Assignment 2: fact checking, natural language inference (NLI)

Piero Castriota, Usha Padma Rachakonda, Yellam Naidu Kottavalasa

August 30, 2022

Abstract

In this assignment we are focusing on a small portion of the whole fact checking problem: we want to train a model able to decide if a set of claims, given a set of related evidences that support or reject each of them, can be considered true or fake.

1 Data Preparation

The dataset has been downloaded and divided in the three sets: train, validation and test.

First of all, for each of the sets, an operation of text cleaning has been carried out, turning the content into a uniform, easy-to-read text. In particular, as last step of the cleaning process, each field has been turned into a set of unique words, keeping the original order: most of the fields presented a large amount of repetitions, in this way the text is much more meaningful, and most importantly, the size of the sequences is significantly reduced. This process requires a couple of minutes but improves significantly the efficiency at training time.

2 Dataset conversion

We downloaded the GloVe embedding with embedding dimension 50, in order to significantly speed up the training process, since it doesn't really compromise the results. At first a vocabulary of known words was created, initialized as the one of the embedding, in order to find the Out Of Vocabulary (OOV) words in the training set. The embedding matrix was updated with a new vector randomly initialized for each of the oov words. The same process of finding the oov and update the embedding matrix was repeated for both validation and test set.

Preparing the data for the training, we created a list for the labels, two matrixes for each set representing the content of the dataframe through ints and padded them (to have a single dimension for all the entries) to the max seq length, which was drastically reduced thanks to the pre-cleaning process, dropping from 220 to 94.

3 Model definition

Different models were inspected. They all share the same input: two input layer, one for the evidences and one for the claims. Each of the input layer is followed by an embedding layer which is non-trainable. At this point, the different models are characterized by different layers for sentence embedding, for both the inputs:

- RNN with last states (model1)
- RNN with average of all the output states (model2)
- MLP layer (model3)
- Bag of vectors (model4)

Also for the merging of claim and evidence sentence embeddings different options were tried: concatenation, sum and mean, with or without considering the cosine similarity between the two sentence embedding.

4 Training

For the training we decided to use a batch-size of 128 in order to speed up the process. After some initial attempts, it's been observed that: considering the cosine similarity was always improving the results and that the most promising way of merging seemed the concatenation. Accordingly, four models were trained using cosine similarity and concatenation merging, utilizing each of the different ways for the sentence embedding.

The model considering the last states of the RNN was the one performing better, so other two models were trained similar to model1, but one with the addition merging (model5) and the other with the averaging (model6)

Summary of evaluation on val data		
	val_loss	val_accuracy
model1	0.5552	0.7132
model2	0.5480	0.7125
model3	0.7184	0.5958
model4	0.7473	0.5347
model5	0.5441	0.7164
model6	0.5428	0.7253

BinaryCrossentropy was chosen as loss function since we are developing a binary classifier and Adam was chosen as default optimizer.

5 Evaluation

In order to evaluate the performances of the trained model two approaches were used: Multi-input classification evaluation and Claim verification evaluation

The first one concerns the simple evaluation of the network by evaluation metrics, assessing its performances, the second one instead classify the claim based on a majority-voting system based on the evidences.

Multi-input classification evaluation			
	test accuracy	f1-score (0)	f1-score (1)
model1	0.71	0.62	0.76
model2	0.69	0.60	0.75
model3	0.59	0.38	0.70
model4	0.53	0.18	0.67
model5	0.71	0.63	0.76
model6	0.71	0.64	0.76

Claim verification evaluation			
	test accuracy	f1-score (0)	f1-score (1)
model1	0.70	0.62	0.76
model2	0.69	0.60	0.75
model3	0.59	0.38	0.70
model4	0.53	0.18	0.67
model5	0.71	0.62	0.76
model6	0.71	0.63	0.76

The two evaluations give quite the same picture: models 1,2,5 and 6 have very similar performances with accuracy and f1-score around 0.70, which can be acceptable. Models 3 and 4 (mlp and bov) instead give poor results.

From these results we can deduce that,in this case, RNN was the best way to perform sentence embedding, while the choice of the merging strategy, between the ones tried, doesn't really alter the results

To furtherly improve the performances of the networks in the future, we could add more pre-processing step, in order to have more clean and meaningful data, and also, since all the models encountered an early stopping, a longer and more accurate process of tuning of the learning rate decay's parameters.