Word-in-Context Disambiguation - Homework 1 NLP 2021

Francesco Montano 1744183
Sapienza University
montano.1744183@studenti.uniroma1.it

Abstract

Word-in-Context Disambiguation is the task of addressing the disambiguation of polysemous words, without relying on a fixed inventory of word senses. In this report I am going to propose a model based on a LSTM to solve this problem.

1 Data

The dataset used is already splitted in train and development: the train and development have 8000 and 1000 entries respectively. The labels in the data are balanced so there is no need to weight the loss, moreover we can directly use accuracy (and not F1-score) in the evaluation phase.

2 Pre-trained embeddings

The usage of Pretrained embeddings comes from the fact that them allow the model to quickly get the semantic meaning of the terms. In this approach I decided to use the Glove [4] 840B embeddings with dimension of 300.

3 Preprocessing

In order to takle the problem of out of vocabulary terms I have decided to do some preprocessing of the sentences. Has been enough to remove punctuation and lowerizing the strings in order to remove the majority of the oov terms from the dataset. The remained oovs have been associated to a single random tensor in the embeddings. Other than removing punctuation and loweirize the text, I decided to replace each target term with a constant string *TARGET_TERM* that is associated to a constant embedding tensor, this in order to try to let the model understand that has to consider more the context rather than that term in order to produce the output.

4 Model

Long Short Term Memory (LSTM) models are commonly used to capture long-term relationships inside a sequence [3]. I decided to use a Bidirectional LSTM (BiLSTM) in order to capture the relationships between the terms of the sentence in both directions.

The model that I am proposing takes as input the two sentences (stored as tensors of indexes) that are associated to the two sequences of embeddings relative to the terms inside the phrases through an Embedding layer. The two sequences of embeddings pass through the same BiLSTM (one sentence per time). From the output of the BiLSTM we take the tensor relative to the target term $term_i$ (where i is 1 or 2 representing the sentence), the forward output of the last term $last_i$ and the backward output of the first term $first_i$. After doing this for both the sentences we have to aggregate the results: since I wanted a network whose result is independent from the order of the sentences to compare I decided to do the following: I concatenate $abs(term_1 - term_2)$, $abs(first_1 - first_2)$, $abs(last_1 - last_2)$, $cossim(term_1, term_2)$, $cossim(first_1, first_2)$, $cossim(last_1, last_2)$, where $abs(x - y)$ is the tensor equal to the difference of x and y where each element is chosen in its absolute value and $cossim(x, y)$ is the cosine similarity of the two tensors. This concatenation than is passed through two fully connected layers. The model can be seen in Figure 21.

5 Training

In order to train the model I used Adam Optimizer with learning rate of 0.0001. In order to avoid overfitting I setup the early stopping with patience of 10 and use both dropout and batch normalization (the latter one helped also in speeding up the training process). Moreover I decided to shuffle the train data in order to help to avoid overfitting. From the image 1 and 2 we can see that the methods described above worked well in avoiding overfitting till epoch 35 when the model started overfitting the train data.

6 Results

Since the dataset is a balanced one, I decide to consider directly the accuracy score, rather than the F1-score. With the lstm model I achieve an accuracy of 0.684 over the development set. We can see from the Figure 3 the confusion matrix and from the Figure 4 the ROC curve and its value. From the confusion matrix we can see that the model is unbalanced towards the False class, this could be due to the fact that the LSTM was not able to generate outputs such that their absolute difference was enough to let the model understand the difference between the two sentences.

7 Conclusions

The baseline approach (the model of the first part of the homework) that I used is very simple: it takes as input the weighted average of the embeddings of each of the two sentences. I decided to use 0 as weight for the target term since is surely present in both the two sentences and 0.1 as weight for the stopwords since them are likely present in both sentences, by doing so I am mostly considering the other terms that should better define the context in which the target term is used. These values for the two phrases are passed through the same fully connected layer (one sentence per time). The outputs of this layer are than merged by doing a element wise product in order to have an output independent with the order of the two sentences. The element wise product than passes through two fully connected layers.

Even though the baseline model is so simple the results are better: this model achieves an accuracy of 0.726. We can see this difference in performance also from the Figure 7 and 8 (that are the results obtained with the baseline model) compared with 3 and 4 (obtained with the lstm model). Other than the higher scores obtained by the baseline model we have also that this simple model is not unbalanced like the lstm one (as we can see from the confusion matrix 7) and is much faster in predictions and train, also if we can see in 6 and 5 that the number of epochs is higher than the number of epochs needed to train the lstm model, but with the baseline model the epochs last less than a minute, while with the lstm model, the epochs last more than 2 minutes.

