DistilBERT for stance detection on Twitter

Pierre Groshens
pieg@itu.dk

Frederik Skolvund Kilpinen
fkil@itu.dk

Abstract

In order to maintain democratic enlightening conversation threads online, stance detection on dubious social media posts caught the attention of both academic and commercial environments. In this paper we utilise transfer learning to detect tweet-stances based on the dataset released for Subtask A in RumorEval 2019. We implement a chain-based DistilBERT model that achieves performance close to the best performing models, but still suffers from imbalanced classes.

1 Introduction

Due to recent events in the international political landscape, the spread of disinformation on social media has been a widely debated subject of interest. An interesting aspect about the spread of disinformation is the stances taken in the conversations regarding the piece of information. This has led the way for the study of *stance detection*, which can be defined as the classification of a piece of information's position in regard to a prior statement (Gorrell et al., 2019). In order to learn the anatomy of misinformation and do veracity identification of posts, stance detection therefore plays a key role.

However, detecting stances on social media by manually monitoring all conversations is costly and unfeasible without machines. For this reason it is important to develop proper methods to automate such tasks. In this paper we do exactly this by employing the transformer DistilBERT as suggested in Sanh et al. (2019) to a set of annotated Twitter conversations to detect the various stances taken.

2 Background

Our choice of the stance classification subject comes from the RumourEval 2019 task proposed in Gorrell et al. (2019). The competition was split into two tasks from which we selected Subtask A: "*given a source tweet, tweets in a conversation thread discussing the claim are classified as either supporting, denying, querying or commenting on the rumour mentioned by the source tweet*".

In addition to this participants were provided with two annotated datasets: One from Twitter and from Reddit - our paper relies solely on the Twitter data. The data consists of a list of source posts introducing a rumour and their corresponding Twitter conversation. The objective of this paper is then to label each reply with one of the following stances in respect to the source post: comment, query, deny or support.

In the competition 22 teams participated in Subtask A, which were later won by the team *BLCU-NLP*. They used the pre-trained model GPT from OpenAI and achieved a macro F1-score 0.62. One of the key takeaways from this and other of the top performing models was that the use of time sequences in each conversation seemingly improved performance. That is defining each conservation starting with an initial post, the source tweet, which is followed by replies each responding to responses earlier in the time sequence (Gorrell et al., 2019). Therefore leveraging the time sequence of earlier responses is a key feature when doing stance detection on Twitter and in Section 4 we describe how this has affected our model selection.

3 Method

3.1 Data

Our data consists of 6349 tweets, split up into a training set of 4519 tweets, a validation set of 1049 tweets and lastly a testing set of 1066 tweets. Each tweet comes with a label (comment, deny, query or support). These labels are annotated following the code scheme from Zubiaga et al. (2015), who define the labels as the following:

- **Comment**: When the author of the reply makes their own comment without a clear contribution to assessing the veracity of either the

tweet they are responding to or the source tweet.

- **Deny**: When the author of the reply disagrees with the statement they are replying to.
- **Query**: When the author of the reply asks for additional evidence in relation to the statement they are responding to.
- **Support**: When the author of the reply supports the statement they are replying to.

The annotation was done via crowdsourcing by persons from the USA and UK. We consider the annotations to have a high degree of reliability, as an annotator agreement of 70 % was required for each tweet (Gorrell et al., 2019).

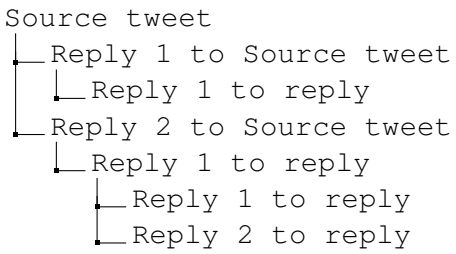
The distribution of the labels are fairly imbalanced with "comment" tweets dominating the dataset. Around 65 % of the tweets are labeled as comments, whereas only ≈ 15 % of the tweets are labeled as either query or deny. Lastly, tweets labeled support make up ≈ 20 % of the training data.

