Homework 1 Written Report

NLP 2021 Course - Prof. Roberto Navigli

Andrea Trianni - 1806198
Master Degree in Computer Science
Sapienza University of Rome
trianni.1806198@studenti.uniroma1.it

1 Introduction

The task of this homework is Word-in-Context Sense Disambiguation, that consist in disambiguate polysemous words without relying on fixed set of word senses, but only using context information. In more detail we have to analyze words that are present in different pairs of sentences, and predict when they have the same meaning or not, without relying on any additional information except the contexts in which they occur.

2 Data Visualization

Both the dataset and the task are widely descript here (Navigli et al., 2021), anyway, before proceeding, it is a good practice to view and analyze the data we are going to use, to build models that can fit well the problem. As shown in image [2] the labels of the dataset are well balanced both in training and dev set. This is fine and allow me to don't care about fighting class unbalancing. The number of training samples is 8k, that is good enough to train a small neural model. Another important aspect is to understand what kind of words we are going to disambiguate. As shown in the image [4], they are mostly nouns, but also verbs, adverbs and adjectives. To keep things simple i decided not to incorporate this information into the models, since it is not very informative in the prediction process, because the lemma has the same POS in the pair of sentences. As shown in figure [3], the length of the sentences is distributed like a gaussian, analyzing it, i decide to fix the max size of the sentence to 32 token. This admit to fit most of the text samples and to not heavily pad some other sentences, that can add undesired noise in some cases. All this information is fundamental for the choices i've made in the construction of the models.

3 Preprocessing Stage

The machine learning pipeline starts from preprocessing the data, that means mainly clean the text and vectorize it, following NLP best practice. Without going deeply into the details, i am going to illustrate the two most important parts for the model: the vocabularies and the embeddings.

3.1 Vocabularies

All the vocabularies used in the homework are built only using words *lemmatized* from the training dataset, to avoid any risk of information leakage. A crucial decision for the perfomances is to decide how big has to be the vocabulary. A huge vocabulary does not guarantee to achieve best performance, because our models can suffer both of bias but also of variance problems, and the goal is to find the best tradeoff to not encounter underfitting or overfitting situations. For this reason i decide to use two vocabularies: the first one small (10k most popular lemmas) and another one with all the token founded (23k) The figure [1] clearly shown the size of a potential vocabularies respect the size of the training dataset. It is a good consideration to note that more than half of the total tokens are founded in the first quarter of the dataset.

3.2 Embeddings

To encode the token of the sentences i decide to use pretrained word embedding, in particular GloVe (Pennington et al., 2014). For the same reason explained in the Vocabularies section i have trained my models both on 300D embedding and 50D embedding. The GloVe embeddings, that are un-contextualized, were trained using a wikipedia dataset, that is similar enough to the sentences that i am going to use for training. All the embedding weights were frozen during the training stage.

4 Model Architectures

To solve the problem i design two neural models: one based on simple multi layer perceptron , and the other one using Bi-LSTM cells.

4.1 Multi Layer Perceptron

The architecture of MLP is illustrated in the figure [5]. It is fed with the embeddings of the two sentences, that are averaged across each dimension, and then concatenated together. The network is composed by 3 linear layer, 2 hidden and a final one with sigmoid. The hidden layers use ReLU activation function to model non-linear patterns and use Dropout and Normalization to fight overfitting and improve generalization. The layers progressively reduce the dimensionality of the vector space to capture hidden priors.

4.2 Bi-LSTM

A more complex approach was realized with recurrent neural network to try to catch sequence patterns and reach better performance. The architecture is illustrated in the figure [6]. The network takes as input the packed embeddings of the two sentences, that fed one by one a Bi-LSTM cell. I decided to set hidden matrix size equal to the size of the embeddings to establish a dimensional balance. For each sentence, the mean of the forward and backward pass of the last hidden state is taken as output. These lasts were concatenated together to pass trough a Dropout an Normalization layer. This feed the last part of the network that consist of two linear layers, with once again, dropout, normalization, and non-linear activation function.

5 Training Stage

As just a bit explained before, i trained a total of 8 different models, because i tried to use 2 sizes of vocabularies, 2 different dimensionality of embedding, both for MLP and LSTM models. Each of this was trained using a grid search strategy over learning rate and batch size. An example of this, for the best model founded is shown in table [1]. After any grid search the model with the lowest loss was selected as a candidate final model. The real final model selected to predict the labels of the test set is the one with the highest accuracy among these eight. Every network is trained using L2-Norm to regularize the models, and also adopt early stopping technique. The learning rate follow a dynamic schedule that halves it after the scores on dev set stops improving for more than some epoch. In all the models patience parameter is set to 3.

