Stance Classification

Nicola Poggialini - 0000949125
nicola.poggialini@studio.unibo.it

February 9, 2023

1 Introduction

Stance classification is the task of automatically identifying the position of the author of a document towards a certain target.

In this report, I'll start with explaining briefly the main concepts behind my work. Then, I'll present the dataset used, the architecture proposed and the experiments done. I'll conclude with analysing the results obtained and a brief reflection on some future work.

2 Background

2.1 Transformers and BERT

Text data are traditionally processed using Recurrent Neural Networks (RNNs). This type of model processes sequences, so it receives an input which has some defined ordering. This is why they suit perfectly text data, where the order of terms is certainly important. Some variants of RNNs have been proposed during the years in order to improve the training of these models, such as LSTMs and GRUs: in fact, they manage to alleviate the problem of gradient vanishing.
A ground-breaking improvement in NLP was given by Transformer [7], an encoder-decoder model which processes all the words in a sentence simultaneously and converts them into embeddings. It is based on the self-attention mechanism: for every word, it computes which are the tokens in the sequence to which we should assign a bigger weight. It also uses positional-encodings, in order to not lose the information given by the position of the words in the sentence.
Based on multiple Transformer encoders, BERT [1] was then designed. What is relevant about this model is its famous pre-training process. In particular, this was based on two self-supervised tasks: masked languange modeling (MLM) and next sentence prediction (NSP). Thanks to the pre-training (and also a task-specific fine-tuning), amazing results were obtained exploiting the word embeddings generated by BERT.
However, word embeddings are not a new topic. Before BERT, algorithms like Word2Vec [4] were able to generate numerical vectors to be efficiently used in downstream tasks. Still, BERT (and other transformers-based language models) provide better representations than models like Word2Vec. In fact, while each word has a fixed representation under Word2Vec regardless of the context within which the word appears, BERT produces word representations that are dynamically informed by the words around them.

2.2 BERTweet

BERTweet [6] is the first public large-scale language model pre-trained for English Tweets. Based on the same architecture as BERT, BERTweet is trained with the RoBERTa [3] pre-training procedure (which is an optimized version of BERT's pre-training). The corpus used to pre-train BERTweet consists of 850M English Tweets.

3 Dataset

The dataset that I've used in this work is the SemEval-2016 dataset (task 6) [5], which is focused on tweets. There are 3 possible stances (*Against*, *Favor*, *None*) and 5 different targets (*Hillary Clinton*, *Feminist Movement*, *Legalization of Abortion*, *Atheism*, *Climate Change is a Real Concern*).

(The term 'target' will be used in this report only with this meaning; 'stance' will be used to indicate what we want to predict)

The authors of the data provided 3 sets:

- Training (2914 instances);
- Test for task A (1249 instances), containing the same targets seen during training;
- Test set for task B (707 instances), with only a target (*Donald Trump*) but unseen during training;

3.1 Data exploration

Let's see some statistics about the training data.
In Figure 1, we can observe that the majority of samples belong to the targets about Hillary Clinton, feminism and abortion, which are quite equally distributed. Less instances instead for the targets about atheism and climate change.

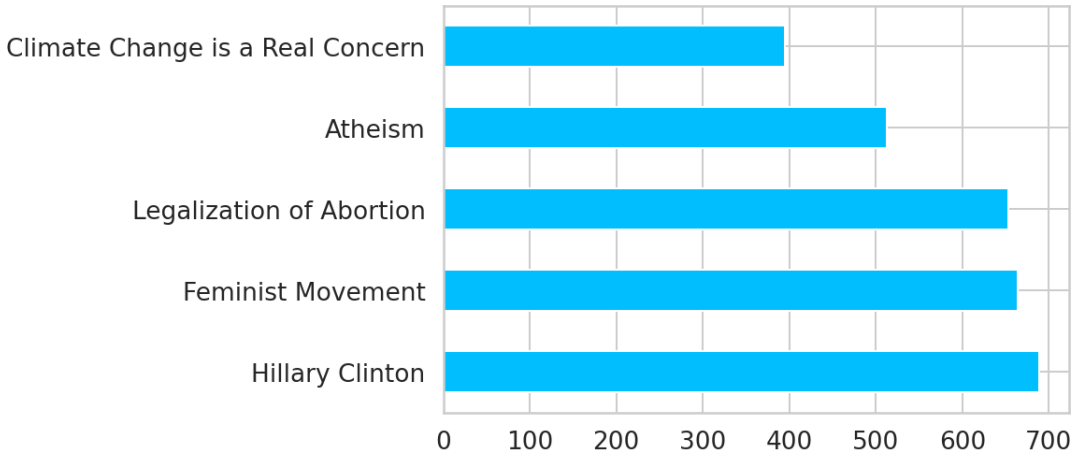

Figure 1: Number of instances per target.