Table 1: Training set distribution

Comment (%)	Query (%)	Deny (%)	Support (%)
2907 (64.3 %)	358 (7.9 %)	344 (7.6 %)	910 (20.1 %)

Given this imbalance of data care must be taken when we assess the model performance, as a simple majority baseline would do well in terms of accuracy, but fail to distinguish between the labels support and deny. In Section 4.1 we outline the consequences of this for our assessment of model performance.

Figure 1: Data structure



Regarding the structure of the data, it is stored in a tree-like structure with the source tweet as the trunk of the tree and then the replies and the nested replies as the branches (see Figure 1). This allows us to track the conversations' entire time sequence and therefore treat them as dependent tweets.

Until now we have not touched upon the actual input values, namely the text of the tweets. They are made of 280 unprocessed characters at maximum, including links, emojis, etc.

3.2 Baseline

In order to determine whether or not stance detection requires a complex statistical model with millions of parameters, we need to decide on a relevant baseline.

The most simple model of comparison is a majority baseline. The majority baseline would use a simple heuristic and always guess unseen tweets as comments due to its high proportion of the training data. This will result in an accuracy of 0.72 and a macro F1-score of 0.22 for the test set. Although the data is unbalanced and the majority baseline therefore performs considerably better than random, we still expect it to be fairly trivial to outperform.

A more challenging baseline called branchLSTM is provided by Gorrell et al. (2019). For the RumourEval 2019 they outline a baseline that uses a neural architecture with Long short-term memory (LSTM) layers to process sequences of tweets. This model was the best performing system of an earlier iteration of RumourEval (2017) and we therefore expect it to be far from trivial to outperform this model. This model returns an accuracy of 0.81 and a macro F1-score of 0.49.

3.3 The model: DistilBERT

For our attempt at stance detection we rely on the user of transfer learning through *transformers*. One of the notable models, BERT, has shown to provide state-of-the-art results when it comes to a wide array of natural language processing tasks, including classification (Devlin et al., 2018). Due to the design of BERT to pre-train deep bidirectional representations from unlabeled text it can be fine-tuned without much data and still achieve state-of-the-art performance with an additional output layer (Devlin et al., 2018). Given the size of our training corpus we therefore find BERT ideal to classify the stances of the tweets.

However BERT is naturally also an extremely complex model with no less than ≈ 340 millions parameters. Even though we use a GPU, this is computationally heavy, especially as the process

of fine-tuning the model is an iterative process, where different specifications are tried out in order to enhance performance. For this reason we employ DistilBERT instead. DistilBERT is a distilled version of BERT, which drastically reduces the number of parameters ($\approx$ 66 millions), without losing corresponding performance. Sanh et al. (2019) estimate that the DistilBERT retains 97 % of BERT's language understanding capabilities, why we expect it to still perform satisfactorily for the task at hand.

4 Analysis

We divide our analysis into three branches in which each of the branches are made of different input data. At first, we choose to train our most simple model, that we call Non contextual DistilBERT, that relies solely on $tweet_i$ (t_i). This means that we treat each tweet as independent and do not take their context into account, when we train our model. We simply utilise the text of each tweet to predict its label.

However, as outlined in Section 2 prior models for the same task has shown to increase performance when they include the time sequence of Twitter conversations. We therefore estimate two additional models. First, we train a model, that we call Full history DistilBERT where we provide the full history of a tweet to the Distilbert. That is we define a conversation as starting with the source tweet and then feed the model with the entire list of replies appearing before the tweet which class we predict. Lastly, we also train a model, that we call Chain-based DistilBERT, that treats the conversations as inference chains initialised with the parent tweet followed by only the one reply earlier in the time sequence.

4.1 Model evaluation: Macro F1 score

Before we do model selection we have to decide on an evaluation metric. As we have imbalanced data and are dealing with classification of multiple classes, we follow the guidelines from Gorrell et al. (2019) and report macro F1 score as our performance measure. Formally, the macro F1 score is defined as:

$$\frac{1}{N}\sum_{i=0}^{N} F1score_i \tag{1}$$

where N is the number of classes and i is each class. The macro F1 score is therefore preferable in our case, as the F1 score is calculated for each class and then given the same weight, no matter the number of observations in the class. In this way, we avoid that correctly predicting comments boosts the performance to an undesirable degree.

