Report Homework 1 MNLP

Fabio Fraschetti
1834942
fraschetti.1834942@studenti.uniroma1.it

Abstract

In this report I will show how I complete my homework. The report is composed by 6 paragraphs: Introduction, Method, Setup, Experiments, Results, Conclusions. In Introduction I will present the problem, in Method, how I approach the task. In Setup I focus on the architecture that I used. In Experiments, the various tests that I did for setting up the best values for the hyperparameters. In Results I will show the accuracy and F1 score that my model obtained and comment the confusion matrix.

1 Introduction

In this homework we need to develop a Neural Network that is able to classify the words of a sentence in 11 labels that are shown in Table 1 with their number of presence and percentage, taken from Test file. Starting from this data we can observe that the base line to surpass is 90.44% that correspond to simply set all the labels values to O.

Label	Presence	Percentage
O	46775	90.44
B-SCENARIO	629	1.21
B-SENTIMENT	319	0.62
B-CHANGE	1303	2.52
B-ACTION	2257	4.36
B-POSSESSION	275	0.53
I-SCENARIO	47	0.09
I-SENTIMENT	2	0.003
I-CHANGE	62	0.12
I-ACTION	50	0.09
I-POSSESSION	1	0.001

Table 1: Labels and respective number of presence and percentage in Test.jsonl file

2 Method

To complete this task I decided to use a particular type of RNN, the BiLSTM. Once I developed this module I set the hyperparameters and I started optimizing the weights on Train file, valuating the obtained results (accuracy and F1) on Dev file. Once I reached the higher accuracy possible for 10 epochs I determine the hyperparameters to use for a longer training. At the end of the training I tested the final model on a Test file in order to see the final accuracy.

3 Setup

The first thing to do is to read the jsonl file and store it. Now that I got all the words and labels, I created a vocabulary where the key is the word and the value is the word2vec index of that word, in this vocabulary I store only the words that are in word2vec because only those words can be processed from the embedding layer. The other words are set to 'UNKNOWN'. Once that the vocabulary has been created, I built the Neural Network module. First of all I instantiate the embedding layer which permit to transform the semantic and syntax relations of the words in geometric relations of vectors of 300 dimensions, I used a pretrained embedding, word2vec-google-news-300. Then I put a dropout layer that guarantee a better robustness of the Neural Network to the Overfitting. The next layer is the BiLSTM layer, that is an LSTM that reads the sentence starting from the beginning and from the end. Bidirectional because performs better with long sentences since after some words the memory of the first words decreases. So reading the sentences with a different order guarantee a longer memorization of the words at the edges of the sentences, in same cases could be important. Then I put another dropout layer. In the end a classifier layer where I instantiate a logsoftmax function. I choose logsoftmax because it permit to improve performance and gradient optimization. Once the Neural Network has been defined I developed the

Trainer, where I choose stochastic gradient descent (SGD) as optimizer and NLLLoss as loss function. I choose SGD for a faster calculus and NLLLoss because is used for multi-class classification problem and I can use it because I have a logsoftmax in the end of the Neural Network. In order to evaluate the Test I used 3 results: Accuracy, F1Score and confusion matrix.

4 Experiments

The hyperparameters in my project are: embedding dimension, hidden dimension, number of layers, dropout, learning rate. The embedding dimension is set to 300 because is the number of dimensions of vectors of the pretrained model that I used. Hidden dimension is the number of features of the hidden state. I started with 64 and then 128 and saw that the accuracy were increasing and the time for training too. Then I decided to set Hidden dimension to 350 because I saw very big improvement when I set a number grater then embedding dimension. I can't set a too big value otherwise the time for each epoch reaches 4 minutes. That is too much compared to the 2 minutes with hidden dimension set to 350. The number of layers that I choose is 2 because I saw that adding other layers isn't useful. I set dropout at 0.25 because it should be a good balance compared to the size of the network. The learning rate is set at 0.1 because I tried with 0.01 and 0.001 but with this parameters the accuracy increases too slowly.

