Università di Bologna - Master's degree in Artificial Intelligence

**Assignment No. 2: fact checking, natural language inference**

Marco Costante, Alex Costanzino, Alessandra Stramiglio, Xiaowei Wen

## Abstract

In the last few years, disinformation and fake news have been spreading throughout the world at rates never seen before, creating the need for fact-checking organizations. Despite many human-based fact-checking organizations that are currently in operation, disinformation continues to briskly spread throughout the internet, and the existing organizations are unable to keep up. Automated fact checking refers to the process that seeks to verify factual information with the aim to promote veracity and correctness of the text.

In this assignment we experimented an end-to-end siamese classifier network, with shared layers, to handle multiple inputs, along with four encoding strategies and three merging policies. The net was implemented with Tensorflow API, based on Keras backend, and the performances tested with different hyper-parameters.

## 1 Data loading and inspection

We loaded the splits into separate dataframes composed by three columns: `Claim`, `Evidence`, `Label`, dropping everything else.

Then, as maximum sequence length we choose the maximum number of tokens between training and validation set (148 tokens), since we need the whole informative content for such task. In our case we know that such length can be used also for testing, since the maximum length of the sentences of the test set is way shorter. In a real-case scenario, where the test set is unknown, we should be careful when managing the test set to make inference, since we cannot truncate its sentences, otherwise it would change it, altering the performances.

We also noticed an important difference in the length distribution of claim and evidence sentences.

## 2 Data processing

At first we filtered the sentences removing unnecessary information for the task, such as tags, leading numbers and other spurious symbols. We kept stopwords since they are useful to tell apart refutations and support evidences.

We created a vocabulary starting from the GloVe one. Subsequently, we extracted the out-of-vocabularies (OOV) terms from all three sets and its embedding matrix, having care of not reintroduce OOV terms already counted while scanning the splits. Next, the initial vocabulary and OOV terms were merged and the embedding matrix built. Finally, we put our dataframes into matrixes, padding and truncating the sentences, using the aforementioned vocabulary to encode words into integers.

## 3 Models and training

The end-to-end siamese classifier is composed by:

- Two input layer for claim and evidence;
- A shared non-trainable embedding layer, initialized with the GloVe kernel;
- A selectable sentence embedding stack of shared layers;
- A selectable merge layer;
- An optional concatenation layer that adds the cosine similarity between claim and evidence as feature;
- A final binary classification head.

For embedding we decided to choose the minimum available dimension for GloVe (50) since higher dimensions did not improved the performances and also due to the slow train of bidirectional RNNs.

The sentence embedding could be chosen among: an encoding via RNN taking the last state as sentence embedding, the same encoding but taking the average of all the output states, encoding via MLP or encoding as the mean of its token embedding.

The merge layer could be chosen among: add layer, average layer and concatenation layer.

Several activation function has been tested for the dense layers, ending with *selu* activation with glorot normal initialization.

A strong 40% dropout has been used for both dense and recurrent layers, with an additional 25% recurrent dropout for the latter layers.

Dense layers was also regularized with Ridge and Lasso weights decayers.

We decided to chose as hidden units the number of the embedding dimension.

We initially trained our networks with RMSprop optimizer, using a learning rate scheduler that decays exponentially the rate, which starts at 0.001, but such schedule tends to cause stall at higher epochs. We later tried Adam optimizer since it is adaptative.

As loss we used binary focal loss with sigmoid activation function for the very last layer since it is a binary classification problem.

To early stop the training we tracked the binary validation accuracy, with a patience of 15 epochs.

## 4 Results and error analysis

The last state RNN strategy was used as baseline, considering the cosine similarity as additional feature since it slightly improve the training.

The evaluation metrics for the classifiers are:

| **Merge strategy** | **Validation accuracy** | **Test accuracy** | **Test precision** | **Test recall** | **Test F1** |
|---|---|---|---|---|---|
| *Add* | 0.78 | 0.63 | 0.85 | 0.63 | 0.68 |
| *Average* | 0.63 | 0.70 | 0.69 | 0.69 | 0.69 |
| *Concatenate* | 0.79 | 0.68 | 0.75 | 0.68 | 0.69 |
| *Max* | 0.79 | 0.64 | 0.81 | 0.64 | 0.68 |

The evaluation metrics for the majority vote are:

| **Merge strategy** | **Test accuracy** | **Test precision** | **Test recall** | **Test F1** |
|---|---|---|---|---|
| *Add* | 0.63 | 0.57 | 0.96 | 0.72 |
| *Average* | 0.69 | 0.69 | 0.69 | 0.69 |
| *Concatenate* | 0.68 | 0.63 | 0.87 | 0.73 |
| *Max* | 0.64 | 0.58 | 0.93 | 0.72 |

Validation accuracy may not be the best proxy to track down the training since the dataset is quite unbalanced. A more thorough and fine text cleaning may also help in obtaining better performances.

## 5 Future developments

As future developments we could use more deep models to enhance the representation capacity. Then, we could also explore other scheduling strategies for the learning rate and other optimizers as well. Also more refined strategies for embedding sentence (such as BERT or GRUs) and merge policies could be experimented. Since there are very long sequences also attention mechanism may be employed.