Fact Checking, Natural Language Inference

Andrea Proia, Federico Spurio, Cristian Urbinati

December 2021

Abstract. This work aims to accomplish the fact checking task by verifying if a claim is supported by a certain evidences. We use different techniques to obtain the embedding of each sentence, then we merge the embedding for each pair (claim, evidence) and pass through a classifier to predict the label. At the end, we determine if the claim is supported or not based on the majority voting on the predicted label for each evidence. Since we didn't exploit some advanced techniques both for sentence embedding and merging, our best model produces average results.

Introduction

The task is to determine if a claim is supported or refused given an evidence sentence. In some cases, there is more than one evidence per claim so, we consider it as supported or not, based on a majority voting.
We use GloVe pre-trained model to obtain a vector representation of each word in a sentence and then we combine them to compute the embedding for claims and evidences. Finally, we test some merging strategies in order to collapse the two inputs into a single one that is the input of the classifier.

1 Dataset preprocessing and data conversion

We train our model using the FEVER dataset about facts from Wikipedia documents to be verified, that has been already prepared to have for each row the triplet: claim, evidence and label. We also need to perform some cleaning, for example, we remove digits at the beginning of the evidence and tabs separated words after the last dot that do not contribute to the meaning of the sentence. We also remove all the punctuation and parenthesis token strings because, since we aim to obtain the embedding of the sentence, those symbols do not encode much relevant meaning. We also note that some words are not in ASCII format, for example characters used to express phonetic of a word usually found in Wikipedia pages, we choose to discard them as well. In the end, we convert all the words to lowercase in order to match with the GloVe embeddings. Next step was to build the vocabulary and the embedding matrix, the former associates and index to each word, the latter stores for each row the embedding vector of the word matching that index. We manage out of vocabulary (OOV) words associating them a random embedding vector.
Finally, we pad all the sequences of words in each claim and evidence to a maximum length by means of the string `-PAD-` encoded with 0 in the vocabulary. The maximum length, is determined by the longest sentence (claim or evidence) in training set, emulating a real scenario where we can't prevent to have longer sentences at test time. If that happens sentences are truncated.

2 Model definition and training

The model is structured into three macro-blocks: sentence embedding, merging, classification. For the sentence embedding of claims and evidences, we exploiting various techniques:

1. An RNN to encode token sequence and take the last state.
2. An RNN to encode token sequence and average all the output states.
3. A MLP in which we flatten the token sequences and then use a Dense layer to retrieve the sentence embedding
4. Compute the mean of token sequences (bag of vectors).

Next, we need to perform the merging between each claim and evidence pair so that they can be passed to the classifier. We tested three strategies by making the concatenation, the sum or the average of the embedded sequences and we also implemented the extension concatenating the cosine similarity between the two embedding vectors. Finally, a classifier composed by two

Dense layers that takes the merged input and predicts the support label.
For the train, we use the sparse categorical crossentropy loss and, as evaluation metric, the sparse categorical accuracy. We also exploit some regularization techniques since the model tends to overfit, in particular we use early stopping to stop the training as it get worse by checking the validation loss, weight decay on the first Dense layer and a Dropout layer between the two Dense. We think that this happens since our models struggle to generalise because there are examples like:

Claim	Evidence
google search can find stock quotes	these include synonyms weather forecasts time zones stock quotes maps earthquake data movie showtimes airports home listings and sports scores

in which the subject of the claim doesn't appear in the evidence but is supported, so is difficult to find a correlation.

3 Performance evaluation

In the following table we can see best values obtained during the validation phase:

Model	loss	accuracy
sentence emb. 1 + conc. merge	0.6042	71.24%
sentence emb. 1 + sum merge	0.5792	71.42%
sentence emb. 1 + avg merge	0.5773	72.64%
sentence emb. 2 + conc. merge	0.6939	66.46%
sentence emb. 2 + sum merge	0.5745	69.99%
sentence emb. 2 + avg merge	0.6225	68.60%
sentence emb. 3 + conc. merge	0.7478	58.73%
sentence emb. 3 + sum merge	0.7830	55.24%
sentence emb. 3 + avg merge	0.7564	55.84%
sentence emb. 4 + conc. merge	0.8113	50.40%
sentence emb. 4 + sum merge	0.8088	50.40%
sentence emb. 4 + avg merge	0.8076	50.40%

We finally choose to evaluate on test set the best model with higher accuracy score on the validation set. Below, a table that contains all the relevant metrics such as the accuracy, f1-macro, precision and recall on the test set using the majority voting strategy in order to assign the label to claims with multiple evidences:

Model	accuracy	f1-macro score	precision	recall
sentence emb. 1 + avg merge	71%	70%	74%	71%

4 Conclusions

As expected, we observe that in general the use of the LSTM produces a better sentence embedding since it can highlight the most significant terms, so it learns the semantic of the sentences, while MLP and the average of the words embedding, are not capable of doing so and therefore perform worse. In particular, the simple average of the words embedding is not a valid strategy since it performs nearly random.
On the other hand, the different merging strategies lead to similar performances, in fact, we haven't seen a big difference between keeping all the information with the concatenation or aggregating the vectors using sum or average, neither concatenating the cosine similarity. We believe that the cosine similarity is not so helpful because this score can be high whether the evidence supports the claim or not. For example, if the evidence is simply the negation of the claim, in case the model doesn't put enough emphasis on the negation, the two vectors will be very similar (high cosine similarity), even if the evidence doesn't support the claim.
An important limit of the model is that, since there is a fixed maximum length for the input sentences determined at training time, it would work only with sentences of length smaller or equal than this size.
Also the assignment of random vector for OOV terms may not be ideal, we also try to make the Embedding layer trainable so that it can learn a posteriori a better interpolation for word embedding vectors and we have achieved better results in LSTM and MLP models (around 74% of accuracy for the first two and 71% for the third), but we decide not to keep it because it overcomes to most of the work done by sentence embedders since results are very similar.
We also notice that a classifier composed of two Dense layers helps to improve the accuracy both in train and validation and we think that can be another point in order to improve the model.
In conclusion, however, our results are not really good, there are some parts that can be tuned and we believe that using transformers capable of applying attention mechanisms to do the sentence embedding can boost performances of the model a lot.