5 Results

I stored the results in 2 images. In the first one we can see in blue the training accuracy and in red the accuracy on the Dev in 45 epochs. We can see from this graph that at the beginning the accuracy of the Dev (0.936) was higher then the train (0.931) at the 15th epoch this 2 accuracy hit the same value and after this point the accuracy of the Train keep increasing and the accuracy of the Dev grows little, reaching the highest value at 27th epoch (0.9457) and then gradually goes into overfitting. In the second figure there are the same epochs but with F1 score. Here we can see that the two functions meet at the 20th epoch and got the higher value in the 27th epoch (0.7457) as in the first figure. The results that I obtained in the Test file are accuracy (0.9495) and F1 score (0.7010). In the third and fourth figure there are the Confusion Matrices on the Test file and Train file. I add Table 2 for a better comprehension of the percentage of Figure 4. In the Test confusion matrix we can observe that the labels that perform better respect to the number of presence are: O, B-CHANGE, B-ACTION. Then we have I-CHANGE and I-POSSESSION that perform well but are only 63 labels in total that are quite few. All the labels of I-ACTION and I-SENTIMENT are wrong, the Neural Network changes I-ACTION for POSSESSION type and I-SENTIMENT for I-SCENARIO. In general there is a lot of noisy in the I-POSSESION label. it seems that if it is uncertain on the result, try to associate it to I-POSSESION label. In fact I-POSSESION appears 9501 times when all labels are 51720 that is the 18.4%. Which is an high percentage for an I label. We have, more or less, the same behavior in Train confusion matrix.

Label	Presence	Percentage
O	463402	90.35
B-SCENARIO	6405	1.24
B-SENTIMENT	3024	0.58
B-CHANGE	13256	2.58
B-ACTION	22458	4.37
B-POSSESSION	2711	0.52
I-SCENARIO	505	0.09
I-SENTIMENT	24	0.004
I-CHANGE	583	0.113
I-ACTION	457	0.089
I-POSSESSION	30	0.005

Table 2: Labels and respective number of presence and percentage in Train.jsonl file

6 Conclusions

The results that I have obtained (accuracy: 0.9495 and F1 score: 0.7010) surpass the base line, maybe with another architecture, like BiLSTM CRF the Neural Network could have performed better.

