Natural Language Processing - Homework 3
Word Sense Disambiguation of Word-in-Context data

Lorenzo Nicoletti - 1797464
Università di Roma "La Sapienza"
nicoletti.1797464@studenti.uniroma1.it

Abstract

Word Sense Disambiguation (WSD) is the task of identifying the most suitable sense of a word expressed in a sentence by extracting its semantic meaning given the context where the word occurs. The following sections of this report will exhaustively analyze the proposed approach to solve the task, describing the starting point of the work, the adopted architectures and the achieved results, mentioning the advantages and the encountered difficulties.

1 Related work

The proposed approach is strongly based on the recent innovations in the open problem of WSD with the introductions of *Transformers* (Vaswani et al., 2017), model architectures that rely on attention mechanisms and have widely surpassed recurrent neural networks in terms of performances, time and parallelization. In particular, the project has exploited the breakout of *BERT* (Devlin et al., 2019), a very recent bidirectional encoding language model that will be fine-tuned, trained and tested on the evaluation framework of Raganato et al. (2017), that provides the *SemCor* train set and five test sets from the *Senseval/SemEval* series. Moreover, the proposed implementation is fully centered on some recent trends proving a boost of performances that a combination of the sentence with the definitions (or glosses) of the ambiguous word in that sentence can bring, as testified by Huang et al. (2019) first and Yap et al. (2020) later.

2 Method

2.1 Preprocess: WordNet and BertTokenizer

Data preprocessing is the very first crucial aspect that this work has faced. The aforementioned requirement of having access to sense keys and glosses is satisfied through the famous and easily accessible *Princeton WordNet* (Miller et al., 1990) (version 3.1): each data item, namely a sentence containing a bunch of words to be disambiguated, creates a batch of elements composed by the sentence itself and the glosses of each of these ambiguous words. The second essential operator of this stage is the *BertTokenizer*, a tokenizer and a converter able to parse data and make them suitable for BERT models. Since, in fact, the input of BERT must be embedded into numerical representations, each sentence-gloss pair is tokenized such to obtain the expected format in case of two-sentences input (i.e. *[CLS] Sentence [SEP] Gloss [SEP]*, where *[CLS]* and *[SEP]* are respectively the classification and the separator tokens), that is then converted to vocabulary indices together with the related attention mask, to make model distinguish real data from padding, and segment IDs, to differentiate between words of the first and of the second sentence. Moreover, the ambiguous word is surrounded by special added target tokens, *[TGT]*, to highlight its occurrence in the context. *Figure* 1 offers a simple example of how the whole preprocessing stage is performed; at the end, data are ready to be passed to the language model.

2.2 BERT models and fine-tuning

The proposed approach has tested several models belonging to the BERT family that are briefly listed.

BERT-base-uncased: composed by 12 layers, 12 attention heads with a hidden dimension of 768 and a number of 110M total parameters.

BERT-large-uncased: composed by 24 layers, 16 attention heads with a hidden dimension of 1024 and a number of 336M total parameters.

BERT-large-uncased-whole-word-masking: BERT-large architecture where the Whole Word Masking (WWM) technique of masking all the tokens corresponding to a word at once is applied.

According to the BERT literature, state-of-the-art performances on multiple classification tasks (included WSD) are achievable by fine-tuning the model and adding a linear fully-connected layer on top of it, therefore all the adopted BERT models have an extra (hidden size $\times$ 1)-dimensional layer whose output is used for evaluation and prediction.

3 Experiments

The experiment step has faced tremendous difficulties: the huge dimension of the *SemCor* train set, combined with the incredibly high number of parameters of the network, has rapidly exceeded the limited resources available for this project, always provoking a saturation of the device memory. Some safety measures have been adopted to at least start and proceed as long as possible. These have concerned about preserving the memory by removing all the unnecessary data, reducing the batch size until training was able to start by gradually resizing from a dimension of 128 to 4 and, more particularly, adapting a different training procedure: instead of simply fine-tuning the model, the implementation has verted also on the so-called pre-training "as is" technique. This 'exit strategy', based on freezing BERT weights and backpropagating only for the classification layer on top, has lengthened the execution and guaranteed a deeper learning for the classifier. The above limitations, despite their blockage of a traditional flow of the experiments, have suggested different solutions not even considered at the beginning of this work and the related results have exceeded the expectations.