Let's also observe in Figure 2 the distribution of the instances regarding the stance. We can see that the data are quite unbalanced: the 'Against' instances double in number each one of the other stances.

3.2 Pre-processing

BERTweet's authors provide in their implementation a function (*normalizeTweet*) to pre-process raw tweets as they did. This constists of:

- tokens are extracted from Tweets using TweetTokenizer from the NLTK;
- emojis are translated into text strings;
- tweets are normalized by converting user mentions and web/url links into special tokens @USER and HTTPURL, respectively.

I've used this function in my work to produce a cleaned version of the tweets in the dataset.

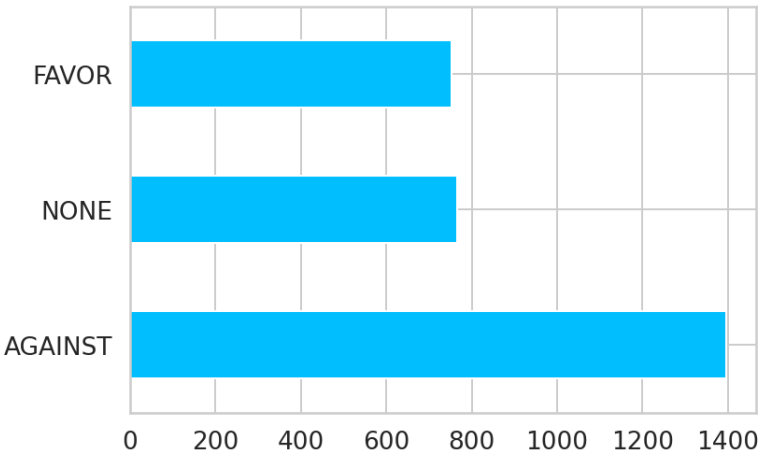

Figure 2: Number of instances per stance.

4 Model

In this section I'll describe the model used to predict the stances. First of all, I'll present the original architecture that I've selected from the literature. Then, I'll focus on the modifications made to it and the experiments done.

4.1 Original architecture

I've selected *Target-specific Attentional Network (TAN)* proposed in [2] : in Figure 3 we can see an overview of its structure.

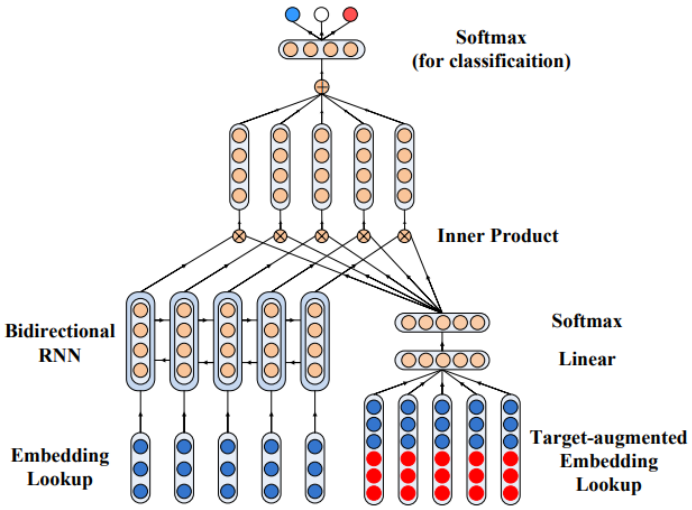

Figure 3: TAN architecture.

The model is based on the following components:

- word embeddings from tweets are extracted using Word2Vec and are passed to a bidirectioal LSTM (the outputs are concatenated);
- word embeddings from targets are extracted in the same way as tweets;
- target embeddings are used to create an augmented version of the embeddings extracted from tweets;

- the augmented embeddings are passed to a linear layer (to obtain a scalar value per item in the sequence) and then to a softmax (this results in *attention scores*);
- the output from the LSTM and the attention scores are combined with a product;
- after the product, the resulting word vectors are combined with a mean;
- finally, a linear layer and a softmax are used for classification.