4.2 Pre-processing on Non contextual DistilBERT

Traditionally, the preprocessing of text has played a central part of NLP tasks. However as we are dealing with a very complex model the need for preprocessing diminishes. In fact there is an imminent risk of worsening our model performance following a loss of information, if we are not careful in the pre-processing phase. For this reason we will e.g. do not lemmatisation, as we find this to remove too much information that DistilBERT is actually capable of processing meaningfully. Instead we have selected three preprocessing steps to consider for our models:

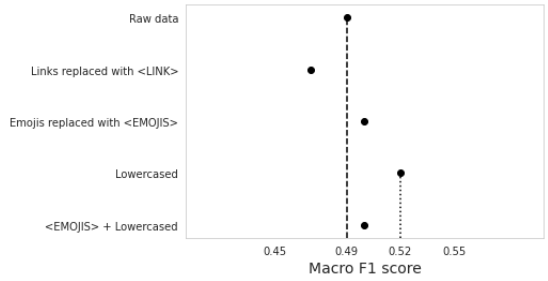
1. Turning links into <LINK> tags
2. Turning EMOJIS into <EMOJI> tags
3. Lowercasing the text

The way we assess whether or not to do a particular preprocessing step is by training the Non contextual DistilBERT with the different text inputs and then compare performances. One might expect each of the steps to have interaction effects with the other steps, however we do not want to try every combination. We have 4 different input variations (including raw data that has not been preprocessed in any way) so there are 2^4= 16 outcomes. This would be too timely to compute. Furthermore we have three different models, which would result in $16 \times 3 = 48$ outcomes. Instead, we compare the results of each method individually and only for the Non contextual DistilBERT, which is our most simple model that treats the tweets as independent. The results are visualised in Figure 2.

Surprisingly, the model performs better when we do *not* replace the links with tags. We were actually expecting better performance as it seemed unlikely for the model to extract information from rather idiosyncratic links.

While removing links does not seem to help, the opposite goes for replacement of emojis with tags, and especially lowercasing of text. We therefore mix up both of these techniques to create a final version of our inputs. However, the results do not

Figure 2: Macro F1-score for preprocessing steps



perform better. In fact, combining emoji tags plus lowercasing decrease the performance, as macro F1-score drops back to 0.50, why solely lowercasing seems to be the best strategy.

In order to make sure that we are not just interpreting statistical noise we train the lower cased model multiple times and test it on the validation set. The results from this seem rather robust with macro F1-scores of 0.52, 0.51 and 0.51 Based on these computations we decide to lowercase text for all of the models.

For the estimations referenced above we used the cased DistilBERT base model, that distinguishes between uppercase and lowercase characters. However, as we plan to lowercase the tweets for the rest of the models, we will instead implement the uncased DistilBERT base model. Therefore we also rerun the model with lowercase text using the uncased DistilBERT base model instead. Once again this resulted in a 0.52 macro F1-score (slightly higher for the third decimal), which serves as the score for our Non contextual DistilBERT.

4.3 Full history DistilBERT

Our second model takes the entire prior history (starting at the source tweet) of a tweet into account. In practice this is done by concatenating the tweets together. If a tweet (t_3) is a reply to a tweet (t_2) that is itself a reply to a source tweet (t_1), the text for t_3 will look like this: $t_1 < reply > t_2 < reply > t_3$.[1]

This also means that the shape of the input is no longer limited by the number of characters allowed in a tweet (280). We therefore allow for a lot more context for each prediction, which intuitively should enhance performance. Whether or not a tweet is to be seen as a supporting remark greatly depends on the initial statement. This also seems to be case for this model as the macro F1-score increases to 0.53 from 0.52 in the simple model without context. Although we allow for full context in this model, the improvement is quite minimal.

A possible explanation for this is, that providing a model with this much context is not without drawbacks: By using the full history of Twitter conversation we risk to drown important details into the mass. For example, suppose we have a query tweet with a question mark at the end. This character carries a lot of information and is probably one of the most discriminatory features for predicting query labels. The question mark would count for 1 of potential 280 characters. If we sum up all the tweets that preceded it, the question mark would then have less importance than before. Over time, the model would learn that a question mark posted at the end makes a lot of sense for queries. But as the models are provided with a lot more information, it would probably take more time to come up with this conclusion.