4 Results

4.1 Evaluation on WSD task

The explained limits have caused a poor set of experiments with short trainings, where all the BERT models have shared the same configuration shown in *Table* 1 and composed of *Dropout* (Srivastava et al., 2014), Cross Entropy as loss function and Adam optimizer with *Weight Decay* (Loshchilov and Hutter, 2019) with desired learning rate and epsilon for numerical stability. BERT_{base}, thanks to its smaller architecture, has proceeded for a longer time in the training and is, therefore, the only one to be evaluated in terms of F1 score and loss on the available WSD validation sets. A comparison between training procedures is reported in *Table* 2, where both models behave extremely well if considering the initial conditions: the average loss of 1 point, marker of an uncompleted training, is quite high; however, the achievements in terms of F1 are strongly encouraging, especially with the "as is" freezing. Except for the poor 65% for *SemEval 2007*, the average value is stable with a confident 70-71% of accuracy and also a 74-75% as the best performance for *SemEval 2013*. These results are overall satisfying and acceptable, taking into account the impossibility faced during the experiments and that state-of-the-art models have recently overcome the threshold of 80% (Bevilacqua and Navigli, 2020), only 10 points above the proposed approach. Analyzing now the performances on the given development set summarized in *Table* 3, BERT_{base} networks are the best model in terms both of loss and F1, where BERT_{base}-freezed reaches a 67% score of accuracy, highly surpassing the models of BERT_{large} and $\text{BERT}_{large-wwm}$, that are not so efficient because of their greater structure and, consequently, their shorter training. A completed procedure would have reasonably guaranteed more confident and robust results.

4.2 "Bridge" between WSD and WiC

The most exciting considerations can be derived by analyzing the exploitation of WSD systems in the Word-in-Context (WiC) task. A comparison between BERT_{base} and the best model of the previous project on 'pure' WiC (denoted here as 'Baseline') is provided in *Table* 4, which testifies consistent improvements with respect to all the three adopted metrics (precision, recall and F1), to each of the two binary classes and to the average. The Baseline is surpassed in every measurement with a relevant difference of even 11-14 points in most of the cases. Another meaningful aspect is reported in *Table* 5: the proposed implementation leveraging WSD affects positively also WiC performances, registering the highest gain of +10.66% obtained with the accuracy value of 0.7836 for freezed BERT. The confusion matrices, shown in *Figure* 2, highlight this gap between models: considering the different number of entries, the ability of the WSD systems to better recover on false positives and negatives is one of the strengths of the implemented approach. A final graphic visualization of the introduced benefits in the topic is offered in terms of *ROC curves and AUC scores* (Bradley, 1997) (*Figure* 3), where the gain is perceived by observing the shapes of the curves, with a definitive improvement proved by the differences between the corresponding scores.

5 Extras

This is just a brief mention to the 'Extras' seen in this work: Machine Learning practises like the usage of Dropout have been applied (Early Stopping and best checkpoint saving were implemented but not used and even mentioned because of the problems in the training), pre-trained contextualized embeddings were adopted through BERT, comparative analysis among models and informative plots are attached (please, refer to the Appendix below while discussing images or tables in the report). I also notice that freezing ("as is") BERT is not so diffused, preferring a fine-tuning, but it can be indeed a good solution in some cases, as this project suggests.

References

Michele Bevilacqua and Roberto Navigli. 2020. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864, Online. Association for Computational Linguistics.

Andrew P. Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159.

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.

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.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, 3(4):235–244.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research (JMLR):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

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.