	Precision	Recall	Accuracy
Baseline Glove	0.724	0.730	0.726
Baseline Fasttext	0.711	0.674	0.700
LSMT Glove	0.719	0.604	0.684
LSTM Fasttext	0.634	0.614	0.630

Table 1: Scores obtained with different classifiers and embeddings

8 Extras

From 12 and 11 we can see the results obtained with the fasttext embedding [2] (with the lstm model). As we can see them are worse than the one obtained with the Glove embeddings but in this case the model is not unbalanced towards the False class like with the Glove embeddings. Also in the case of the baseline model as we can see from 10 and 9 that with fasttext we obtain worse performances. This comparison can be seen easily from Table 1: we can see that with Fasttext both the models have lower accuracy and precision, but in the case of the lstm model, with the fasttext embedding we have an higher recall than the lstm model with Glove Embeddings.

I tried also the models over another english dataset [1]: this dataset has only 6066 entries that are equally divided into True and False elements. From 19, 20, 15 and 16 we can see the training of the baseline and lstm models respectively. From these plots we can see that maybe due to the fact that this dataset is much smaller than the one provided for the homework, the baseline model tends to overfit the train quicker with respect to the same model (with same layer sizes) over the homework dataset. From 18, 17, 14, 13 we can see the confusion matrix and the roc curve of the baseline model and lsmt model trained and tested over the new dataset. As we can see from both the confusion matrices and the table 2, in this case we have not a large difference among the two models for what concerns the accuracy, but as before, the LSTM model is unbalanced towards the False class obtaining an high precision and low recall.

	Precision	Recall	Accuracy
Baseline	0.650	0.659	0.657
LSMT	0.722	0.579	0.667

Table 2: Scores obtained with the two models (with Glove embeddings) over the new dataset