4.4 Chain-based DistilBERT

Our last idea for a model that utilises the context of a tweet is to concatenate only the previous tweet with the tweet. So if we use the example from before, with a tweet (t_3) that replies to (t_2) that has replied to a source tweet (t_1), the text for t_3 will now simply be $t_2 < reply > t_3$. Therefore this input actually has less context than the corresponding input from Full history DistilBERT, and relies on a premise resembling the Markov assumption, as we assume that only the previous tweet will have an impact on the stance (Jurafsky and Martin, 2020).

This is inspired from the model implemented by Yang et al. (2019) that presented the best performing system for the RumorEval 2019 (Gorrell et al., 2019). For this reason, we also a priori expected this model to perform the best, as it uses the best from both worlds by taking a limited context into account to avoid drowning important details in text. Indeed, this was also the case, as these new inputs reached the macro F1-score 0.54 the highest of all the models tested on the validation set.

Granted, this improvement is once again marginal. Nonetheless, the Chain-based DistilBERT, is our final model that we will use for testing. This is because (1) although only slightly better

[1] Importantly, tweets are only concatenated within their own branch. So if two tweets reply to a source tweet they create their own branch, wherein nested replies would not get concatenated with the initial reply to the source tweet from the other branch

Table 2: Validation set perfomance

Performance metric	Non contextual DistilBERT	Full history DistilBERT	Chain-based DistilBERT
Macro F1-score	0.52	0.53	0.54

than other models, it still performed best on the validation set, and (2) based on Yang et al. (2019) we have theoretical reason to believe, that this model should be superior for stance detection.

4.4.1 Test data performance

In Table 3 we have listed the test performance for our final model, the Chain-Based DistilBERT and compared it with our two baselines. At first glance it might be a bit surprising, that Chain-based DistilBERT is achieving a considerably higher macro F1-score for the test set. We believe that there is a simple explanation for this: The distribution of the training set closer resembles that of the test set compared with the validation set's distribution (see Appendix A).

Table 3: Test perfomance

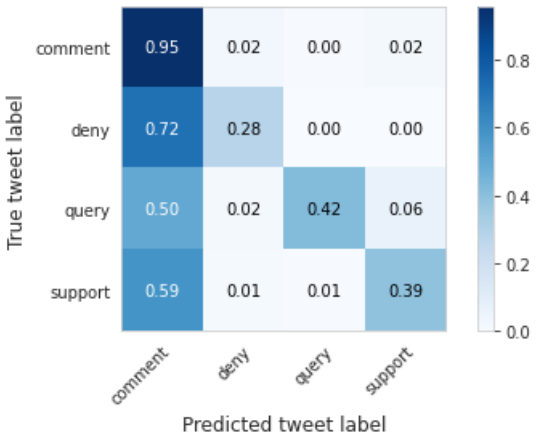
Perfomance metric	Majority baseline	branch-LSTM	Chain-based DistilBERT
Macro F1-score	0.22	0.49	0.58

As expected our model is clearly outperforming the majority baseline. More interestingly, our model also seems to perform significantly better than the branchLSTM-baseline. It is however difficult to assess exactly how good a macro F1-score is in absolute terms. Instead we can compare with the results from Gorrell et al. (2019), where our model would be in the high end of the contenders in Subtask A (number 3 out of 22). Given the time frame for this assignment, we believe that this is fairly good and highlights the potential of transfer learning. However, there is still a long way up to the best performing system that reached a macro F1-score of 0.62.

Based on the unbalanced data distribution outlined in Section 3.1 the patterns from the confusion matrix in Figure 3 are hardly surprising: Our model is over-predicting tweets as comments. Therefore we have relative low recall for tweets labelled deny ($Recall_{deny}$ = 0.28) and support ($Recall_{support}$ = 0.39). On the other hand the precision is fairly high for all tweets, except those predicted as comments.

