 1997 12016 1998 8346 1010 11645 4324 1037 3145 1 2377 1 1999 2151 2832 1997 6502 2458 1012 102 1037 3315 1 2377 1 2006 1996 2168 3395 2001 2036 9813 1999 28045 2005 2093 2420 1012 102]*

Note that input word ids vectors should be padded with zero token ids to make them equal length.

We also add two vectors as extra inputs to the BERT model. One of them is called input mask, in which, for every non-padding token, we put "1", and for every padding token, we put "0". Another one is called input type ids in which, for each token of the first sentence, second sentence, and padding tokens, we put "1", "2", and "0" respectively. Thus, the input vectors equivalent to the above sentence pairs (without padding tokens) are as follows:

> *input_word_ids: [101 1999 2008 6123 1997 12016 1998 8346 1010 11645 4324 1037 3145 1 2377 1 1999 2151 2832 1997 6502 2458 1012 102 1037 3315 1 2377 1 2006 1996 2168 3395 2001 2036 9813 1999 28045 2005 2093 2420 1012 102]*
>
> *input_mask: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]*
>
> *input_type_ids: [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]*

After preparing the input, we define the BERT classifier with the following configuration:

> *'attention_probs_dropout_prob': 0.1,*
> *'hidden_act': 'gelu',*
> *'hidden_dropout_prob': 0.1,*
> *'hidden_size': 768,*
> *'initializer_range': 0.02,*
> *'intermediate_size': 3072,*
> *'max_position_embeddings': 512,*
> *'num_attention_heads': 12,*
> *'num_hidden_layers': 12,*
> *'type_vocab_size': 2,*
> *'vocab_size': 30522*

As mentioned in fine-tuning-BERT Tutorial : *"BERT adopts the Adam optimizer with weight decay. It uses a learning rate that firstly warms up from 0 and then decreases to 0."* We choose the batch size= 32 and train the BERT classifier with maximum epochs=10. We also use an early stopping call back, which is based on the validation set loss, and make the training process stopped, when the validation loss increase in two steps.

3.2 BERT plus WordNet method

As mentioned before, in some WSD cases using just the target word context can not be helpful, so we proposed a new method in which some information from the WordNet is added to the end of each sentence.

In this method, for every synset of the target word, we define a score that indicates how much that synset is related to the target word in this context. To calculate this score, we used the following equation:

$$
\begin{aligned}
&path\_dist \\
&= \sum_{w \in context} \frac{1}{position\_dist(w, target\_w)} \\
&\times min_{syn_2 \in syns(w)}(spd(syn_1, syn_2))) \qquad (1)
\end{aligned}
$$

where $position\_dist$ indicates how much the target word is far from the word "w" in the context. So, the farther the word is, the effect of that to calculate the path dist is less. "$syns(w)$" are all synsets of the word "w" in the context. "syn_1" is the synset of the target word that we want to calculate its path distance. Note that, before calculating the score, we remove all stop words, punctuations, and numbers from the context of the target word. spd is a function that is already implemented in NLTK package[4]. It returns the distance of the shortest path linking the two synsets (if one exists). For each synset, all the ancestor nodes and their distances are recorded and compared. The ancestor node common to both synsets that can be reached with the minimum number of traversals is used. If no ancestor nodes are common, -1 is returned. If a node is compared with itself, 0 is returned[5].

As can be seen, if the $path\_dist$ increases, the synset is less related to the target word in this context. Therefore, we sort the synsets based on how relevant they are to the target word and use their Lexname[6] (which is called supersense in (Levine et al., 2020) to represent them. After that, we add all the sorted Lexnames at the end of each sentence separated by a "[LX]" token.

Here is an example for a pair of sentences:

> **sentence1:** in that context of coordination and integration , bolivia holds a key [TGT] play [TGT] in any process of infrastructure development .[LX] time [LX] state [LX] act [LX] communication [LX] event [LX] attribute [SEP]
>
> **sentence2:** a musical [TGT] play [TGT] on the same subject was also staged in kathmandu for three days .[LX] time [LX] communication [LX] act [LX] state [LX] attribute [SEP]

Finally, we define $input\_word\_ids$, $input\_mask$, and $input\_type\_ids$ same as in Section 3.1 and train the model defined in that section.

[4] https://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html

[5] https://docs.huihoo.com/nltk/0.9.5/api/nltk.wordnet.synset.Synset-class.html

[6] More information about Lexname can be found at https://wordnet.princeton.edu/documentation/lexnames5wn

3.3 BERT plus Paraphrase Method

As paraphrasing changes the structure and some words of a sentence while the meaning remains the same, it can be helpful to add some extra information as input to the BERT encoder. By adding this information, we help the model to see other meanings of the context words.

We used Machine Translation to generate paraphrases of the sentences. To do that, we use Microsoft Translator Text API [7] to translate all of the sentences from English to Spanish and then translate them back from Spanish to English. As this API has reliable results on translating from and to the Spanish language, we choose it as the intermediate language. However, other languages such as French and Germany can be used as well.