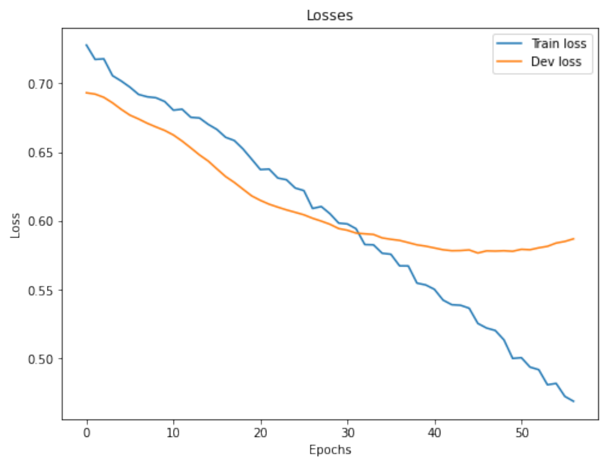

Figure 1: Train and Dev loss of the lstm model over the homework dataset

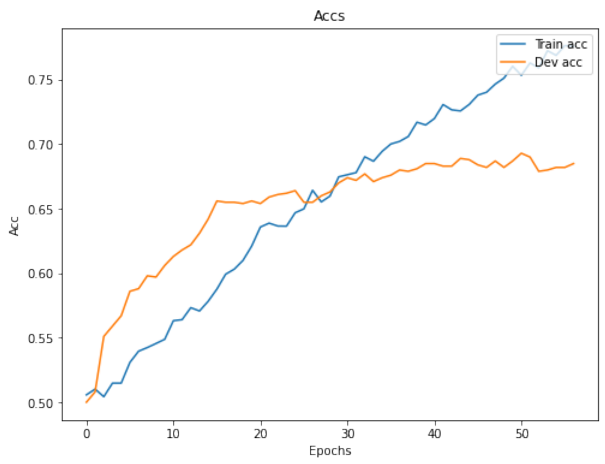

Figure 2: Train and Dev accuracy of the lstm model over the homework dataset

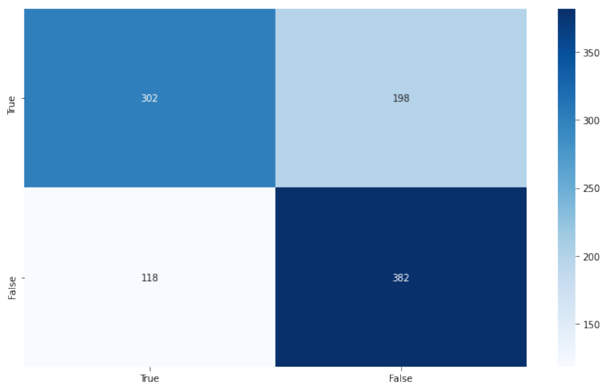

Figure 3: Confusion Matrix of the lstm model over the homework dataset

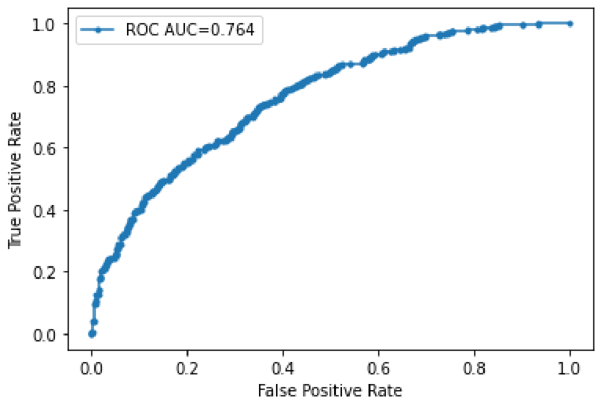

Figure 4: Roc score of the lstm model over the homework dataset

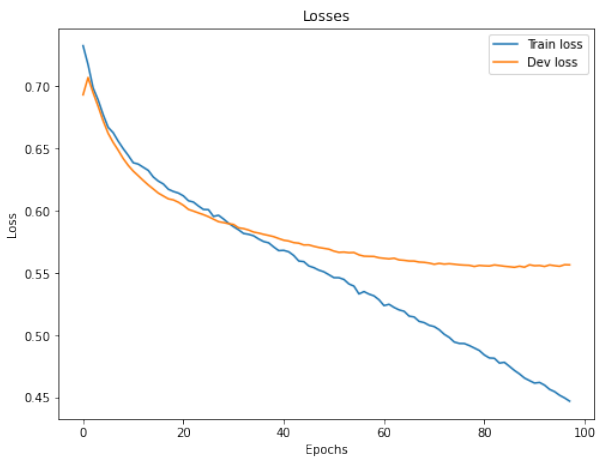

Figure 5: Train and Dev loss of the baseline model over the homework dataset

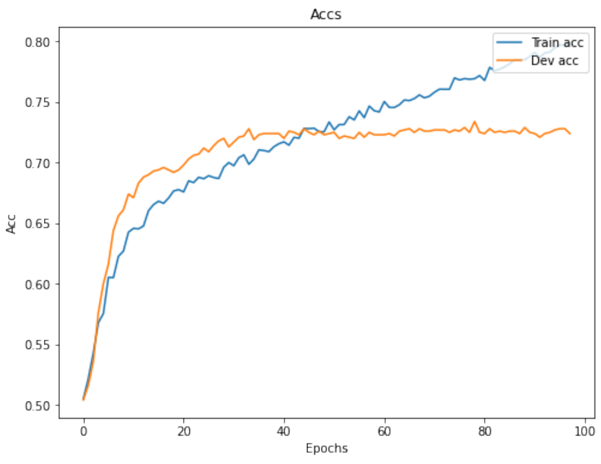

Figure 6: Train and Dev accuracy of the baseline model over the homework dataset

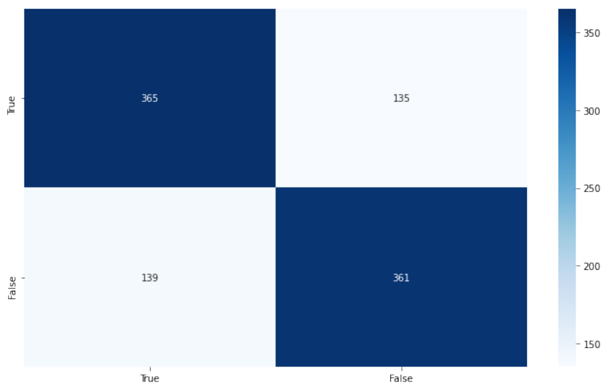