A Images

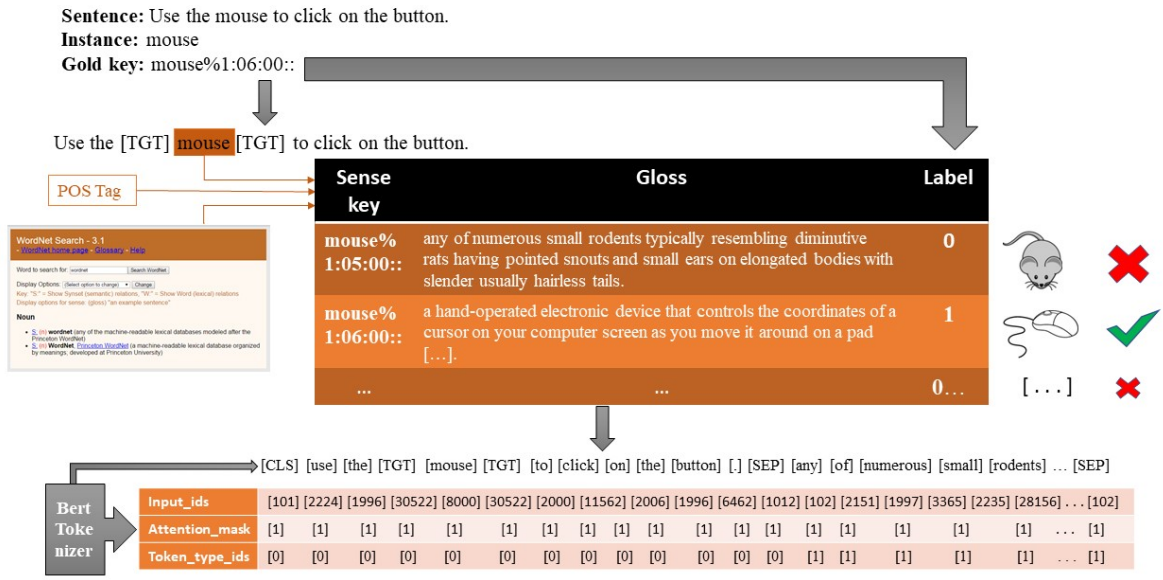
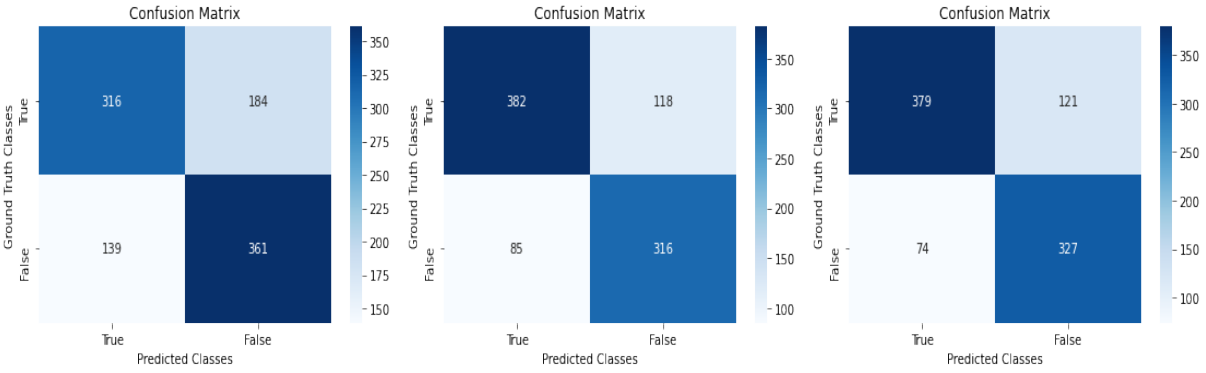


Figure 1: Example of WordNet + BertTokenizer preprocessing on a sentence, given an ambiguous word (instance) contained therein and its gold sense key: a batch of items is created such that each item is the combination of the sentence and one of the WordNet glosses of the instance, the binary labels are straightforwardly derived. BertTokenizer provides the BERT input taking as input the tokenized sentence-gloss pairs.



(a) Baseline on 1000 entries (b) BERT$_{base}$-finetuned on 901 entries (c) BERT$_{base}$-freezed on 901 entries

Figure 2: Confusion matrices of the compared models of Baseline and BERT$_{base}$ on WiC data. Please, note that the Baseline matrix is directly taken from the previous project on WiC Disambiguation.

