Homework 1: Word-in-Context Disambiguation

Mattia Pannone 1803328

1 Introduction

In all different languages there are many words where a same word inserted in sentences with different contexts can assume a different meaning. For two interlocutors (intelligent enough) to distinct the meaning of a word and in particular of the context in a speech is a very fast process, sometimes thanks also to the mode to talk (such as expression, intonation). In a machine it is different, we need methods to recognize words and to understand context only from other words, which are represented with some other methods. In this Word-in-Context Disambiguation homework, there are two difference sentences with a particular same word (called target word) and the task is to classify if a target word have the same meaning for each tuple of sentences. I will be show the methods I use to: transform the words, organize such words, train a model and try to improve its performance, confronting different methods to deal with model and data organization.

2 Word Embedding

As I said, each word must be represented with some methods and in a way to let the machines to deal with it. So first thing I done is to create a vocabulary, where for each word correspond a vector with N elements. The words of a language are really many, so organize this words is a very long and difficult task, so I used a pre-trained word embedding, that is an embedding environment trained previously with some techniques on a large amount of data with as a result a dictionary of word with the different vector. This word embedding are very useful for task like this one. I used GloVe (Global Vector for word representation) made by Stanford University, in particular I used a dictionary of 400K words with a representation of the words with a 100-dimensional vector each one. So I used this GloVe file to create a vocabulary, where each word is represented by a tensor (the vector converted in tensor), this is useful for the next organization of data and for a readable input of the next neural network model.

3 Organization of Dataset

Given the input set, I elaborated and transformed it in a way that it could be trainable from a model. First I separated the two different sentences and replaced the target word of each sentence with the costant '¡TGT¿', so I can find it in a easy way to deal with it in different ways for all sentences. Than I created a list of tuple (Sentence1, Sentence2) and from it I obtained a new list of tuple whit the corresponding tensor for each word using the previously vocabulary. Now data are in a form more usable to manage. With this approach could be some problems, for example if a word is not in the vocabulary. A simple resolution could be ignore that words and compute result only on the present word. However I chosen to replace that words with a mean of each other (in the same sentence), this because if I don't know where is one or more words in the space, replacing them with an average point won't change much of the final sentence in the space (a point computed as an average of all words of the sentence). Another thing I done is to replace the target word with a constant, in particular with a tensors of all zeros, for two motivations: first, if some target words is not in the vocabulary, some sentences would be treated differently, but my scope was to use a unique strategy to treat all sentences, and so all target words, in the same ways; second a all zero tensor is not equal to ignore the words, because in the final average them will be counted. The final set is computed as a difference between two tensors, each tensor is, as I said, the average of the tensors corresponding each word of the same sentence, for each of tuple of sentences. The average of words is to compute a reference point around all words that represent a sentence. The difference between the two reference points of the sentence is to reduce the value of the final tensor (we will see that this is more performing than a sum). Each final tensor is

100-dimensional (deriving from the chosen Glove that represent each word with 100-dimensional vector) and these represent the 100 features where will be trained a model classifier. Oss.: Also label are transformed to be trainable, since they are ’True’ or ’False’, I created a tensor with ones if ’True’ and zeros if ’False’

4 Model Classifier

The model classifier I chose to classify sentences is a neural network model with linear layer of neurons, in particular a model with: an input layer, two hidden layer and the final output layer with a decreasing number of neurons, form 100 (input features); after each layer there is the activation function ReLu except the last one, after the last layer there is a sigmoid function which return a probability between zero and one, in the forward step I transform this probability rounding to the nearest integer, so it will return ’True’ if output is near one (¿= 0.5), ’False’ otherwise. Since it is a binary classification task, I used the Binary-Cross-Entropy cost function and the optimizer Stochastic Gradient Descent with learning rate = 0.01 to compute the loss.