Figure 7: Confusion Matrix of the baseline model over the homework dataset

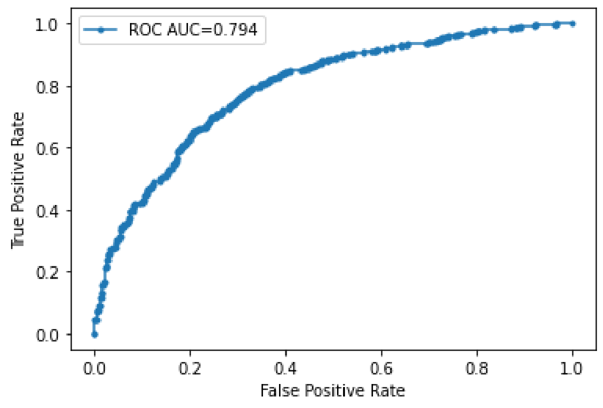

Figure 8: Roc score of the baseline model over the homework dataset

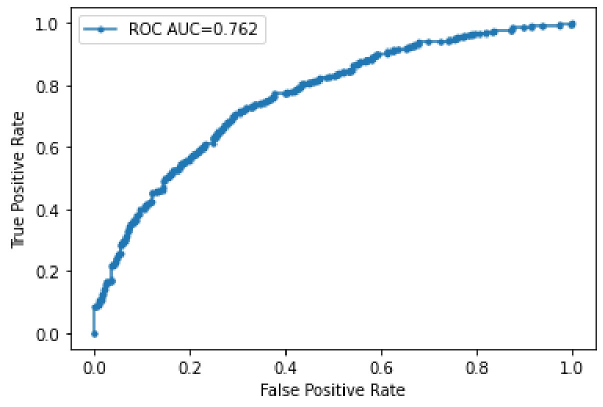

Figure 9: Roc score of the baseline model over the homework dataset with fasttext embeddings

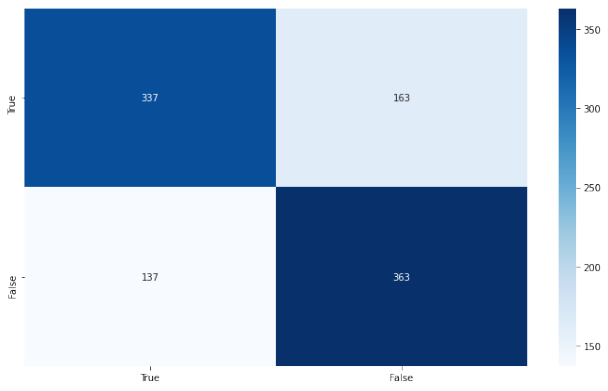

Figure 10: Confusion Matrix of the baseline model over the homework dataset with fasttext embeddings

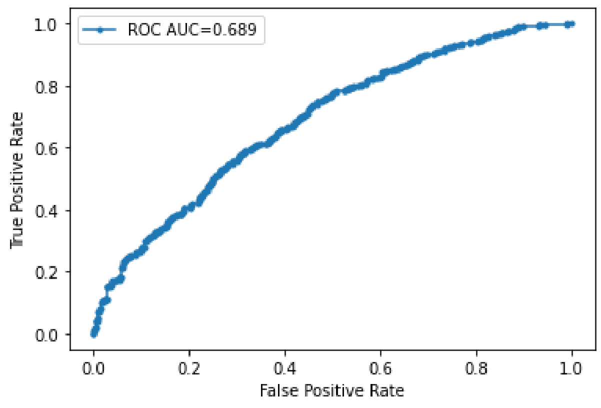

Figure 11: Roc score of the lstm model over the homework dataset with fasttext embeddings

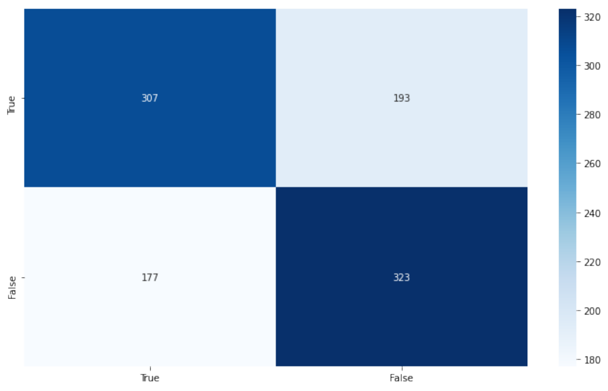

Figure 12: Confusion Matrix of the lstm model over the homework dataset with fasttext embeddings