4.2 Base model

My base model is based on the idea of exploiting modern language models for this task. As I already mentioned, to generate word embeddings, these are better than methods like Word2Vec, used in the original architecure. So replacing Word2Vec embeddings with those from BERTweet for tweet and targets could produce an improvement in the performances of TAN.

4.3 Experiments

Except for the embedding part, the rest of the base model is the same as the original one. This includes also the hidden dimension of the bi-LSTM, which is equal to 100, and the number of layers in the recurrent neural networkm, which is 1.
Although, using BERTweet, now we have an embedding dimension of 768, which is much greater than the one used for Word2Vec (300).
My experiments were based on the idea that this abrupt change in dimensions (from vectors of 768 to vectors of 100*2) could result in some information loss. I've tried to address this potential issue with the following 3 strategies:

1. Increase the hidden dimension of the bi-LSTM to 200: this way we have a less abrupt change.
2. Increase the number of layers in the bi-LSTM to 2 and use the best dimension found in the first experiment: doing this we have more processing of the embeddings, which could result in less information loss.
3. Use 2 bi-LSTMs but with different hidden dimensions, the first bigger (300) and the second smaller (100): this could produce a more gentle change of dimension.

5 Experimental set up

The following is a summary of the hyper-parameters used for training the models.

- **Train set split**: 0.9
- **Loss**: Cross-Entropy loss
- **Optimizer**: AdamW
- **Batch size**: 32
- **Learning Rate**: fixed 1e-5
- **Number of epochs**: 50
- **Input length - Tweet**: 50
- **Input length - Target**: 8

I chose the train set percentage in order to have a quite consistent number of instances for training the model, since the data don't contain many instances.
The optimizer, the batch size and the learning rate are from BERTweet's paper, where the authors fine-tune their model on a specific task. They used 30 epochs, I decided to increase them to 50 to have some margin. The loss function is taken instead from TAN's paper.

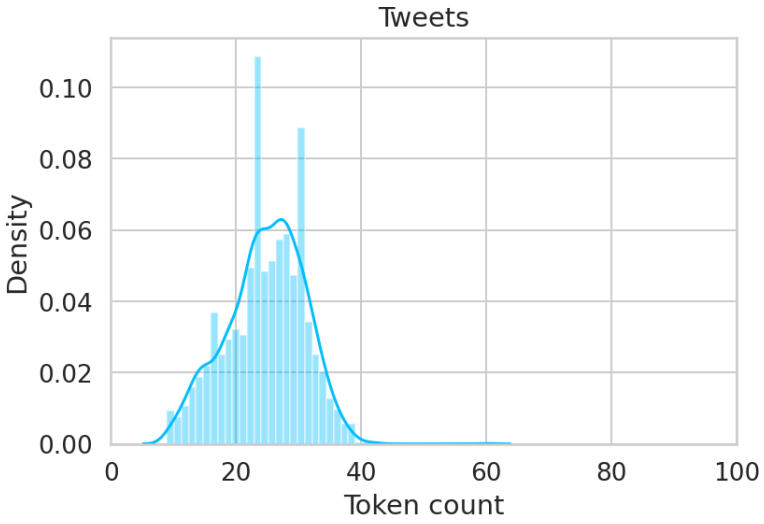

Figure 4: Distribution of tokenized tweets' length.

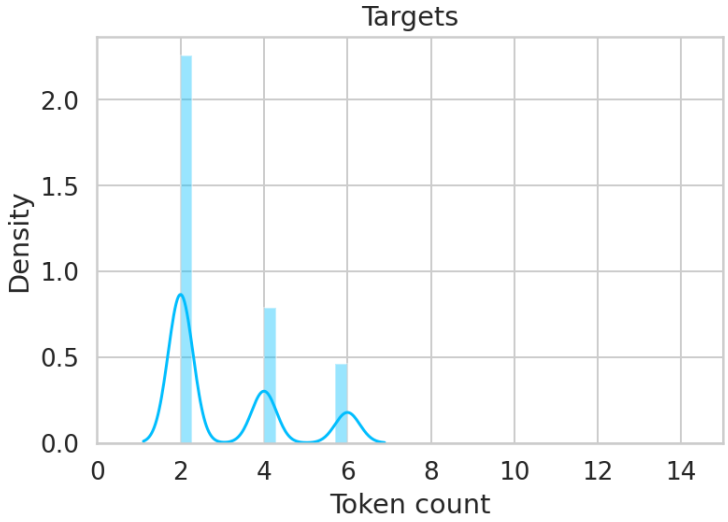

Figure 5: Distribution of tokenized targets' length.