5 Train and best performance

I splitted the train set provided into training set (90 per cent) and validation set (10 per cent) to evaluate the model during its training. This above described model is the last one I used after different evidence with different hyperparameters, since I thought it is the best in performance and time of training. With this model I reached 68 per cent of accuracy with 250 epochs, we can see in Fig. 1a and 1b the trends of loss and accuracy, in Fig. 1c the confusion matrix and in Fig. 1d the classification report. With this model I tried the Adam optimizer with learning rate = 0.001, without improvements. With this same model trained with more epochs I obtain an overfitting situation, with the accuracy that goes down (Fig. 2a,2b,2c,2d). This model is used also with a different data organization, trying with sum between tensors instead of difference, in two ways: with sum where which tensor was the average of all tensors in that sentence, (graphs in Fig. 3a,3b,3c,3d), we can note that the learning is slower, to reach a better result it is necessary increase number of epochs, however it not reach anyway the accuracy of previous approach. Other way I tried is with the sum where each tensor was the sum of of all tensors in that sentence (graphs in Fig. 4a,4b,4c,4d), in this case we can note a very irregular learning whit as result an accuracy very slow with respect to the first model. I tried other methods (not documented) changing for example number of hidden layer, number of hidden neurons, loss functions, optimizer, learning rate, number of batches of data (fixed at 64 in my model), anyway without reaching better results.

6 Conclusion

In this homework I tried to implement and train a model that it was able to recognize if a same word in two different context has the same meaning. First important task to do this is organize and elaborate well dataset to take the optimal features that are useful for the classifier model, in order to improve my approach, it should be useful have a vocabulary with many other words to soccumb to the miss words and try other different methods to aggregate data aside from sum and average (for example distance between words). Other thing to do to improve performances is change the classification method using more sophisticated methods such as Recurrent Neural Network and LSTM.