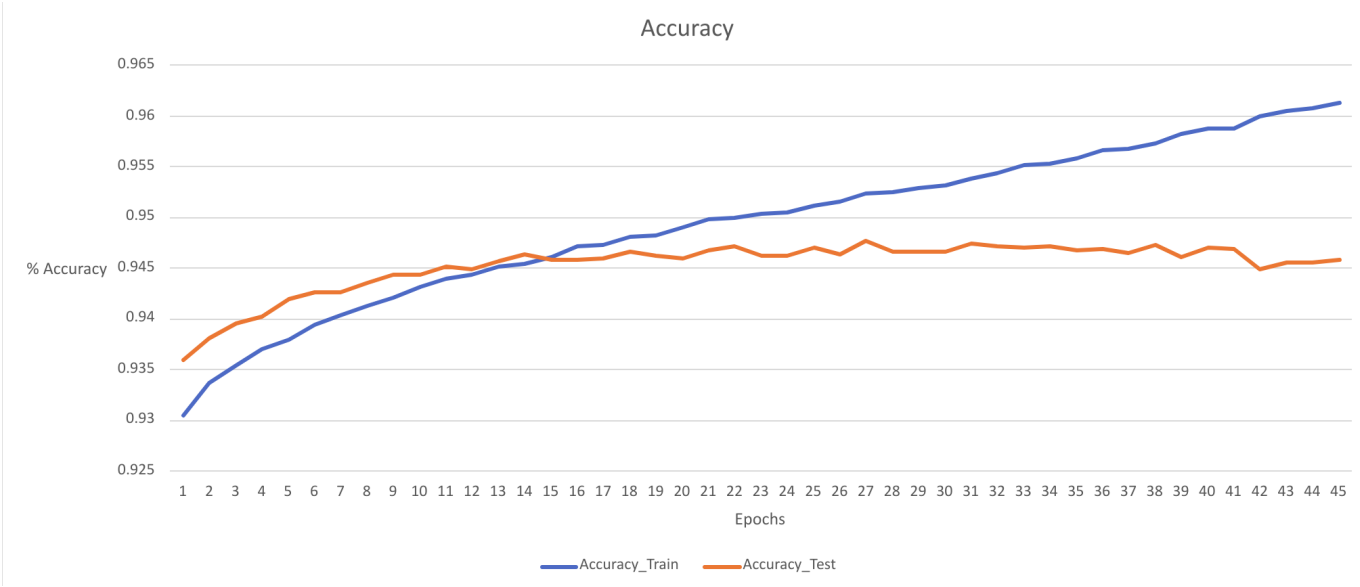

Figure 1: Accuracy trend

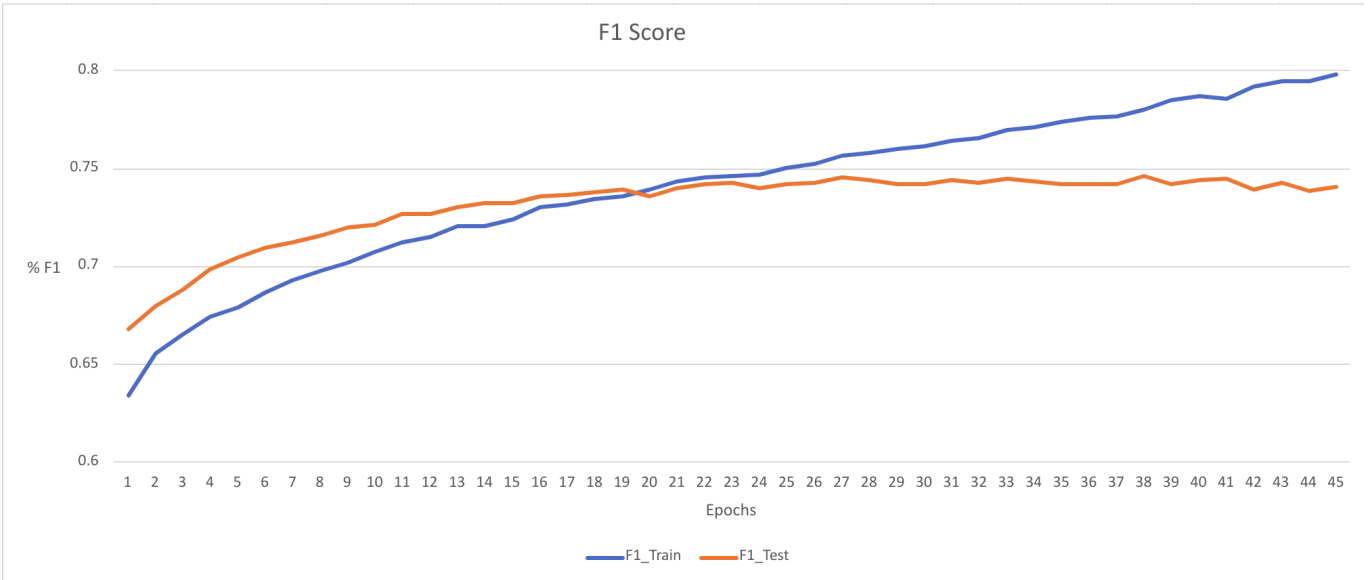