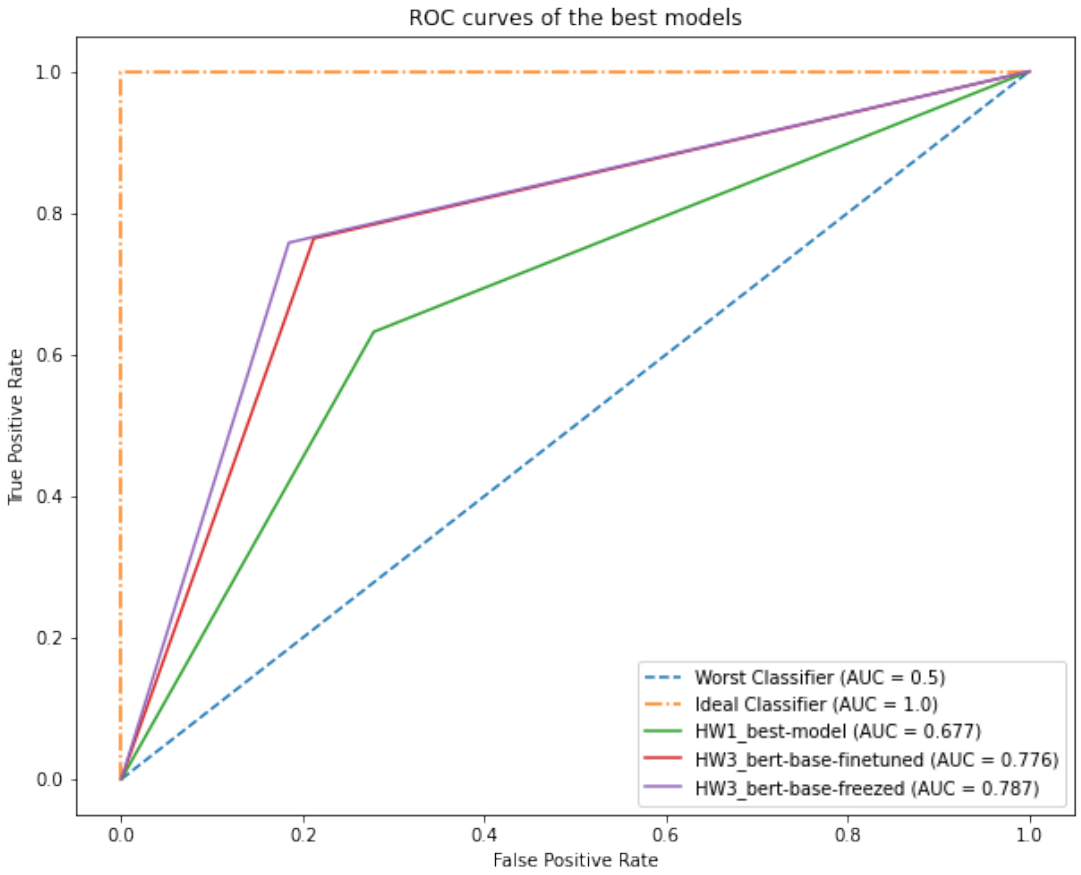


Figure 3: ROC curves and related AUC scores (on the bottom-right corner) of the compared models of Baseline and BERT$_{base}$ on WiC data. Please, note that the Baseline values are directly taken from the previous project on WiC Disambiguation.

B Tables

Model	Dropout	Loss	Optimizer	Learning Rate	Epsilon
BERT$_{base}$	0.1	Cross Entropy	AdamW	$2e-5$	$1e-8$
BERT$_{large}$	0.1	Cross Entropy	AdamW	$2e-5$	$1e-8$
BERT$_{large-wwm}$	0.1	Cross Entropy	AdamW	$2e-5$	$1e-8$

Table 1: Tested BERT models configuration with dropout, loss, optimizer, fixed learning rate and epsilon, where 'AdamW' stands for Adam optimizer with Weight Decay and 'BERT$_{large-wwm}$' for BERT-large-uncased-whole-word-masking model.

Model	Metric	SemEval2007	SemEval2013	SemEval2015	Senseval2	Senseval3	ALL
BERT$_{base}$-finetuned	F1	0.6176	**0.7482**	0.6830	0.7023	0.6805	0.6993
	Loss	1.3361	**0.7953**	**1.0437**	0.9936	1.0936	1.0223
BERT$_{base}$-freezed	F1	**0.6549**	0.7403	**0.6996**	**0.7248**	**0.6886**	**0.7112**
	Loss	**1.3329**	0.8236	1.0972	**0.9421**	**1.0039**	**1.0016**

Table 2: Performances of BERT$_{base}$ models in terms of F1 score and Loss on the six evaluation datasets: five belonging to the *Senseval/SemEval* series and the sixth as a concatenation of the previous ones.

Model	Training Set	Batch Size	Loss on Dev Set	F1 on Dev Set
BERT$_{base}$-finetuned	SemCor	4	**0.9606**	**0.6620**
BERT$_{base}$-freezed	SemCor	4	**0.9129**	**0.6687**
BERT$_{large}$	SemCor	2	1.2764	0.5305
BERT$_{large-wwm}$	SemCor	2	1.5393	0.3846

Table 3: Performances of the BERT models in terms of F1 score and Loss on the provided development dataset (Dev Set). BERT$_{large}$ and BERT$_{large-wwm}$ required a smaller batch size of 2 but their training was still too short with, consequently, worse results.

Model	Class	Precision	Recall	F1
Baseline	1	0.69	0.63	0.66
	0	0.66	0.72	0.69
	avg	0.68	0.68	0.68
BERT$_{base}$-finetuned	1	0.82	**0.76**	0.79
	0	**0.73**	0.79	0.76
	avg	0.77	0.78	0.77
BERT$_{base}$-freezed	1	**0.84**	**0.76**	**0.80**
	0	**0.73**	**0.82**	**0.77**
	avg	**0.78**	**0.79**	**0.78**

Table 4: Precision, Recall and F1-score for the models performing the Word-in-Context (WiC) task with respect to each class and to the average.

Model	WiC Accuracy	Improvement
Baseline	0.6770	-
BERT$_{base}$-finetuned	0.7747	+ 9.77%
BERT$_{base}$-freezed	**0.7836**	**+ 10.66%**

Table 5: Performances of the WSD models for the WiC task in terms of accuracy and improvement in percentage.