Regarding the length of the input, the values are chosen by looking at the distribution of the tokenized tweets and targets (Figure 4 and Figure 5, respectively).

We can observe that all tweets have less than 50 tokens, so I can safely use that as the maximum length of the input sequences. Regarding the targets, they are all under 8, so we can apply the same reasoning.

6 Results

6.1 Metric

To evaluate the models, I computed the same metric used in TAN's paper, which is defined as the mean between the F1 score of the classes *'AGAINST'* and *'FAVOR'*:

$$F_{Favor} = (2 * P_{Favor} * R_{Favor})/(P_{Favor} + R_{Favor}) \tag{1}$$

$$F_{Against} = (2 * P_{Against} * R_{Against})/(P_{Against} + R_{Against}) \tag{2}$$

$$F_{average} = (F_{Favor} + F_{Against})/2 \tag{3}$$

In the equations, F_{class}, R_{class}, P_{class} are, respectively, the F1 score, the precision and the recall of the corresponding class. $F_{average}$ is the metric that will be used to evaluate and compare the models.

6.2 Evaluation on validation set

In this section, I'll report the results obtained for each model. In particular, I'll show the trend of the $F_{average}$ during the epochs and I'll report the best value and at which epoch was achieved.

6.2.1 Base model

- Best F average: 0.7063
- Epoch: 33

Figure 6: Base model - results during training.

6.2.2 Experiment 1

- Hidden dimension = 200
- Best F average: 0.6764
- Epoch: 29

The base model returned a better value for the evaluation metric: I'm going to use its hidden dimension (100) for the bi-LSTM in the next experiment.

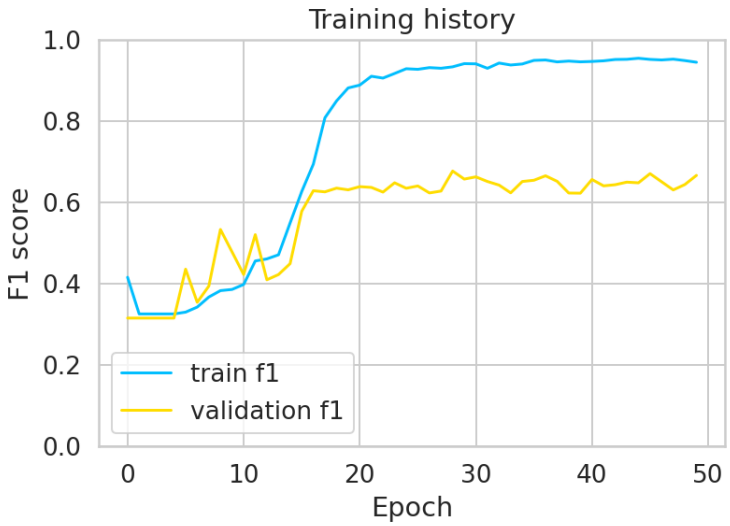

Figure 7: Experiment 1 - results during training.

6.2.3 Experiment 2

In this case, I noticed that the scores were still going up around the 50th epoch, so I decided to continue the training for 20 more (Figure 9).

- Hidden dimension = 100, number of layer = 2
- Best F average: 0.6824
- Epoch: 58

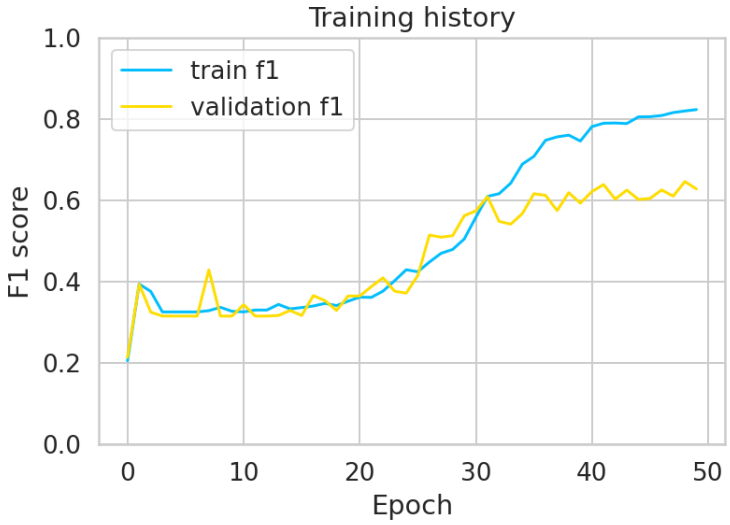

Figure 8: Experiment 2 - results during the first 50 epochs.

