Two approaches for Word-in-Context disambiguation

Niccolò Morabito
1808746
morabito.1808746@studenti.uniroma1.it

1 Introduction

The aim of this work is to build a binary classifier for Word-in-Context disambiguation. Given two English sentences, the classifier must determine whether the indicated target words have the same meaning.

I worked in parallel on two approaches (a simpler one and that based on sequence encoding) since the former achieved better performance at the beginning while the latter seemed more promising.

The final models are the result of multiple experiments aimed to increase accuracy on the validation set through different combinations of preprocessing choices and several attempts on hyperparameters tuning and architectural options.

2 Pre-processing of sentences

In all the NLP tasks, pre-processing has primary importance. That is why several analysis and attempts focused on the different possibilities. In particular, the main alternatives were the following:

- Tokenization with or without punctuation;
- whether to remove stopwords or not;
- whether to perform lemmatization or not.

It is important to notice that, for each combination, the percentage of words recognized by the used GloVe embedding on the total number of words in the training set's sentences is always very high (from a minimum of 98,81% to a maximum of 99,61%). Of course, to include punctuation in the tokenization lead to a lot more vectors and consequentially to more information provided to the neural network. This is not necessarily a good outcome, that is why the real comparative analysis has been carried out on the accuracy that the various pre-processing choices achieved. The results are summarized in the Table 1. Different approaches require different pre-processing solution: punctuation and stopwords need to be removed in the case of the simpler approach, which computes the average of all the vectors corresponding to a sentence. In this case, the embedding of a comma, a "while" or an "if" represent just noise. On the other hand, with a sequence encoding approach, these symbols or words seem useful during the learning process and the LSTM is able to elaborate the additional information.

3 Pre-trained embeddings

In order to extract features from sentences, a pre-trained word embedding has been used since it helps to get good results for sentence classification (Kim, 2014). In particular, pre-trained **GloVe (Global Vectors for word representation)** (Pennington et al., 2014) word vectors have been loaded into the code. I used the Wikipedia 2014 version and, in particular, the one with 50 features per word. It is composed of 6B tokens, 400K uncased vocabs. All the attempts with a greater number of features (100, 200 or 300) caused overfitting more easily and have been excluded.

4 Simpler approach

The method which achieved the best results is based on a simple standard neural network composed of five consecutive linear layers: the input layer has 2*50=100 nodes: 50 is the number of features of a vector in the GloVe vectors and the two sentences are combined concatenating the mean between the vectors of all the words in the first sentence and the vectors of all the words of the second one. The output layer reduces 250 nodes to one value which is passed to the sigmoid function to obtain a value between 0 and 1. This value is finally rounded to obtain the prediction.

The selected optimization algorithm for both approaches is the Adam (Kingma and Ba, 2017),

which is one of the most popular and widely used, especially in NLP researches. Some of the tuned hyperparameters are summarized in the Table 2 distinguishing between the two approaches.

5 Sequence encoding approach

The architecture of the model based on sequence encoding is slightly more complex. First of all, there is an embedding layer that is initialized with the vectors of the pre-trained GloVe embedding. Before passing a sentence to this layer, padding is added in order to have always the same input length - the resulting embedding is not aggregated because an LSTM is going to be used to evaluate the whole sentence from left to right.

The two resulting tensors are then passed to an LSTM layer *separately* and padding is removed away from each output retrieving the last token value for each sentence. This is going to contain the "summary" of the entire sentence without any wrong information for padding. The original idea was to concatenate the sentences using a separator and elaborate the full resulting sequence passing its last token to the following neural network. In this way, however, also the padding would have been taken into account.

The two resulting vectors are then concatenated and passed to a neural network composed of 3 sequential layers which are finally scaled by the sigmoid function. The number of nodes is an order of magnitude lower than that of the simpler approach because this approach was more prone to overfitting.

The dropout layer is an effective technique for regularization and preventing the co-adaptation of neurons (Hinton et al., 2012). Despite the dropout rate that has been shown to be effective in the majority of scenarios is 0.5 (Kim, 2014), the best results were achieved setting p (i.e. the probability for elements of the input tensor to be zeroed) to 0.0001. Greater values led to a deterioration in the performance on both training and validation set.

Moreover, adding some weight decay to the Adam optimizer helped to slow down the natural overfitting too (Loshchilov and Hutter, 2019) [Table 2].

Other experiments were carried out on this approach. In particular, I tried to add more layers to the LSTM with an incrementing or decrementing number of units. I also implemented a bidirectional LSTM to process a sentence from right to left as well as from left to right, by concatenating the two resulting last tokens. In both cases, however, the accuracy values throughout the epochs were comparable to those obtained without a deep or a bidirectional LSTM but with greater learning times.

6 Results

The models have been evaluated during training using the validation test after each epoch, therefore the final results discussed in this section must not be considered definitive. Instead, they only have been used to select the best model during training before the validation accuracy starts to decrease (generally after 40 epochs). Confusion matrices can be referred to for numerical results [3 and 4]. As the Figure 1 and Figure 2 show, only the first part of the training process is really useful for the model since these neural networks tend very easily to overfit.