References

https://nlp.stanford.edu/projects/glove/

https://pytorch.org/docs/stable/index.html

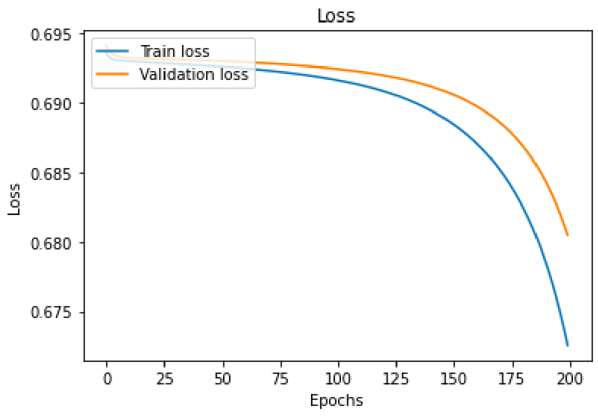


Figure 1: Fig. 1a

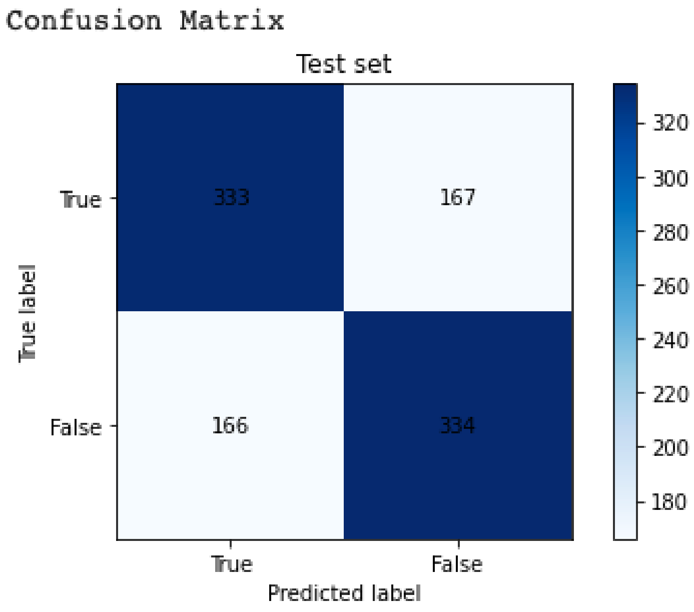


Figure 4: Fig. 1d

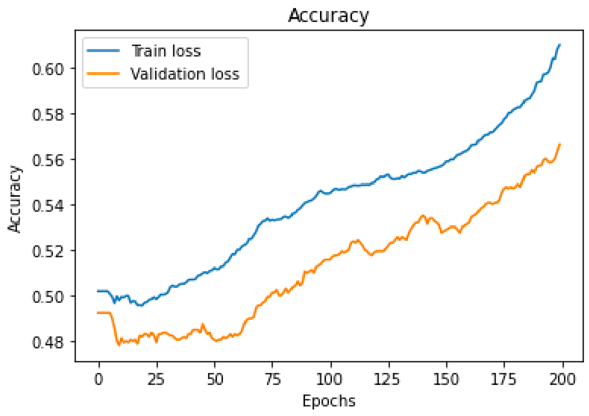
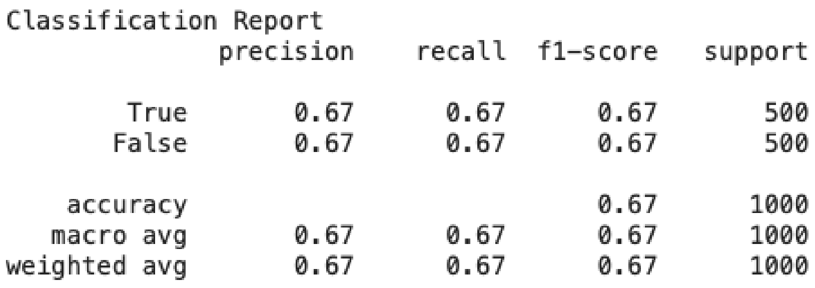


Figure 2: Fig. 1b



```
Classification Report
              precision    recall  f1-score   support

        True       0.67      0.67      0.67       500
       False       0.67      0.67      0.67       500

    accuracy                           0.67      1000
   macro avg       0.67      0.67      0.67      1000
weighted avg       0.67      0.67      0.67      1000
```

