WSD of Word-in-Context Data
Bonus exercise

Giuseppe Masi
Sapienza Univesity of Rome
Matricola 1962771
masi.1962771@studenti.uniroma1.it

1 Introduction

Since many words can be interpreted in multiple ways depending upon the context of their occurrence, this homework addresses a Word Senses Disambiguation task [3] in which we are required to determine the exact sense of a target-word occurring in a sentence, relying on a lexical resource like WordNet. Moreover, once we get a prediction of the sense of a target-word, we get for free a prediction on the pairs of target-words in two different sentences of the Word-in-Context task.

In this work, I implement the models described in [2], in which the definition of a word sense meaning, i.e. the gloss, provided by WordNet, is used to influence the decisions of the model when predicting the sense of a target-word in context.

2 Dataset

I use `SemCor` and `SemEval2007` [4] respectively as training and development corpus, and the provided WiC `dev.jsonl` file for testing.

Starting from the above corpora, I generate *sentence-gloss* pairs for each possible sense of the target-word in the *sentence*. The senses for the target-word are retrieved from WordNet considering its lemma and its Part-of-Speech tag. Using the POS tag we have fewer possible senses for a target-word w.r.t. not using it. This is meant to help the model as he has to choose between fewer significant candidates.

Each *sentence-gloss* pair (denoted as *context_sentence*) is associated to the sense-key of the *gloss* and to the label *True/False* if the gold meaning of the target-word in the *sentence* is that one explained by the *gloss*. Thus, we obtain a sentence-pair classification problem.

Since a single sentence in the corpora may contain multiple target-words, I consider the same sentence multiple times with a different target-word each time.

In this way, the number of rows in the final training dataset grows fast, so I also test to limit the number of target-words to consider in each sentence (like considering only the first one for each sentence of the training corpus).

3 Generating *context_sentence*s

Starting from a *sentence-gloss* pair, `[CLS]` and `[SEP]` special tokens are added to make it suitable for the input of BERT model [1]. I implement two variants for generating the *context_sentence*s.

The first one just concatenate the *sentence* and the *gloss*, obtaining inputs of the shape:
`[CLS] sentence [SEP] gloss [SEP]`

In the second one, signals to highlight the target-word are added. In particular the target-word in the *context_sentence* is surrounded by `'"'`. Moreover it is replicated before the gloss obtaining inputs of the shape:
`[CLS] sentence [SEP] target-word: gloss [SEP]`

A complete example of the variants is shown in Figure [1].

4 Model Architecture

Given the *context_sentence*, I use the pre-trained uncased BERTTOKENIZER to obtain a sequence of tokens to give in input to the pre-trained uncased BERT_{BASE}, that I fine-tuned on the WSD task during training backpropagating the error on the BERT layer. It is composed of 12 Transformer blocks, 768 hidden layer, and 12 self-attention heads.

I consider the final hidden state of BERT as the latent representation of each token in the input sequence.

On top of it, there is a classifier (MLP) consisting of two linear layer, with a RELU and a DROPOUT [5] layer in between, and a SIGMOID on top.

The overall architecture is shown in Figure [2] and the hyper-parameters are shown in Table [1].

5 Training time

Given a *sentence-gloss* pair, as ground truth I use the *True/False* label if the gold meaning of the target-word in the *sentence* is that one explained by the *gloss*. So, for each *context_sentence* the model provides a probability that the above label is *True* or *False*.

The loss function used is the BINARY CROSS ENTROPY function.

6 Validation time

At validation time, I use the trained model to get a prediction *True/False* for each *sentence-gloss* pair in the dev set. So, given a *sentence* I obtain a list of probability for each sense of the target-word in the *sentence*. In order to predict the meaning, I take the sense corresponding to the highest probability.

7 Test time

In the same way of validation time, at test time I obtain a predicted meaning of the target-word in the *sentence*. Moreover, using the prediction for the WSD task, I also obtain a prediction for the WiC task on a sentence pair of the test set (just checking if the two predicted senses for the two sentences are the same or not).

8 Experiments

Before running the model considering the whole `SemCor`, I tested different settings to generate the inputs considering a subset of all the target-words. The dimensions of the considered datasets are shown in Table [2].

Token-CLS-1TW-NP. The first experiment consider only (the first) one target-word for each *sentence* in `SemCor`. I take the final hidden state of BERT corresponding to the target-word token(s). Since the BERTTOKENIZER uses a WordPiece approach to tokenize words, if the target-word corresponds to more than one token, I average them.

Sent-CLS-1TW-NP. The second experiment is like the first one but I take the final hidden state corresponding to the first token `[CLS]` as the representation of the whole *context_sentence*.

Sent-CLS-WS-1TW-NP. The third experiment is like the second one but the signals to highlight the target-word are added.