The overall results are summarized and compared in the Table 3. The best model achieves about 70% on all the metrics taken into account. Also during the training, all the metrics were always balanced, meaning that all the models do not tend to predict more easily a class than the other.

As we can see, the second approach, despite is based on sequence encoding and has a more complex architecture, is not able to overcome nor reach the performance of the simplest approach which is based only on a simple embedding and three linear layers.

7 Conclusion

The above results, however, do not imply that the simplest way to approach the word-in-context disambiguation problem is also the one able to reach the best results. As anticipated, these results are not fully reliable; moreover, they can be improved since other ideas can be followed. In particular, more attempts should be done to add information about the target words to the network. Both approaches are based on the context and the network learns from the vectors mean or from the sequence encoding, without taking into account the target words it should compare. A few attempts were carried out during this work - adding the embedding vector corresponding to the lemma of the target words to the network and adding the encoding of its POS, both without any particular success - but more experiments can be carried out.

References

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

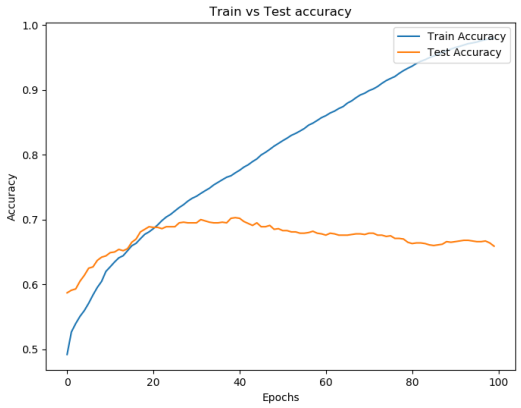

Figure 1: Accuracy on the training and validation sets with respect to the epochs of training for the simpler approach.

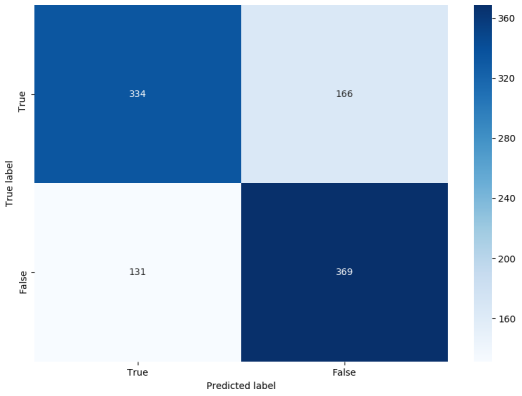

Figure 3: Confusion matrix of the best simpler model computed on the validation set.

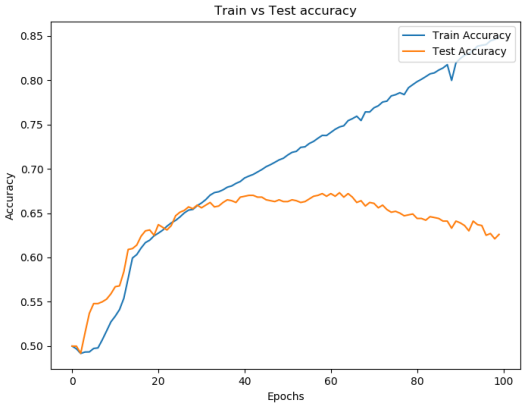

Figure 2: Accuracy on the training and validation sets with respect to the epochs of training for the sequence encoding approach.

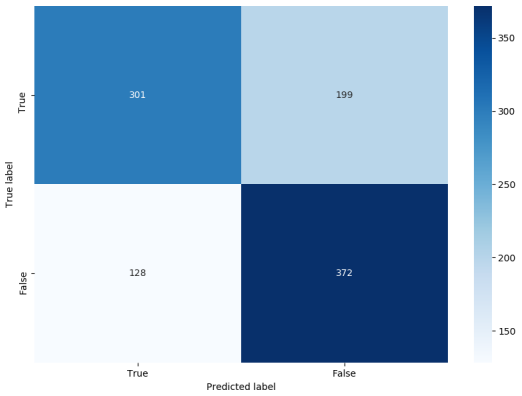

Figure 4: Confusion matrix of the best sequence encoding model computed on the validation set.

Punctuation removed	Stopwords removed	Lemmatization	Accuracy (simpler)	Accuracy (sequence encoding)
✓	✓	✓	0.703	0.666
✓	✓	✗	0.699	0.670
✓	✗	✓	0.697	0.643
✓	✗	✗	0.702	0.661
✗	✓	✓	0.690	0.647
✗	✓	✗	0.697	0.645
✗	✗	✓	0.681	0.673
✗	✗	✗	0.698	0.667

Table 1: Maximum accuracy for the two approaches reached with all the possible combinations of pre-processing choices highlighting the best one for each approach.

	Optimization algorithm	Learning rate	Weight decay	Loss function
simpler approach	Adam	10^{-5}	10^{-5}	Binary Cross Entropy
sequence encoding	Adam	10^{-4}	10^{-6}	Binary Cross Entropy

Table 2: Some of selected hyper-parameters for the two approaches.

	Precision	Recall	F1-score	Accuracy
simpler approach	0.704	0.703	0.703	0.703
sequence encoding	0.664	0.659	0.656	0.659

Table 3: Main metrics for both the approaches.