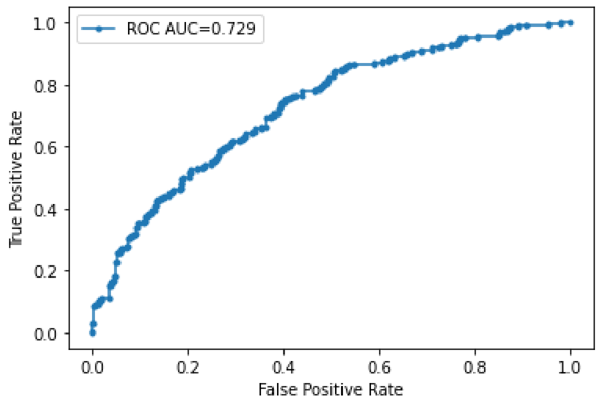

Figure 13: Roc score of the lstm model over the new dataset

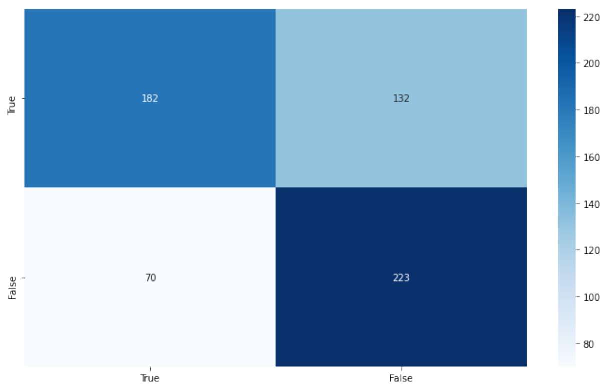

Figure 14: Confusion Matrix of the lstm model over the new dataset

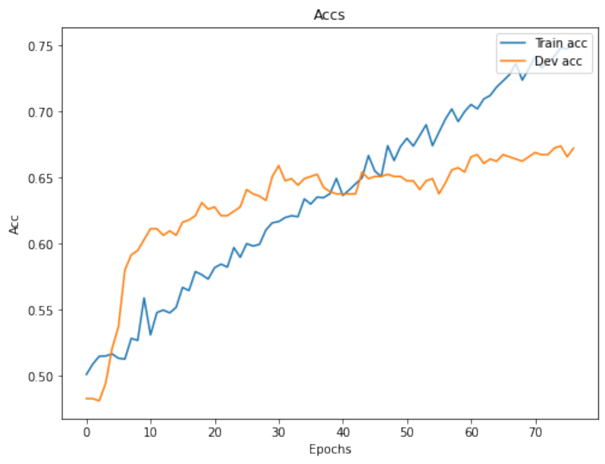

Figure 15: Train and Dev accuracy of the lstm model over the new dataset

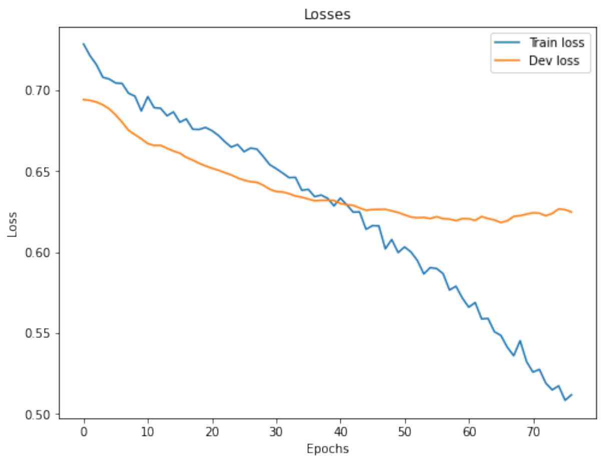

Figure 16: Train and Dev loss of the lstm model over the new dataset

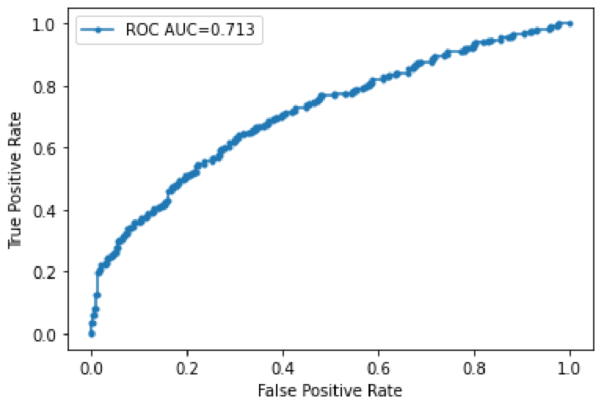

Figure 17: Roc score of the baseline model over the new dataset

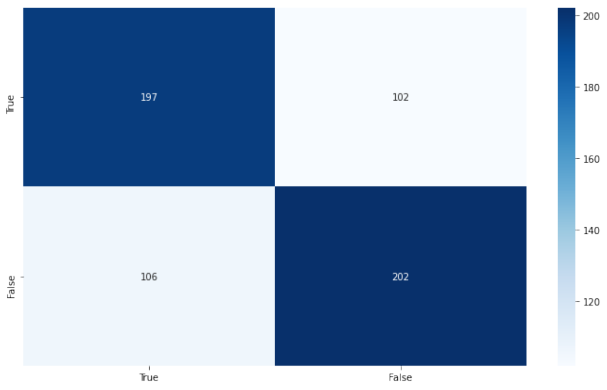

Figure 18: Confusion Matrix of the baseline model over the new dataset