Sent-CLS-WS-1TW-P. This experiment is like the previous one, but pre-processing steps are added. In particular, before generating the *context_sentence*, both the *sentence* and the *gloss* are lower-cased, and stop-words and punctuation symbols are omitted.

Sent-CLS-WS-ATW-NP. The last experiment is like the third one but it considers all the target-words for each *sentence* in `SemCor`.

9 Results

For each test, I train the model for 5 epochs and I use the accuracy to evaluate the performances (which is equal to the $F1$-measure since the model always provides an answer).

The best performances of the models on the validation set are summarized in the Table [3].

The performance trends across the epochs of the models on the test set are reported in the Plot [3] for the WSD task, and in the Plot [4] for the WiC task (with the best ones summarized in Table [4]). Figure [5] and Figure [6] report the confusion matrix of the best models for the WiC task.

As we can see, the best model for the WSD task is that one considers all the target-words with signals for them added, without pre-processing steps. The best model for the WiC task is equal to the previous one, but considers only one target-word for a sentence. So we can think that removing stop-words and the other pre-processing steps worsens performances. Moreover, highlight the target-word in the *context_sentence* improves performances even w.r.t. the model that considers directly the hidden representation of the target-word token(s).

10 Conclusions

Anyway, the model SENT-CLS-1TW-NP does not perform much better than SENT-CLS-ATW-NP on the WiC task, so I decided to submit the latter one.

I would have expected that consider all the target-words in the training corpus would improve significantly the performances, but that was not the case. In fact, between the two models, there is a $2-4\%$ performance difference on the WSD task, and about 1% on the WiC task.

11 Figures

SENTENCE: No clause in a contract shall be interpreted as evading the responsibility of superiors under international law.
LEMMA TARGET-WORD: superior
POS-tag: NOUN
GOLD KEY SENSE: superior%1:18:01::

CONTEXT-GLOSS PAIRS

CONTEXT_SENTENCE	*KEY SENSE*	*LABEL*
[CLS] No ... superiors ... law. [SEP] one of greater ... [SEP]	superior%1:18:01::	True
[CLS] No ... superiors ... law. [SEP] the head of a ... [SEP]	superior%1:18:02::	False
[CLS] No ... superiors ... law. [SEP] a combatant ... [SEP]	superior%1:18:03::	False
...	...	...

CONTEXT-GLOSS PAIRS WITH SIGNALS TO HIGHLIGHT THE TARGET-WORD

CONTEXT_SENTENCE	*KEY SENSE*	*LABEL*
[CLS] No ... "superiors" ... law. [SEP] superior : one of greater ... [SEP]	superior%1:18:01::	True
[CLS] No ... "superiors" ... law. [SEP] superior : the head of a ... [SEP]	superior%1:18:02::	False
[CLS] No ... "superiors" ... law. [SEP] superior : a combatant ... [SEP]	superior%1:18:03::	False
...	...	...

Figure 1: Example of generating *context_sentence* starting from a sentence in the `dev.jsonl` sentence.