Figure 2: F1 trend

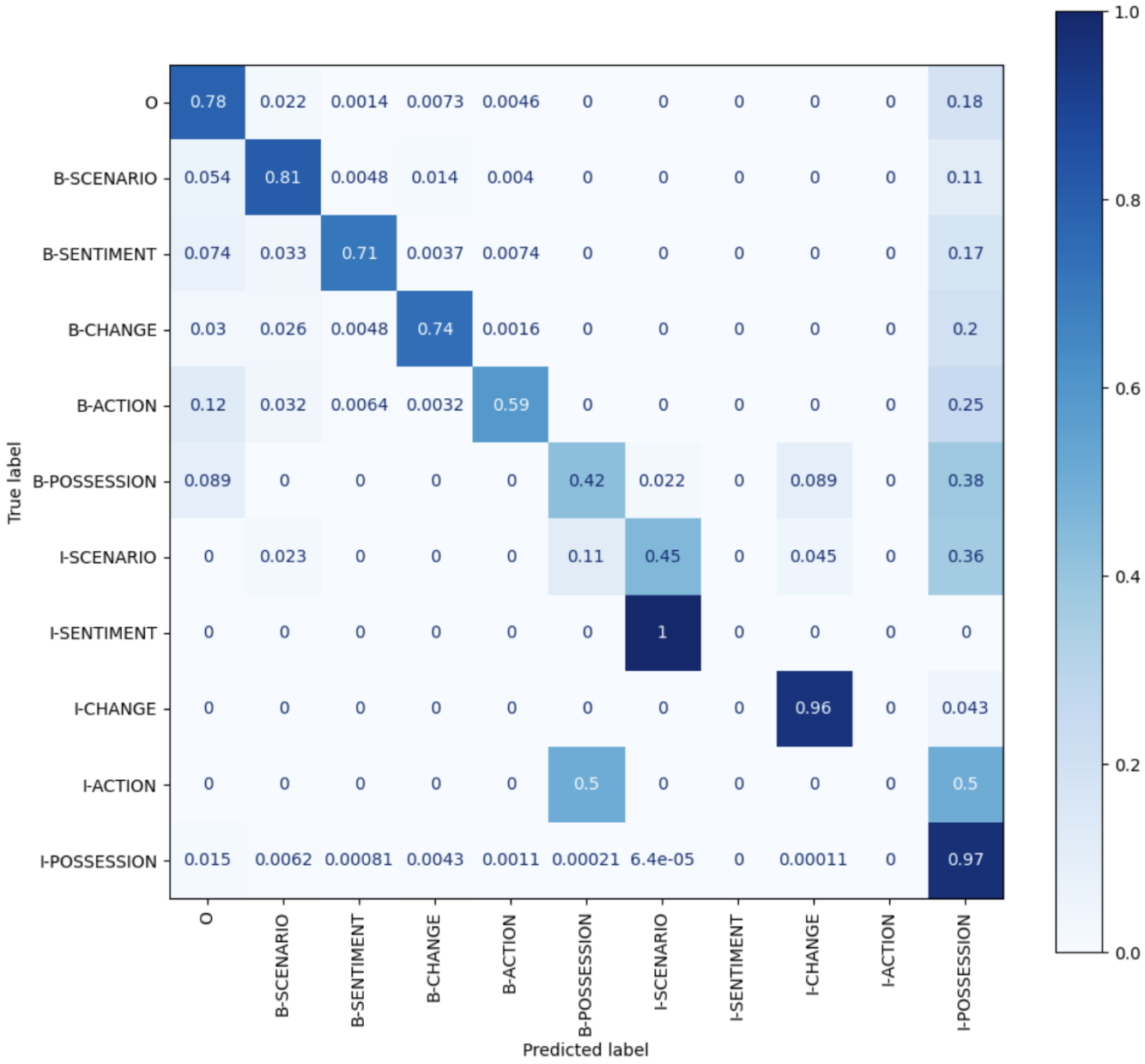

Figure 3: Confusion Matrix