Assignment 2: Fact Checking

Serban Cristian Tudosie
serban.tudosie@studio.unibo.it

Francesco Vannoni
francesco.vannoni2@studio.unibo.it

Mirko Del Moro
mirko.delmoro@studio.unibo.it

Abstract - Fact Checking is the task in which from a given claim and a set of sentences, an agent has to understand if the claim is supported or not w.r.t. the evidence. This task is well suited for neural network approaches, specifically RNNs have been shown to be effective at it. We show different models trained on the FEVER dataset to explain what works best.

1 OUR APPROACH

1.1 Dataset Cleaning

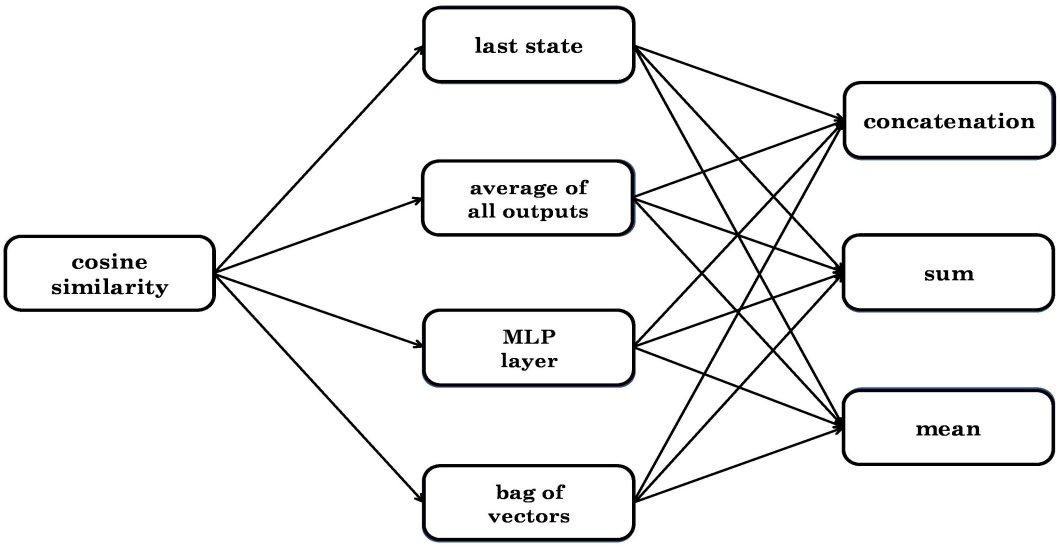
The dataset provided by FEVER contains pairs of claim and evidence set along with 3 labels (Supports, Refutes, NotEnoughInfo). We have cleaned and we have simplified the task by not considering NotEnoughInfo. Then the set of evidences is split in order to have only pairs of claims and evidences. We have removed the punctuation and anything that was enclosed by parenthesis because we think that it was useless and noisy information. We have included in the train set only sentences not longer than 100 words. Then also samples that had the claim or the evidence or the label empty have been deleted.

1.2 GloVe Embeddings and OOV Handling

To get the embeddings we have used GloVe vocabulary but we have to handle the dataset's terms not contained in GloVe.
We have computed OOV terms' embedding by computing a new vector of same dimension but chosen from an uniform distribution. This process must be done in order for train, validation and test. The result embeddings are concatenated to the embedding matrix and the terms are added to the vocabulary. This allows to have an embedding matrix where: if a word is present in both train and test set and not in GloVe vocabulary, the embedding is computed as OOV when the training set is processed, and it is considered as a vocabulary word when the handling of the test set takes place.

1.3 Models and Training

We have computed sentence embedding in order to reduce the token sequence to a single representation and we have encoded token sequences in four different ways: with the RNN last state, with the average of all the output states, via a simple MLP layer and with bag of vectors (mean of token embeddings). For each sentence embedding sentence encoding we have merged claim and evidence sentence embedding and we have worked with three different strategies: concatenation, sum and mean. Therefore we have 12 different combinations of sentence embedding and merging type. For each of these combinations we have computed the cosine similarity metric between the two sentence embeddings and we have concatenated the result to the classification input. This allows us to work with 24 different combinations. We have trained the models for 3 epochs with a batch size of 128 and we evaluate them on the test set with two different evaluations (Simple Multi-Input, Majority Voting). The following DAG summarizes the possible models:



We have chosen to train with Nadam optimizer with a learning rate equal to 10^-3 and use a categorical crossentropy loss with a softmax head for classification. In order to prevent overfitting we have exploited "Early Stopping" monitoring the validation accuracy

2 GENERATORS

We have decided to create two generators: one that produces balanced batches (w.r.t the class labels) for the training of the neural models, and another one that chooses batches sequentially. This is done in order to avoid problems of unbalanced since the training data is very unbalanced.