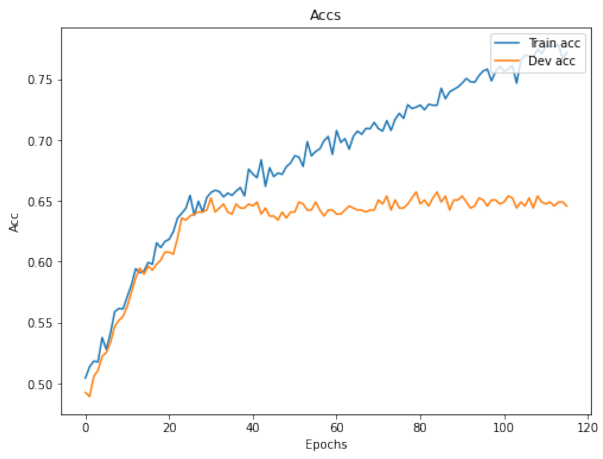

Figure 19: Train and Dev accuracy of the baseline model over the new dataset

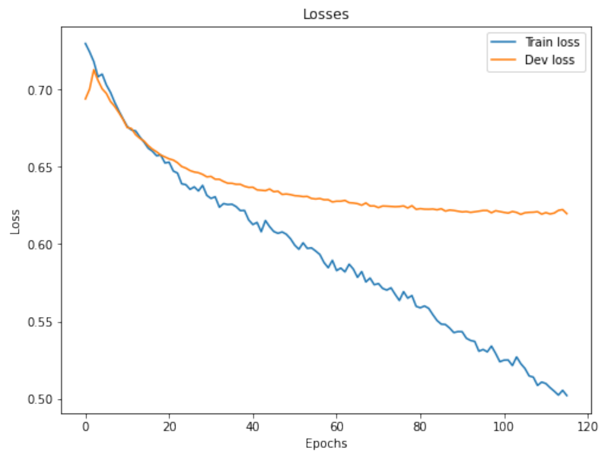

Figure 20: Train and Dev loss of the baseline model over the new dataset

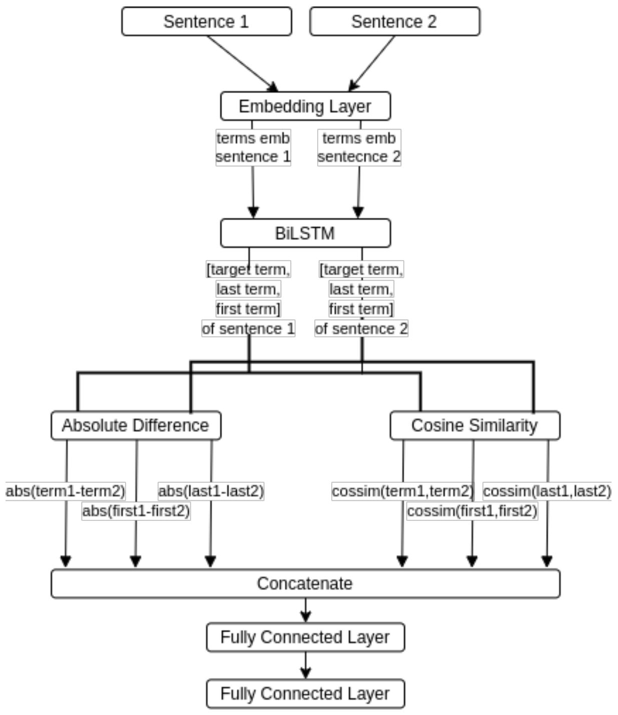

Figure 21: Image of the lstm model

References

[1] Jose Camacho-Collados Alessandro Raganato, Tommaso Pasini and Mohammad Taher Pilehvar. Xl-wic: A multilingual benchmark for evaluating semantic contextualization. *In Proceedings of EMNLP*, 2020.

[2] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

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