on Test file

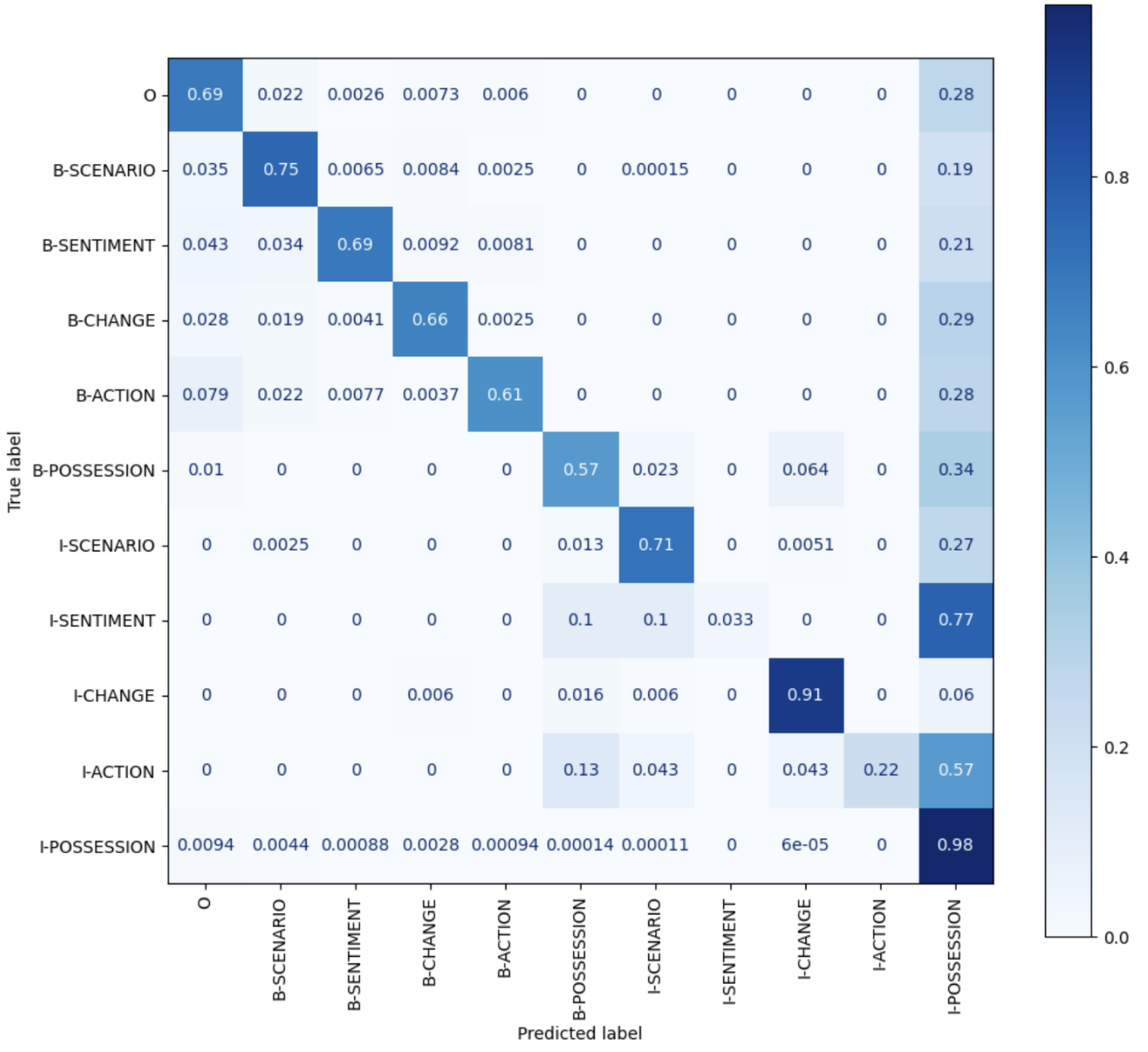

Figure 4: Confusion Matrix on Train file