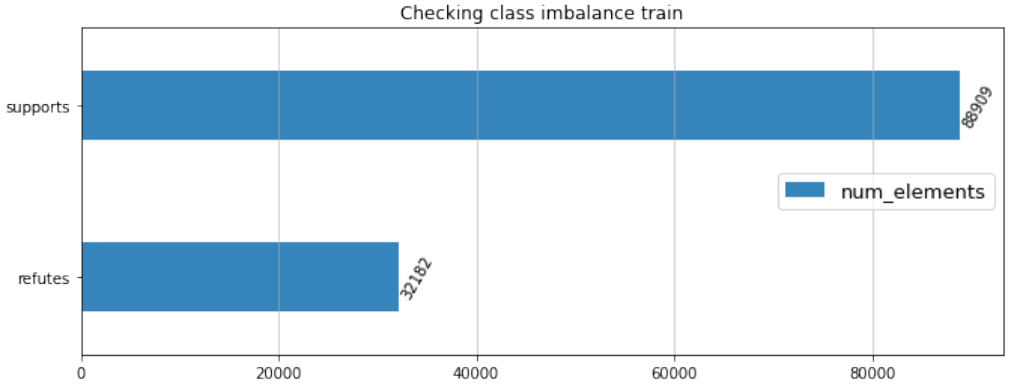


Figure 1: Imbalance of the training set

Both approaches produce batches that do not have duplicates. The generators also have the possibility to dynamically pad claim and evidence sequences allowing better memory performance.

3 RESULTS EVALUATION & ERROR ANALYSIS

Since we have taken a simplistic approach with the dataset proper evaluation has to be considered, because we have split the evidence sets. To evaluate the models we have used two types of evaluation:

1. Multi-Input Classification
2. Majority Voting

The first one is a collection of metrics such as f1 score, accuracy, precision and recall, where we consider each tuple (claim, evidence).

