Assignment 2 - Fact checking using neural architecture

Mattia Bertè* **Simone Giordani*** **Omar G. Younis***
{mattia.berte, simone.giordani2, omargallal.younis}@studio.unibo.it

Abstract

In this report we present our solution to the Fact checking problem. We faced and solved the problem using an Embedding layers for the creation of the sentences embedding and two fully connected layers for the classification. The dataset used for training is the public available *FEVER dataset*, while concerning the words embedding we used the pretrained *GloVe* embeddings, while for the sentences embedding we tried different strategies with RNN, MLP and BOV. In this situation we have to face multi-input classification, so we tried also different strategies for merging claim and evidence. Once we found the best architectures combinations we tuned the hyperparameters and analyzed their results. All the code and the trained models can be found on GitHub[1].

1 Problem description

Fact checking is the process of deciding if a claim is supported or refuted by the evidences in the dataset. The dataset is composed of 121740 pairs of claims and evidences, one claim can have more than on evidence, and each entry is labelled as 'SUPPORTS' or 'REFUTES' and has an ID that refers to the claim.

1.1 Dataset analysis

In the train dataset the classes are not balanced, in fact 75% of the entries has the SUPPORTS label, so we applied random undersampling to balance them. At the end remains 64702 entries in the train set.
Since the evidences are taken from different sources it is necessary a preprocessing step to clean the data, in particular to delete a lot of characters outside of the English alphabet, like the phonetic symbols, and to delete some formatting elements, such as tags after the end of the sentence.
After this first step of preprocessing, we managed some of the OOV terms (with respect the vocabulary of the GloVe Embedding), correcting number and typos or mispelled words (more details about this in the report for the Assignment 1).

	Unique Words	**OOV Before processing**		**OOV After processing**	
Train set	28828	2370	8.22%	179	0.62%
Val Set	8576	328	3.82%	35	0.41%
Test Set	9590	360	3.75%	41	0.43%

Table 1: *OOV processing*

2 Model description

This problem belongs to the categories of the multi-input classification problems. So the architecture is composed of an encoding part fro which we tested 4 different strategies:

- Encode token sequences via a RNN and take the last state as the sentence embedding;
- Encode token sequences via a RNN and average all the output states.
- Encode token sequences via a simple MLP layer;

[1] https://github.com/younik/fact-checking

- Compute the sentence embedding as the mean of its token embeddings (bag of vectors);

Until this step the two sentences as input, claim and evidence, are processed independently. Before passing the sentences to the classifier we have to merge them and also at this point we tried different techniques:

- concatenation of the two sentence embeddings;
- sum of the two sentence embeddings;
- mean of the two sentence embeddings;

Then we implemented also a simple variation concatenating to the input of a value that represent the cosine similarity between claim and evidence embeddings. After the implementation of all the models we tuned them in order to see which was the best combination of hyperparameters and architecture combinations. RNN with the average of the output states shows the best performances and less overfitting on the training data. Regarding the merging method, we found out that the differences between the three methods are very small. That is because the mean is just the sum divided by 2, and the sum can be easily derived from the concatenation in the first layer of the classifier. So we choose to use the concatenation method, as it is the more general one.

3 Results

In this section are reported that perfomances of the best architecutre combination tuned on the validation set. On the test data it performs very similar to the validation data, well balanced between the classes.
We also tried to evaluate the model when considering multiple evidences for one claim proceeding with a majority votes. Here the results are very similar to the simple evaluation done before because the dataset presents very little amount of examples with multiple evidences per claim. For example in the test set, the claims that have 4 or more evidences are only 12 (out of 6613).
We tried, as a simple extension, to concatenate the cosine similarity between the embedded sentences to the input of the classifier

	Macro Average						**Weighted Average**					
	Single Input			**Multi Input**			**Single Input**			**Multi Input**		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
A	66.9%	66.8%	66.7%	66.6%	66.5%	66.4%	66.9%	66.8%	66.7%	66.7%	66.5%	66.4%
B	67.1%	66.9%	66.9%	67.1%	66.9%	66.8%	67.1%	67.0%	66.9%	67.0%	66.9%	66.8%

Table 2: *Training report* (A: RNN sentence embedding and concatenation merging, B: base model + cosine similarity)

4 Error analysis

For the error analysis step we printed some of the misclassified sentences in order to detect possible recurrent pattern. We found out that some problems are related to words that do not belong to english (evidence ID 145789 is written in russian), other errors are related to the preprocessing step, in evidence ID 89049 the claim uses *'Dissociative Identity Disorder*, while the evidence uses its acronym DID that is interpreted as a stop word and deleted during this phase. The most common error is due to complex phrases in which to understand the position of the evidence is necessary to have additional knowledge related to that topic (e.g. ID 124295).

5 Conclusion

In conclusion we can say that the cosine similarity can bring little improvement, maybe because the similarity between some parts of the sentences is lost during the sentence embedding step. Instead the evaluation with more evidences maybe can bring to some good results, but this can not be shown using this particular dataset. Some possible improvements, that we have not tried due to limitations in the computation and time resources, can be to apply lemmatization, or to use some better architectures for the network, such as multilayers RNN, LSTM layers, and a deeper classifier.