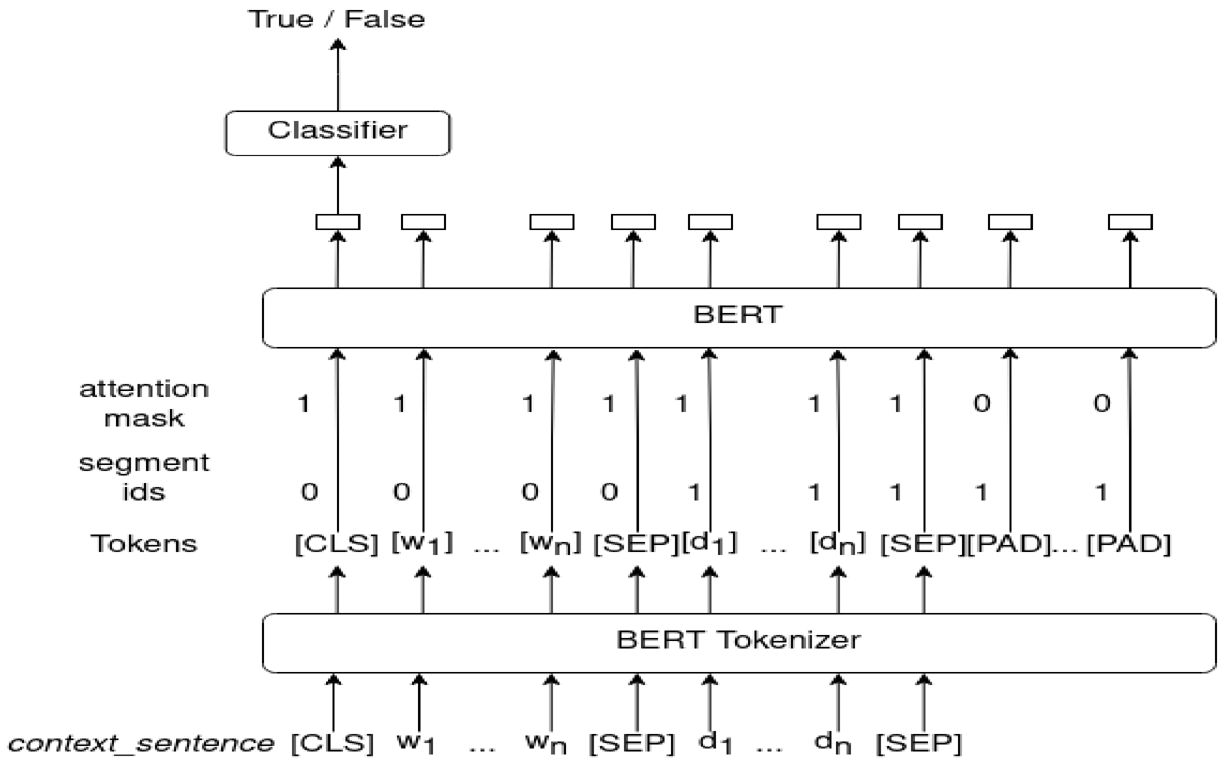

Figure 2: The overall model architecture, considering the first token `[CLS]` as hidden representation of the whole *context_sentence*. Actually, the Tokenizer may tokenize one word in more than one token (WordPieces approach).

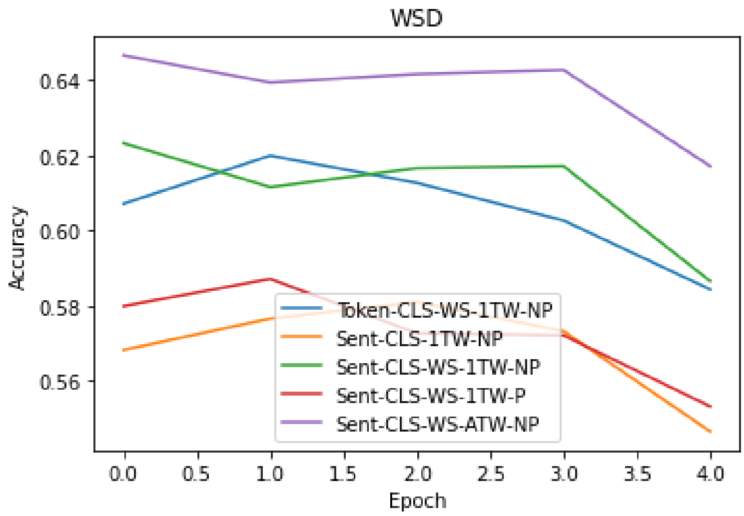

Figure 3: The plot about the accuracy of the models on the test set for the WSD task.

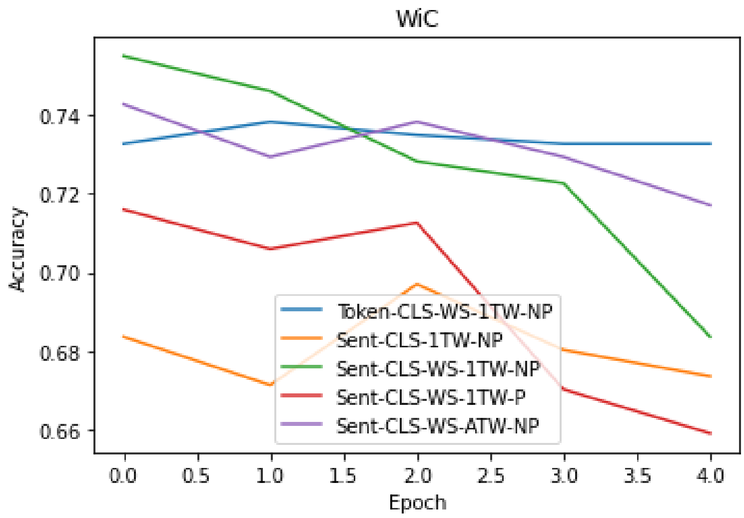

Figure 4: The plot about the accuracy of the models on the test set for the WiC task.