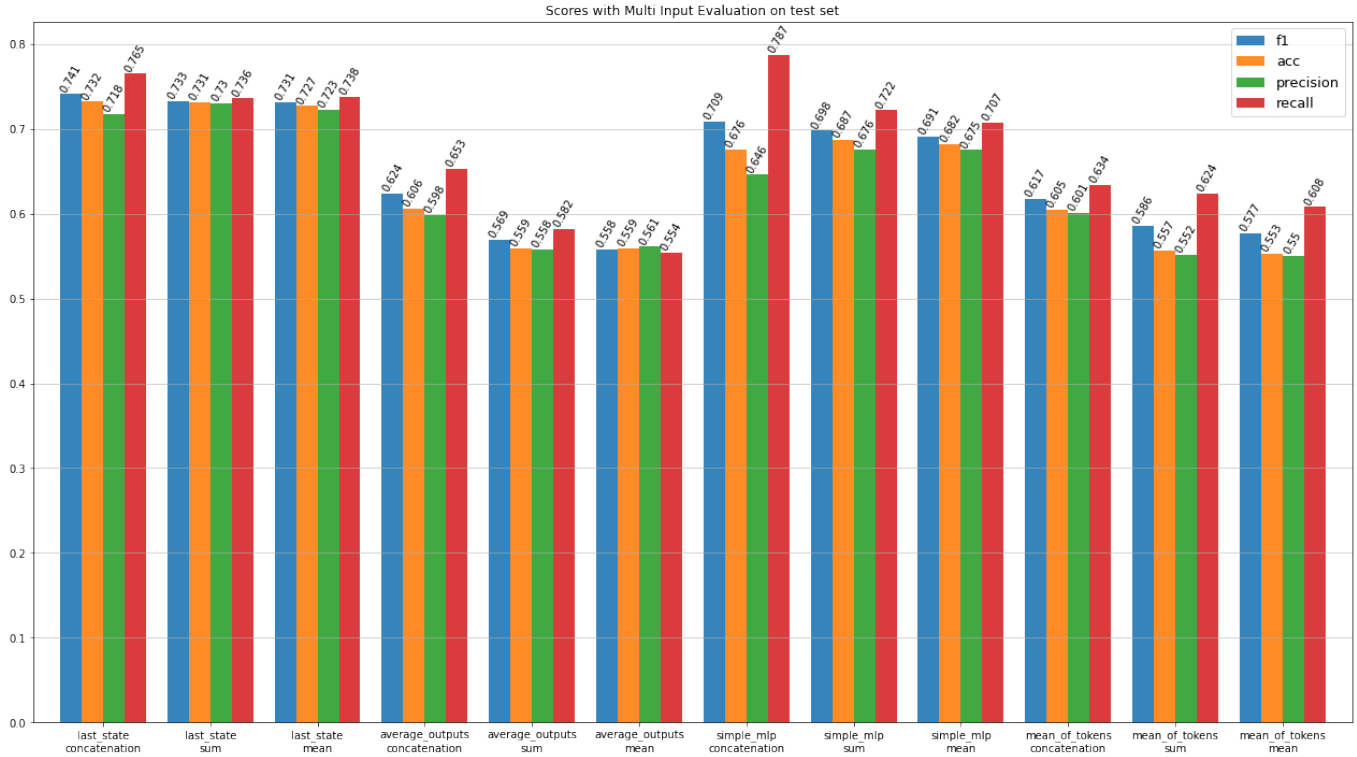


Figure 2: Models Multi-Input Classification Scores

The second one is given by the same metrics but on a different y. For each claim we compute the majority voting between the labels of each evidence in the initial evidence set. In other words the y is smaller and the results are for sure worse and more realistic.