This second experiment achieved better results with respect to the previous one: still, the score is a bit worse with respect to the base model.

6.2.4 Experiment 3

For this experiment (2 bi-LSTMs with different hidden dimensions) I stopped the training after 30 epochs: the scores were stuck at the same values since epoch 1.

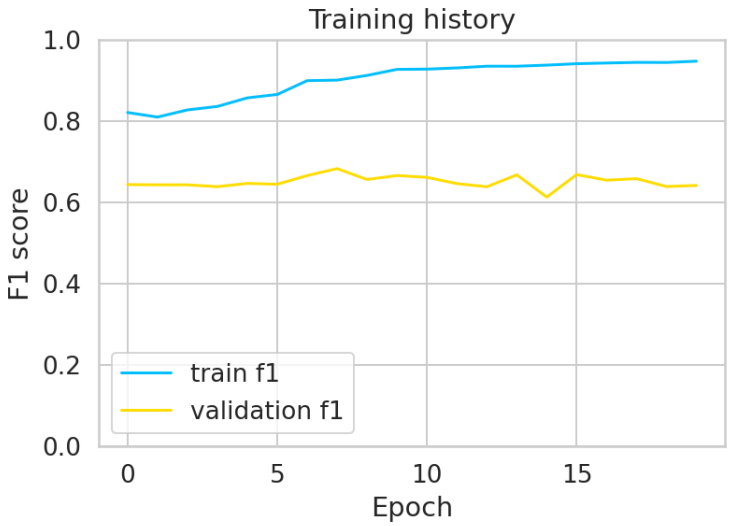

Figure 9: Experiment 2 - results during the last 20 epochs.

6.2.5 Summary

In Table 1 I reported a summary of the results obtained.
The base model is the one that performed better on the validation set and will be the one that I'm going to use for the final evaluation on the test sets.

Model	F average	N epochs
Base model	**0.7063**	33
Higher hidden dim.	0.6764	29
More bi-SLTM layers	0.6824	58
Different bi-LSTM layers	-	-

Table 1: F average on the validation set.

6.3 Evaluation on test set

The results obtained on the test sets are listed in Table 2. We can observe that my proposed model is comparable in terms of performance to TAN: still, it isn't able to replicate the same results.
On the other hand, the scores achieved on the dataset with the unseen target are quite low.

Model	Test set A	Test set B
My model	0.6425	0.2289
TAN	**0.6879**	-

Table 2: F average on test sets.

7 Conclusions and future work

This attempt in the task of stance classification resulted in a model comparable to the original one, but not able to achieve the same performances.
It's still worth noticing that TAN's overall predictions derive from the predictions made by 5 models,

all with the same architecture but trained on the samples of only one specific target. So this means that the model can't be used in a general context, in which a new target appears: in fact, they perform the final evaluation on just the test set with the known targets.
On the other hand, my model was trained on all the instances together, regardless of the target, so it can be used also with a new target not seen during training (however, just in principle: the performances on task B were not satisfying). The motivations behind this approach were not only about the ability to generalize, but also the potential difficulties in training one model per target. In fact, the train data are composed by just a couple thousands of examples, which means that there are only few hundred instances per target: these are probably not enough to train a transformers-based model like the one proposed.
To conclude, some considerations about possible future work. In the proposed model I just experimented on the tweet side of the architecture, leaving the target side fixed. So, some further experiments could focus on this: for example, trying some processing after the extraction of the embeddings and before the attention layer (adding one LSTM, or one more linear layer).

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. Stance classification with target-specific neural attention. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3988–3994, 2017.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[5] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California, June 2016. Association for Computational Linguistics.

[6] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. BERTweet: A pre-trained language model for English tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online, October 2020. Association for Computational Linguistics.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.