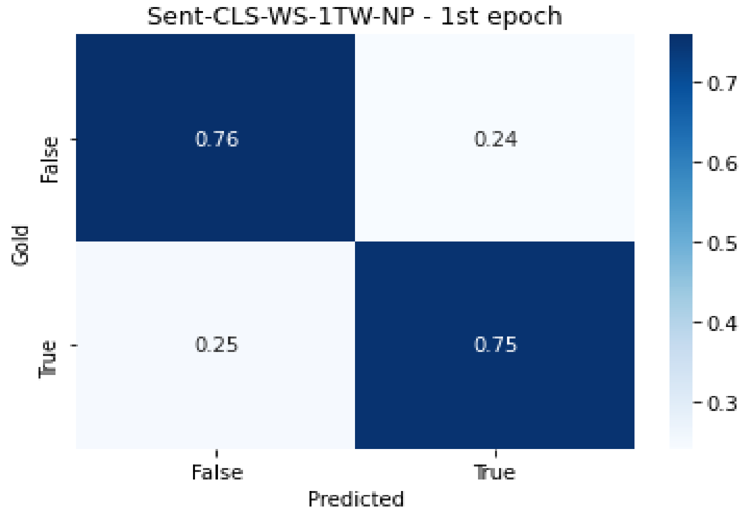

Figure 5: The confusion matrix about the model Sent-CLS-WS-1TW-NP on the test set for the WiC task at epoch n.1.

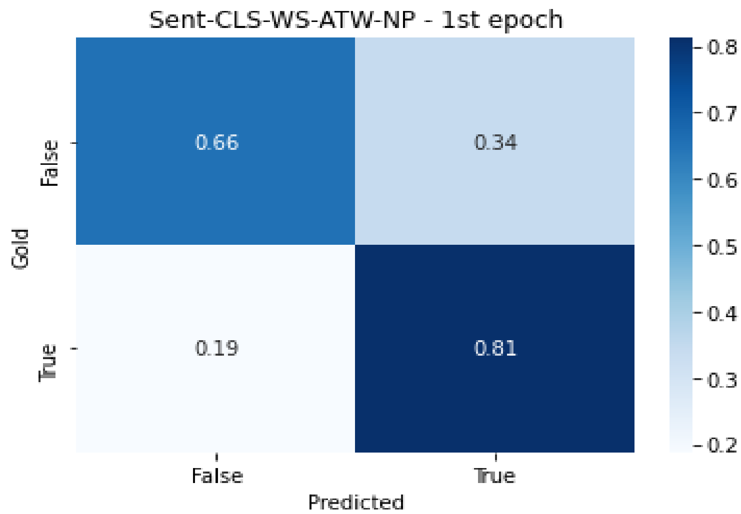

Figure 6: The confusion matrix about the model Sent-CLS-WS-ATW-NP on the test set for the WiC task at epoch n.4.

12 Tables

Name	**Value**
N. nodes Lin1	768
N. nodes Lin2	384
Dropout p.	$1e-1$
Batch size	16
N. epochs	5
Optimizer	$AdamW$
Lr	$2e-5$

Table 1: Hyper-parameters of the model.

Corpus	**N. target-words**	**N. *context_sentences***
`SemCor`	1	$308.5K$
`SemCor`	*All*	$1.5M$
`SemEval2007`	*All*	$3.9K$
`dev.jsonl`	*All*	$8.6K$

Table 2: Dimensions of the datasets.

Model	**Accuracy**	**Epoch**
Token-CLS-1TW-NP	63.7%	3^{rd}
Sent-CLS-1TW-NP	57.6%	4^{th}
Sent-CLS-WS-1TW-NP	62.4%	2^{nd}
Sent-CLS-WS-1TW-P	59.6%	3^{rd}
Sent-CLS-WS-AWS-NP	**66.8%**	3^{rd}

Table 3: Best performances of the models on the validation set (`SemEval2007`).

Model	Accuracy WSD - Epoch	Accuracy WiC - Epoch
Token-CLS-1TW-NP	$62.0\% - 2^{nd}$	$73.8\% - 2^{nd}$
Sent-CLS-1TW-NP	$58.1\% - 3^{rd}$	$69.7\% - 3^{rd}$
Sent-CLS-WS-1TW-NP	$62.3\% - 1^{st}$	$\mathbf{75.5\%} - 1^{st}$
Sent-CLS-WS-1TW-P	$58.7\% - 2^{nd}$	$71.6\% - 1^{st}$
Sent-CLS-WS-AWS-NP	$\mathbf{64.7\%} - 1^{st}$	$74.3\% - 1^{st}$

Table 4: Best performances of the models for the WSD task and the WiC task on the test set (`dev.jsonl`).

References

[1] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[2] Luyao Huang et al. “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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3509–3514. DOI: 10.18653/v1/D19-1355. URL: https://www.aclweb.org/anthology/D19-1355.

[3] Roberto Navigli. “Word Sense Disambiguation: A Survey”. In: *ACM Comput. Surv.* 41.2 (Feb. 2009). ISSN: 0360-0300. DOI: 10.1145/1459352.1459355. URL: https://doi.org/10.1145/1459352.1459355.

[4] Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. “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*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 99–110. URL: https://www.aclweb.org/anthology/E17-1010.

[5] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.