6 Results

The results obtained are quite satisfactory in any case, considering the limitations imposed by the encoding method and the available architectures. The results achieved by all the models are in the range [59%-65%]. As espected the best performance is reached by a LSTM model with an overall accuracy of 0.647. Anyway MLPs can still guarantee good results, and moreover the training plots of the MLPs seem to be smoother than the plots of the LSTMs. F1-Score, Precision and Recall are all stable between 63% and 65%. All the results can be read in the table [2]. Widely reasoning the best combination respect dimensionality of embedding and size of vocabularies seems to be 50D and 23k. This means that knowing more words helps the model to be more precise, and using a low-dimensional vector space helps the model to generalize (at least in this case).

7 Conclusions

Word-in-Context Sense Disambiguation is a very challenging task especially treated as a binary problem. With usual solutions and common architectures it seems that an accuracy of 70% cannot be exceeded. Better performances can be obtained by changing type of encoding for example using contextualized word embedding, or focusing on the models. We can optimize the recurring network even more or we can use more complex architectures, for example fine tuning BERT[1].

References

R. Navigli et al. 2021. Semeval-2021 task 2: Multilingual and cross-lingual word-in-context disambiguation. In *Proc. of the 15th International Workshop on Semantic Evaluation (SemEval), Online.*

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.

[1] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Model Filename	Batch Size	Learning Rate	Loss	Accuracy	Epochs
LSTM_50D_23300TOK	16	0.001	0.6221	0.6472	23
LSTM_50D_23300TOK	32	0.001	0.6275	0.6280	19
LSTM_50D_23300TOK	16	0.0005	0.6350	0.6008	30
LSTM_50D_23300TOK	32	0.0005	0.6441	0.6129	28

Table 1: Grid search training results for the best model. The search is over a 2x2 grid composed by learning rate and batch size. The max epoch parameter is set to 100. The model with the lowest loss is selected as result of the search.

Model	Embed. Dim	Vocab. Size	Accuracy	Loss	F1-Score	Precision	Recall
MLP	50	10k	0.6351	0.6516	0.6239	0.6389	0.6375
MLP	50	23k	0.6411	0.6317	0.6300	0.6467	0.6415
MLP	300	10k	0.6028	0.6425	0.5914	0.5989	0.5987
MLP	300	23k	0.6220	0.6374	0.6152	0.6255	0.6257
LSTM	50	10k	0.6300	0.6417	0.6219	0.6306	0.6297
LSTM	50	23k	0.6472	0.6221	0.6290	0.6563	0.6489
LSTM	300	10k	0.5917	0.6581	0.5727	0.591	0.5879
LSTM	300	23k	0.6159	0.6443	0.6027	0.616	0.6108

Table 2: All the results coming from the various grid searches. I have a total of eight model ready to predict on test set. The one with the higher accuracy is selected as the winner.

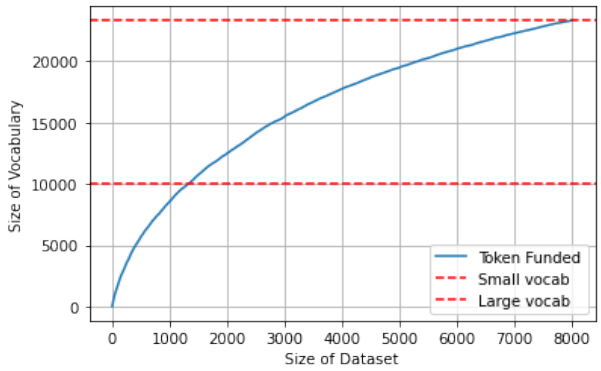

Figure 1: A plot to show how the size of the vocabulary can increase as the dataset do the same

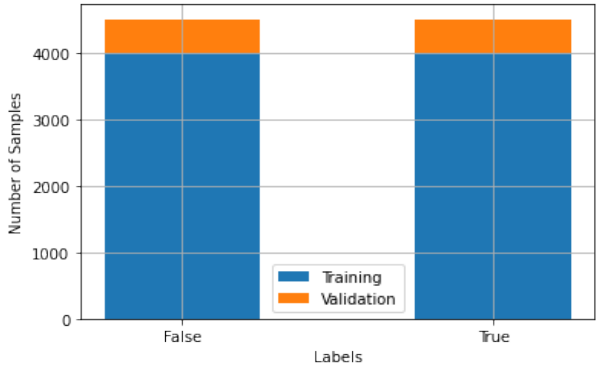

Figure 2: Histogram to visualize dataset size and class balancement.