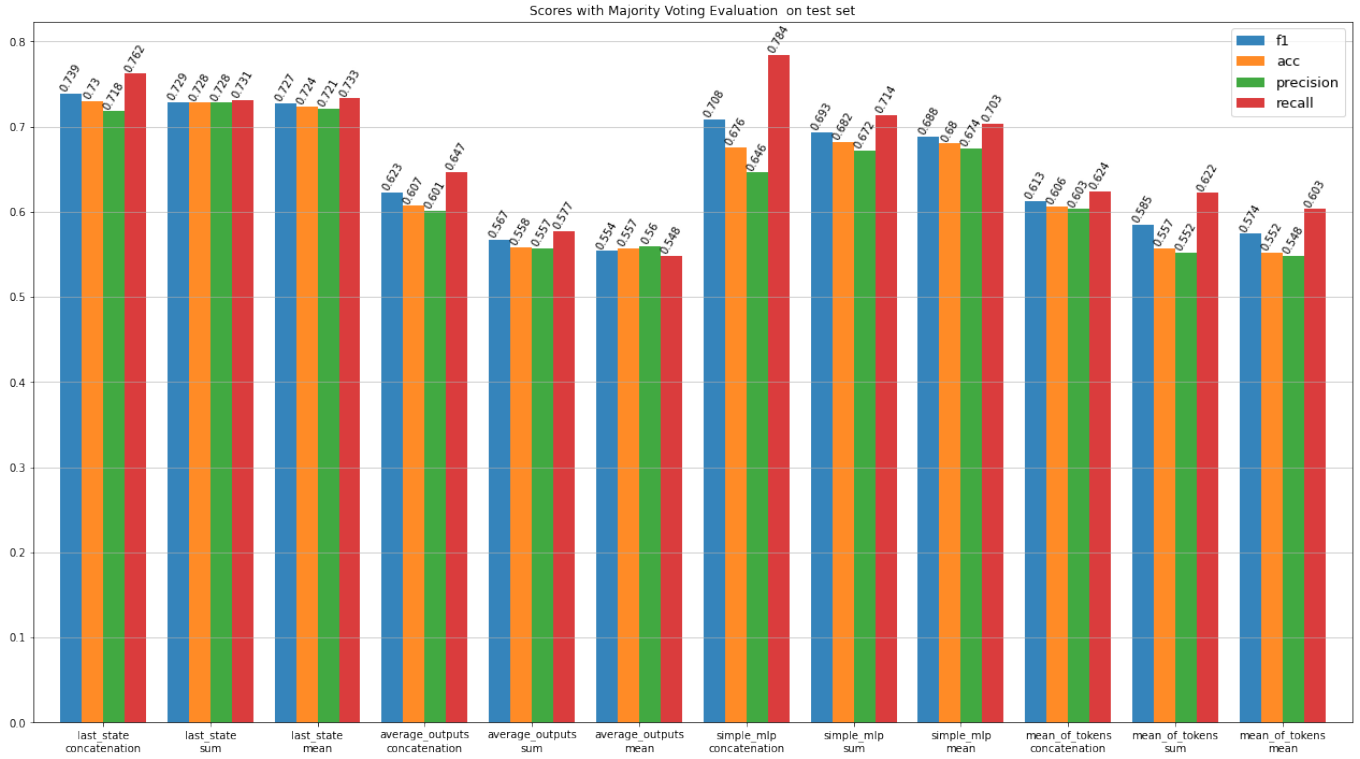


Figure 3: Models Majority Voting Scores (No-Cosine-Similarity)

As we can see the models that implemented the "last_state" technique perform very well with respect to the others. In order to show the differences between the added feature with cosine similarity we propose also the following bar plot only on majority voting.

Figure 4: Models Majority Voting Scores (With-Cosine-Similarity)

Because the task at hand is Fact Checking a big problem would be the classification of a fake news with the "supports" label. This means that overall we are more interested in high precision for our model. In addition to that we expect precision, recall and f1 to be aligned with each other since the batches are balanced. But since the model has a higher probability of seeing the same samples of the "refutes" class some differences can emerge.

Analyzing the differences between the addition of cosine similarity and baseline models we can empirically see that it introduces redundant information that worsens the performance. We can distinguish the influence of cosine similarity between concatenation and the other two techniques. In particular, cosine similarity improves the f1 values when sum and mean are used to merge the sentence embeddings, but at the same time the variance between precision and recall is higher. Since we are more interested on precision we can say that it does not add relevant information for this task. If we take as an example fake news, we can better explain this phenomena, that cosine similarity introduces, by the fact that usually the claim in a fake news is very similar to the evidence. So the sentence embeddings will still be similar and thus having a small cosine distance will confuse the model.

Further possible improvements could regard the size of the dataset and the initial balancing. The first could be tackled by using and/or merging of other dataset. For the latter we think that it can be partially solved by using some data augmentation techniques. In particular, a naive solution could be: combining several claims with randomly chosen evidences to enlarge the refutes class since it lacks lots of samples.

4 EXTRA

Lastly we show the results of a model where the sentence embeddings are created with a multi-head attention module inspired from the encoding part of the Transformer model:

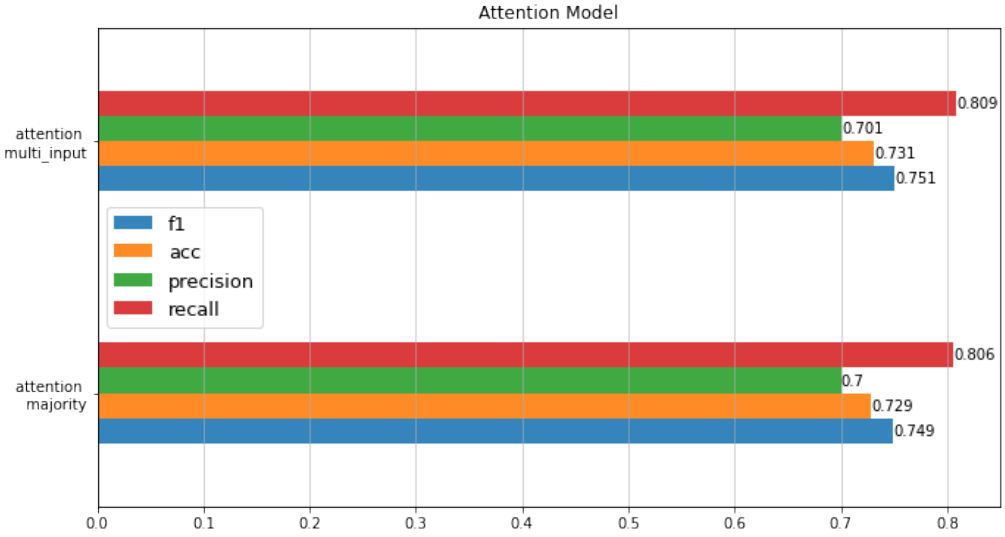


Figure 5: Attention Model

We see that the same phenomena that happened with cosine similarity happens here too. This might be because the attention layer tries to find similarities between claim and evidence, thus resulting in same problem of the model with cosine similarity.