Based on this idea, two methods are proposed in this section:

3.3.1 Using generated paraphrases

In the first method, we add the paraphrased sentences word ids, mask, and type_ids as additional inputs to the BERT encoder. Thus, the input to the BERT encoder is constructed from 6 parts instead of 3 parts defined in Section 3.1. Normally, three input vectors are used for the BERT. In this case, we add three additional input vectors which are made with the paraphrase of sentences instead of sentences themselves. In our opinion, the best way of combining this new data is to concatenate paraphrases to the main sentences, but in this work, due to our internal limitations, we could not do this, and instead, we extended the three inputs to six. In practice, the input is the sum of the embedded vectors of the six vectors. We do not use the "[TGT]" token before and after the target word in the generated paraphrase because it may be changed.

3.3.2 Using just the different words

As the generated paraphrases are not very different from the original sentences most of the time, we propose another method in which we just find the words that are changed in the generated paraphrase and then concatenate them with the original words in the original sentence with a "[S]" token. To do that, we should find the original word of each newly generated word by finding the most similar word from the main sentence to the generated word. We used the *fastText* [8] pre-trained model in English to

[7] https://www.microsoft.com/en-us/translator/business/translator-api/

[8] https://fasttext.cc

Methods	run1	run2	run3	run4	run5	Average	Std
BERT Method (3.1)	84.4	84.9	84.2	85.8	83.8	**84.62**	0.688
BERT + WordNet (3.2)	84.5	84.7	84.3	85.2	85.2	**84.78**	0.366
BERT + Paraphrase Sents (3.3.1)	85.3	85.6	85.6	86.9	85.5	**85.78**	0.571
BERT + Paraphrase Words (3.3.2)	84.1	86.3	85.6	85.4	82.4	**84.76**	1.378

Table 1: Results of the proposed methods in 5 different runs. Std in the last column indicate *Standard Deviation*.

find the most similar word, and then consider it as the paraphrased word.

4 Experimental Setup

As mentioned before, we used the MCL-WiC training set, containing 8000 pair sentences in English as the training data, and the MCL-WiC development set containing 1000 pair sentences in English as the validation data for our system. Similarly, the MCL-WiC test set containing 1000 pair sentences in English was used as the test data. The results displayed in Table 1 are based on the test set. We use the validation set only for finding the best number of fine-tuning iterations. It is worth mentioning that in test phase of the third method, we first paraphrase the sentences of the test set and make them ready for the model input as we did for the training and validation sets. All reported results are based on accuracy that is the main criteria for the challenge.

5 Results

Due to the random initialization of classifier layer weights, different results were achieved from different runs. So, we decided to run each experiment 5 times and report the average and standard deviation of model accuracy.

Table 1 shows the obtained results from different methods based on the accuracy percentage. It is obvious that adding generated sentence level paraphrases proposed in Section 3.3.1 improves the average accuracy compared to the BERT method. However, we got only a marginal improvement by adding paraphrased words proposed in Section 3.3.2. We guess the reason for this has two folds: first, we had some difficulties in finding the original word of the changed word, and using only the fastText may not be helpful. For example, the word "play" changed to "work" in the generated paraphrase, while using the fastText, the system found another word instead of "work". The second reason can be eliminating not-changed words that cause a discontinuity in the word context.

On the other hand, adding more information from WordNet proposed in Section 3.2 is not very helpful, and the accuracy is not changed compared to the base method proposed in Section 3.1. This has happened because of the used way for adding this information to the input and the shape of information. For example, adding only the two first synsets or adding lemmas or glosses of synsets instead of their lexnames may cause better results.

It is clear that there is a variation in the results of each method for different runs. The last column of the table shows the standard deviation of the five obtained results for each method. The second and third methods have smaller STD that means these methods are robust to random initialization classifier's weights, and we can trust more to the results.

In summary, adding generated paraphrases from sentences increase the performance of the model. However, using a better method for generating paraphrases can lead to better results.

5.1 Conclusion

In this work, we proposed four BERT-based methods for the WSD task. To handle some issues such as the same contexts for different word meanings, we proposed to add some extra information from WordNet or generate paraphrases of sentences. We trained and evaluated our proposed methods on the MCL-WiC training and testing sets, respectively. The results show that adding generated paraphrases as an additional input to the BERT can be helpful. Although, the performance of the paraphrase generation method plays an important role. So, for future works, using different methods of generating paraphrases can be considered. Besides, as mentioned before, adding only the two first synsets or adding lemmas or glosses of synsets instead of their lexnames from WordNet can be investigated. Due to lack of time, we could not spend too much effort on this challenge, and we leave it for the future.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiaju Du, Fanchao Qi, and Maosong Sun. 2019. Using BERT for word sense disambiguation. *arXiv preprint arXiv:1909.08358*.

Ping Guo, Yue Hu, and Yunpeng Li. 2020. MG-BERT: A multi-glosses BERT model for word sense disambiguation. In *International Conference on Knowledge Science, Engineering and Management*, pages 263–275. Springer.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China. Association for Computational Linguistics.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.

Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT make any sense? interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*.

Boon Peng Yap, Andrew Koh, and Eng Siong Chng. 2020. Adapting BERT for word sense disambiguation with gloss selection objective and example sentences. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 41–46, Online. Association for Computational Linguistics.