NLP course 2021 - University of Bologna

Assignment 2 Report

Michele Luca Contalbo micheleluca.contalbo@studio.unibo.it

Ehtsham Akhter ehtsham.akhter@studio.unibo.it

Vida Zahedi vida.zahedi@studio.unibo.it

Yiran Zeng yiran.zeng@studio.unibo.it

March 28, 2023

Abstract

This paper is a report which describes the second assignment given at the Natural Language Processing course at the University of Bologna. The assignment is centered on a small portion of the **fact checking** problem, which aims to estabilish whether a given statement is supported or refuted by a set of evidences. We were asked, given the FEVER dataset, to create some neural network models able to face this problem and compare their results. The tested models differ on how the sentence embeddings are extracted and combined together. Based on the metrics used (accuracy, f1-score, recall and precision), among all the architectures tested, the RNNs models provide better results.

Keywords
fact checking, FEVER, RNN, sentence embedding

1 Introduction

In this section we will provide information about the assignment, the dataset structure and its preprocessing.
In this assignment we need to check if some claims are supported or refuted by set of evidences. The dataset used is FEVER, which as been devised with the idea to aid creating software able to tackle false information coming from unreliable sources.

1.1 Dataset Structure and Preprocessing

The dataset is about facts taken from Wikipedia documents and it consists of 185,445 claims manually verified and classified as `Supported`, `Refuted` or `NotEnoughInfo`. The dataset is composed by the following features:

- `ID`: id associated to the fact to verify;
- `Verifiable`: whether the claim is verifiable or not;
- `Claim`: fact to be verified;
- `Evidence`: data structure composed by IDs that can be associated to the claim;
- `Label`: whether the `Evidence` `Supports`, `Refutes` or has `NotEnoughInfo` on the claim.

For this assignment, we are not interested in non verifiable claims, hence some preprocessing is done in order to filter out data which is not useful for our problem. In particular, we will not consider the `Verifiable` feature and focus only on `Evidence` supporting or refuting its corresponding `Claim`. Also, the `Evidence` feature has been modified in order to contain text and not IDs.

1.2 Text Preprocessing

Standard text preprocessing strategies have been applied with the aim to reduce noise and focus on more semantically meaningful words. Each word in all sentences, after having created the vocabulary, have all been converted to their respective vocabulary IDs. The sentences have been padded to 122, which is the maximum sentence length between `Claim` and `Evidence` in training, validation and testing set.

2 Architectures and Evaluation Methods

2.1 General Structure

Each model differs from each other on how sentence embedding are extracted and combined together. Each architecture has the following structure:

Architecture General Structure	
Layer name	**Input → Output size**
Embedding	122 → (122,50)
Sentence Embedding	
Merging	
Dense	50 or 100 (concatenation) → 2

The Embedding Layer is fed with 2 inputs of size 122. The embedding matrix, used in the first layer, has been obtained from **GloVe**[1] with **embedding size** of 50. In the last one, `softmax` as been used as activation. In each layer, **masking** has been applied to avoid considering padded values. The models have been trained on 5 **epochs** and **batch size** of 256, with **Adam** as optimizer and **cross entropy** as loss function.
To reduce overfitting, we applied **dropout layers** of 0.3 and **l2-regularization** of 0.001.

2.2 Sentence Embedding

The models implemented to obtain sentence embeddings are the following:

- RNN layer with last output as sentence embedding;
- RNN layer with mean of all outputs as sentence embedding;
- MLP layer with reshaping;
- Mean of token embeddings (**bag of vectors**)

The following table summarizes each of their structure:

Sentence Embedding Structure		
Model	**Layers**	**Input → Output size**
RNN with last output	RNN	(122,50) → 50
RNN with output mean	RNN	(122,50) → (122,50)
	Average Sentence[2]	(122,50) → 50
MLP	Reshape	(122,50) → 6100
	Dense	6100 → 256
	Dense	256 → 128
	Dense	128 → 50
Bag of vectors	Average Sentence	(122,50) → 50

Further techniques have been tested, like **Bidirectional RNN**, **LSTM** and **GRU**.

2.3 Merging

After having embedded the `Claim` and `Evidence` sentences, we must decide how to merge the results. The following strategies have been tested:

- **Concatenation**: concatenation of claim and evidence sentence embeddings (size doubled);
- **Sum**: sum of claim and evidence sentence embeddings;
- **Average**: average of claim and evidence sentence embeddings.

In addition, we have also tested whether these strategies can be improved by adding the **cosine similarity** to the merging output.

3 Results

The results can be seen from tables at the end of the paper.
The best model seems to be the **RNN with last output** as sentence embedding. Further extensions using more sophisticated techniques don't seem to improve the general result.
From the Figure 1, we can see that the models suffer from **overfitting**. By applying the reduction strategies, the models seem to slightly improve. This behaviour is followed also by the other models.

[1] https://nlp.stanford.edu/projects/glove/
[2] Lambda Layer

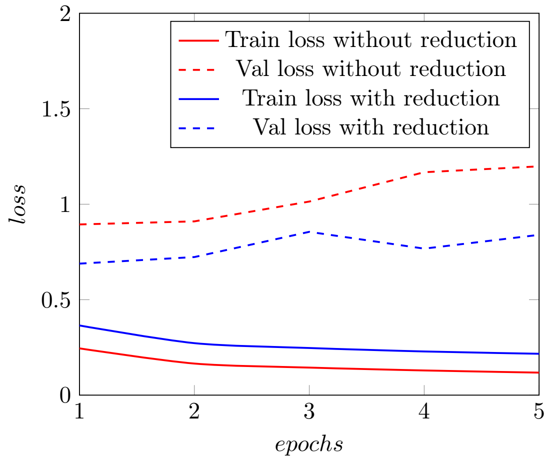

Figure 1: RNN with last output loss comparison

The best merging strategy is **concatenation**, while adding a cosine similarity value does not improve the result. **Majority voting** seems beneficial for every model, improving the scores by ≈ 0.02.

Conclusions

We have seen how different baselines perform on the FEVER dataset. Improvements can be done by tackling overfitting more efficiently. Indeed, the reductions applied in this assignment don't seem to be enough and further simplifications of neural architectures could aid the models to prevent overfitting.

Acknowledgements

We would like to thank prof. Paolo Torroni, prof. Andrea Galassi and prof. Federico Ruggeri for the help provided during the implementation of the tasks.

[3] with cosine similarity

General Results		
Models requested by assignment		
Model	**Concatenation merging**	
	acc	**f1**
RNN with last output	0.6981	0.7256
RNN with average output	0.6847	0.7222
MLP	0.6919	0.7237
Bag of vectors	0.6982	0.7489
Extensions		
Bidirectional RNN	0.6827	0.7176
LSTM	0.7087	0.7510
GRU	0.7030	0.7391

F1 scores with different merging strategies						
Model	**Concatenation**		**Addition**		**Average**	
	W/ c.s.[3]	**W/O c.s.**	**W/ c.s.**	**W/O c.s.**	**W/ c.s.**	**W/O c.s.**
RNN with last output	0.7117	0.7256	0.7202	0.7227	0.7189	0.7196

Results with overfit reduction strategies				
Model	**Concatenation merging**		**Majority voting**	
	acc	**f1**	**acc**	**f1**
RNN with last output	0.7119	0.7413	0.7152	0.7408
RNN with average output	0.6565	0.7247	0.6599	0.7244
MLP	0.7111	0.7368	0.7113	0.7355
Bag of vectors	0.6308	0.7273	0.6331	0.7284