Figure 3: Fig. 1c

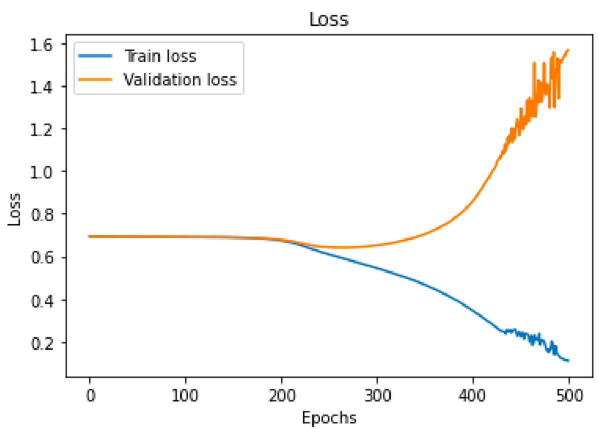


Figure 5: Fig. 2a

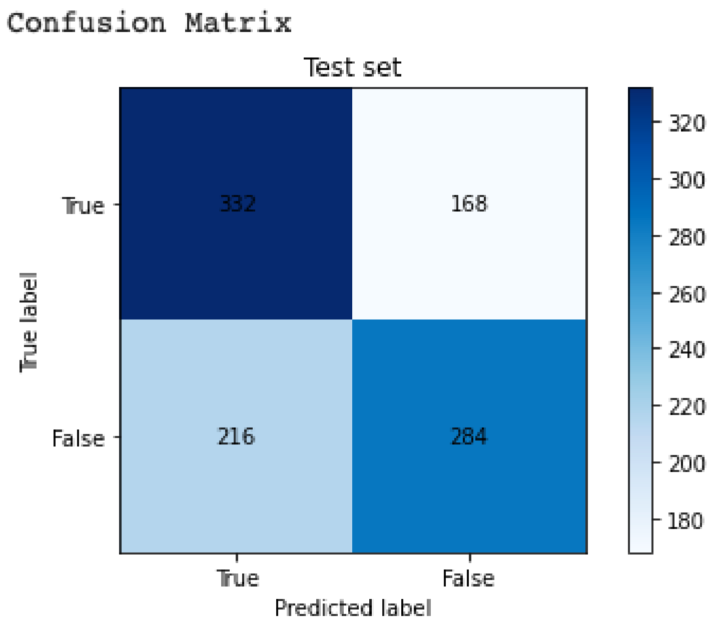


Figure 8: Fig. 2d

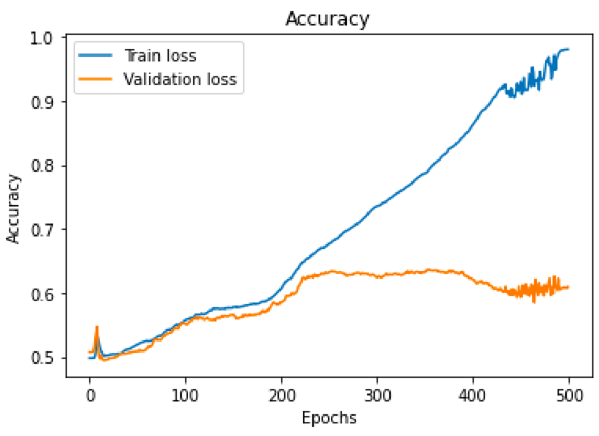
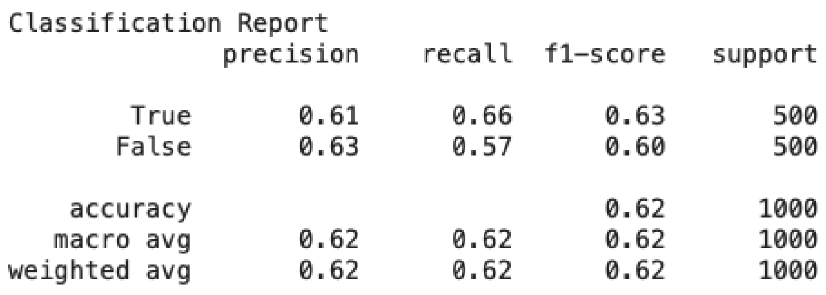


Figure 6: Fig. 2b



```
Classification Report
              precision    recall  f1-score   support

        True       0.61      0.66      0.63       500
       False       0.63      0.57      0.60       500

    accuracy                           0.62      1000
   macro avg       0.62      0.62      0.62      1000
weighted avg       0.62      0.62      0.62      1000
```

