Homework 1: Word-in-Context Disambiguation

Michele Conti
`conti.1599133@studenti.uniroma1.it`

1 Introduction

WiC disambiguation is a binary classification task. Each instance in the dataset has a certain target word w and two contexts (i.e., sentences) c_1 and c_2 in which w is contained. The task is to identify whether the occurrences of the target word in the two sentences carry the same meaning or not.

In this paper I present two architectures to address this task, both based on the use of pretrained embeddings and bilinear layers.

2 Preprocessing

Various options have been considered in order to preprocess the data, but, ultimately, I decided to follow a very basic pipeline in this phase. Specifically, all the pairs of sentences have been lowercased, all the punctuation has been stripped, and finally I removed all the stopwords and digits from the sentences. As for stemming and lemmatization, instead, I specifically decided to avoid these operations for two reasons: the first one being that I already had in mind to exploit pretrained word embeddings, which are often computed for non lemmatized/stemmed words, and also because I felt like the specific form of each word could help disambiguate one sentence from the other.

To create the vocabulary, it was my first decision to retain all the words appearing in the training dataset, no matter their frequency, since the training set is not very large (i.e., 26119 unique words after the preprocessing part).

Anyway, in the next sections we are going to see that even this step can be improved, creating the vocabulary from an external source. Specifically, I decided to inherit the vocabulary of Glove, an unsupervised learning algorithm for obtaining vector representations for words (Pennington et al., 2014).

3 Models

3.1 Model architecture

Two general structures have been used as baselines to assess the complexity of the given task, and to identify possible improvements in the models. Namely, the first structure I implemented is only based on linear layers, while the second one is composed by an LSTM followed by linear layers.

3.2 Embeddings

The very first module each architecture has is an embedding layer, where each token of each sentence gets projected into a vector space, so that the data can subsequently be fed to the next layers. This layer works as a lookup table between an index, indicating the position of the word in the vocabulary, and its vector representation, called embedding. The vector embeddings are randomly initialized, and they are then tuned during the training phase to minimize the loss function.

3.3 Embeddings aggregation

4 Experiments

From this point on, a wide range of features have been tested, and in the next sections we will explain each one of them in detail, keeping at each step only the ones that proved to be effective, actually increasing the performance of the model.

5 Result

6 Conclusion

References

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.