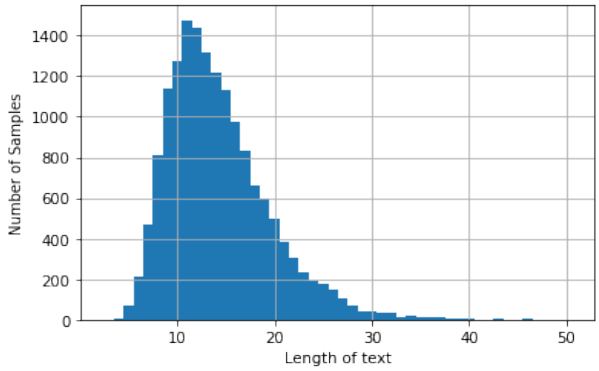

Figure 3: Distribution of the lengths of the sentences in the dataset. In the X-axis there is the number of token.

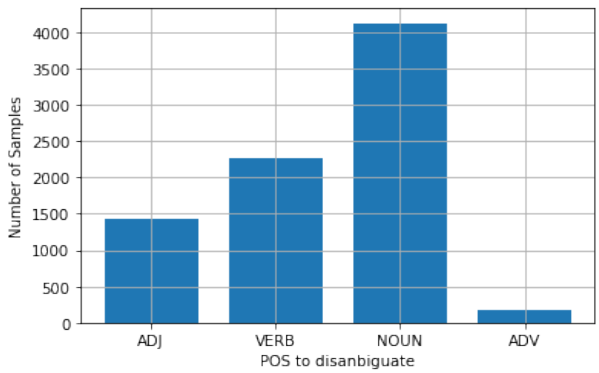

Figure 4: Histogram to visualize the distribution of the POS respect lemmas to disambiguate.

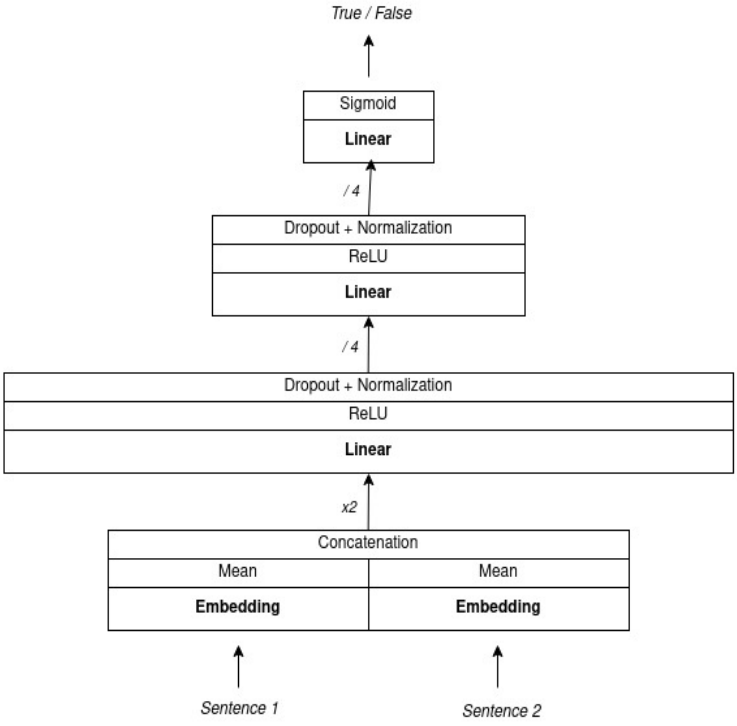

Figure 5: The architecture of the MLP network.