Figure 7: Fig. 2c

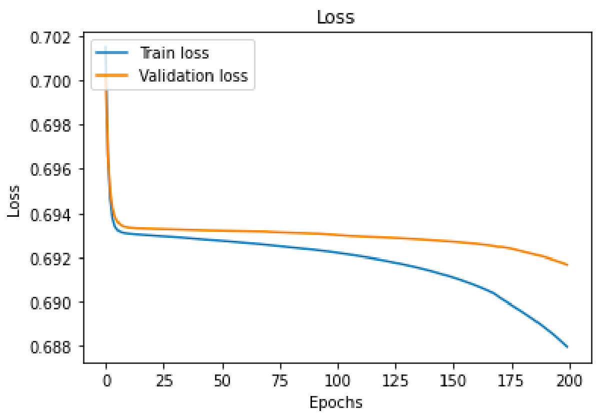


Figure 9: Fig. 3a

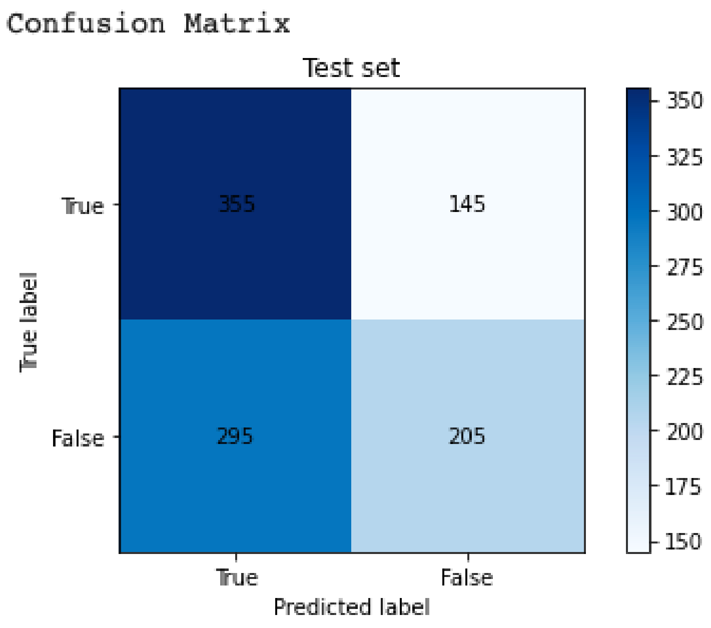


Figure 12: Fig. 3d

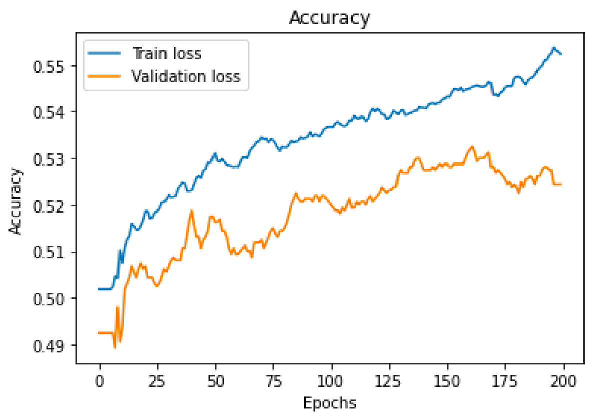
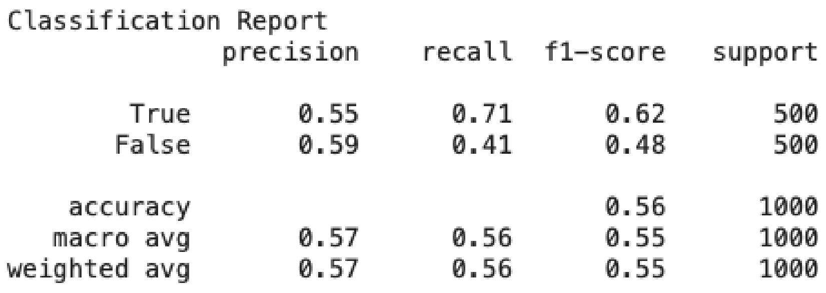


Figure 10: Fig. 3b



```
Classification Report
              precision    recall  f1-score   support

        True       0.55      0.71      0.62       500
       False       0.59      0.41      0.48       500

    accuracy                           0.56      1000
   macro avg       0.57      0.56      0.55      1000
weighted avg       0.57      0.56      0.55      1000
```