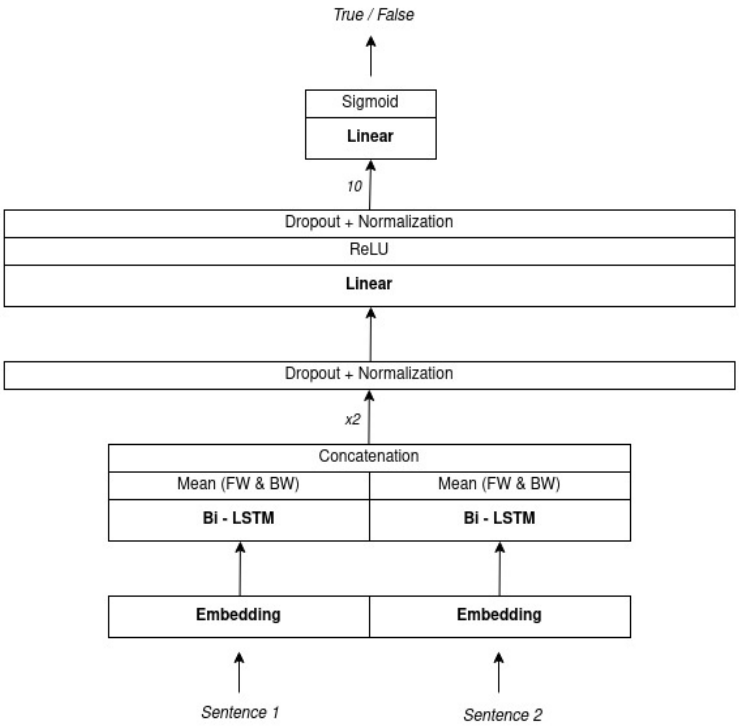

Figure 6: The architecture of the LSTM network.

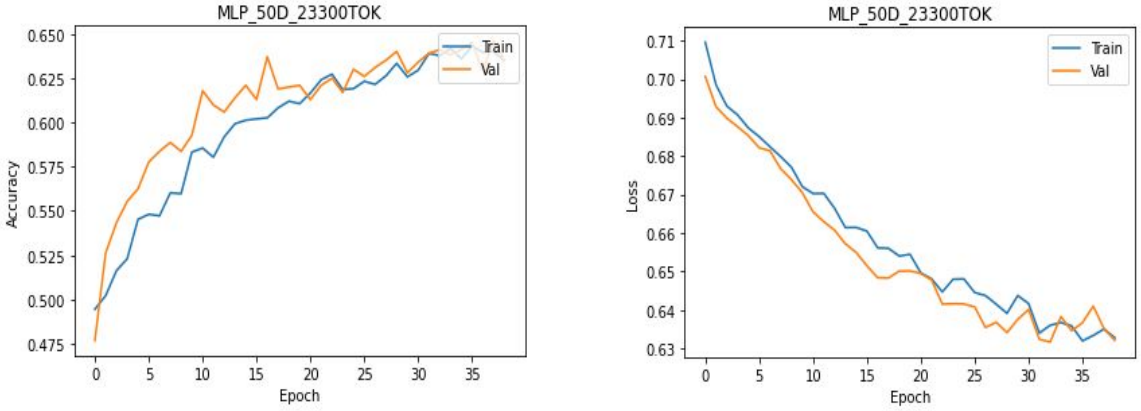

Figure 7: Training plots of MLP.

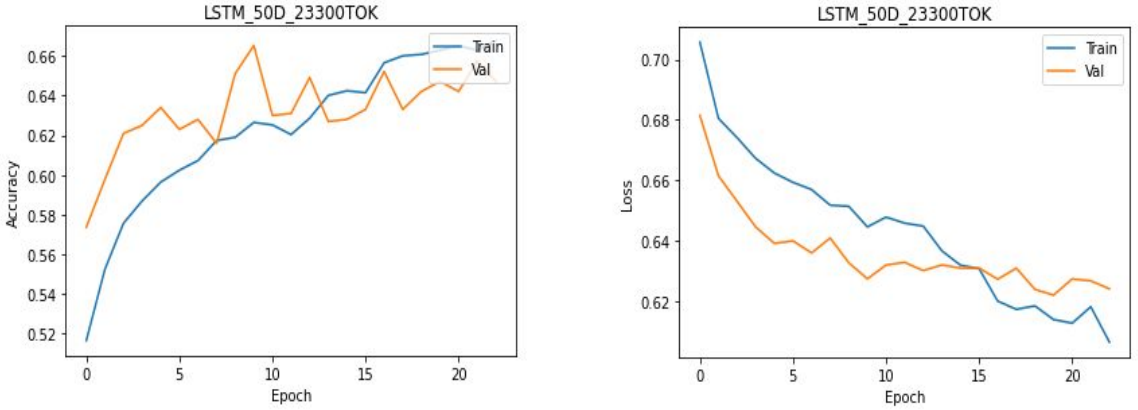

Figure 8: Training plots of LSTM.