Figure 11: Fig. 3c

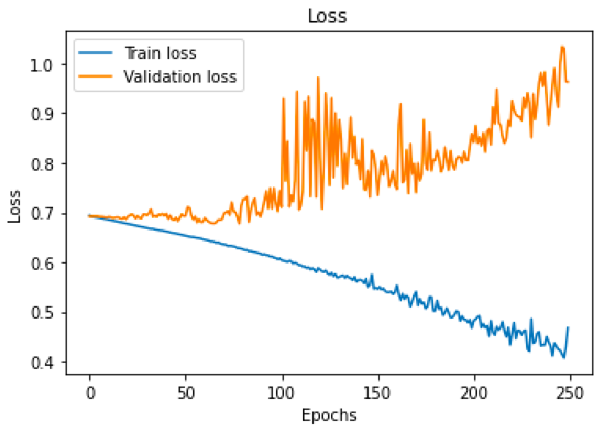


Figure 13: Fig. 4a

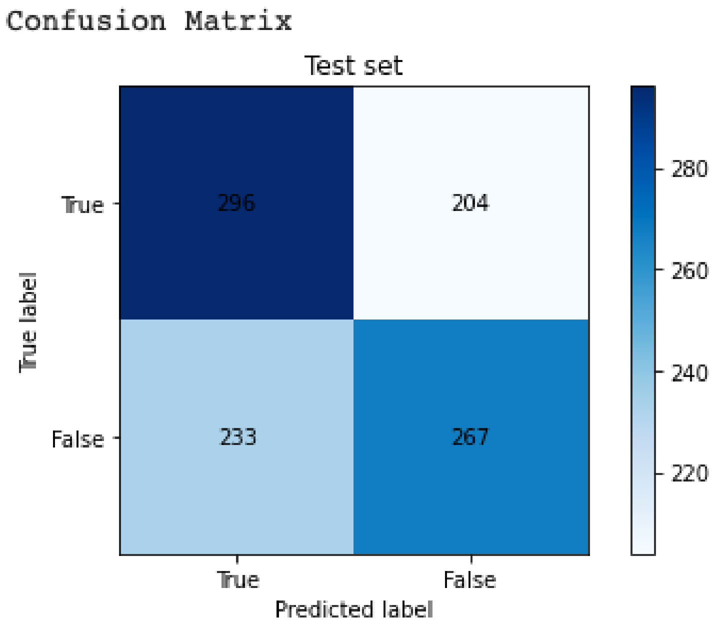


Figure 16: Fig. 4d

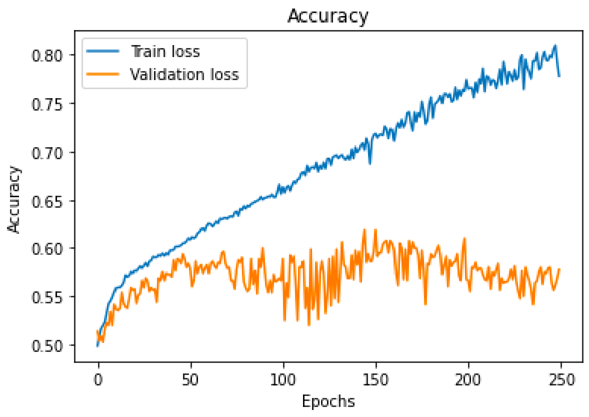
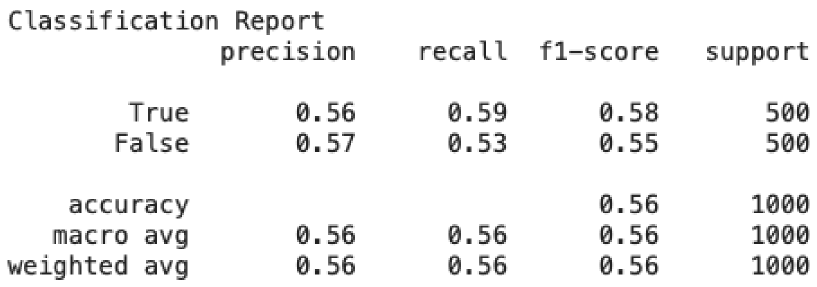


Figure 14: Fig. 4b



```
Classification Report
              precision    recall  f1-score   support

        True       0.56      0.59      0.58       500
       False       0.57      0.53      0.55       500

    accuracy                           0.56      1000
   macro avg       0.56      0.56      0.56      1000
weighted avg       0.56      0.56      0.56      1000
```

Figure 15: Fig. 4c