If we want to achieve as high a class precision as possible these numbers are not bad. However

Figure 3: Confusion matrix for test data with Chained-based Distil-bert (normalised by rows)



whether or not this is desirable depends entirely on the purpose of the model. Imagine a situation where a social media company are looking to implement an algorithm that can help them regulate comments that show support to posts promoting misinformation. In that case the company might be willing to accept a lower precision, as it reduces the risk of encountering unseen supporting tweets. Furthermore, it might still greatly reduce their workload if the initial method relies on manually trawling through shoals of tweets. Therefore the relationship between a model's macro F1-score and its usefulness is not linear.

4.5 Potential improvements

As transformers seemingly outperform other current neural architectures (Vaswani et al., 2017), we believe transformers are needed to perform state-of-the-art stance detection. We find it difficult to pinpoint another transformer that theoretically should considerably outperform our model. Nonetheless as DistilBERT is a distilled version of BERT one might marginally improve performance at the expense of computing time by adapting it. Instead of changing the transformer architecture, we could also consider tuning the optimisation inputs. We have relied heavily on the default options from the `Huggingface` library, which might be preferable to customise more for the task at hand. However, looking at the loss (based on cross-entropy) in Figure 4) it seems that the model is trained enough to reach at least a local minimum after $\approx$ 1200 optimisation steps.

While we think it is a bit unclear how much performance can be improved from the model spec-

Figure 4: Loss for optimisation steps in the training loop (should be updated)

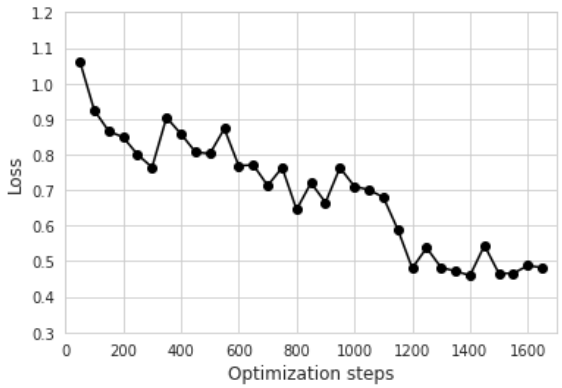
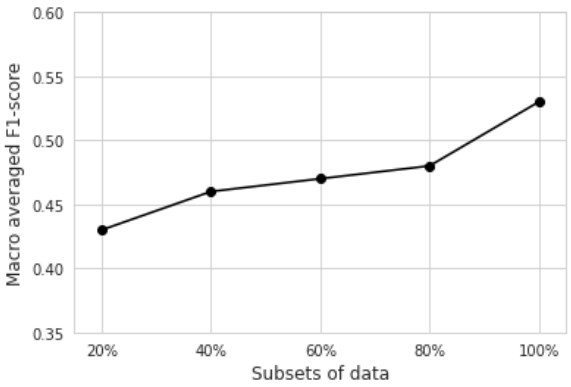


Figure 5: Test performance for subsets of training data (should be updated)



ifications, the picture is less blurry when it comes to the importance of the data. The confusion matrix in Figure 3 clearly showed how the imbalance of our data affected the final predictions. In our experience adjusting imbalanced data is far from trivial. As a means against this we tried to add a class weight element to our loss function and specified it to be $= \frac{N}{N_{class}}$. This and other variations thereof turned out to greatly $decrease$ model performance (macro F1-score e.g. went from 0.54 to 0.49 for our final model)[2]. The regularisation of the comment inputs simply worsened the model as we then began to mislabel comment tweets.

For future research one might therefore consider other strategies to overcome imbalanced datasets. Potential solutions could be to do as Yang et al. (2019) and look for external data of the same type, and then only keep data that has a similarity (they use Levenshtein Distance) below some threshold with the original dataset. In the more creative end, we have also seen people in Kaggle competitions attempted to extend the original dataset by translating it into a foreign language and back into English through Google Translate.

Lastly we have not touched upon the general amount of training data. 4519 tweets do not sound like much data, especially taking their length into account. On the other hand one of the promising results related to transformers is high performance despite limited data for fine-tuning (Vaswani et al., 2017). However in our case we do not seem to have enough data. There is still some large jumps in performance, which we believe partly stems from adding more training data and also random variation due to little data. This implies that to increase the performance of our model we should gather more data, alongside our strategy to deal with the imbalanced data.

[2]Given the discussion in Section 4.4.1 a lower macro F1-score does not per se entail a worse model depending on the purpose. However here we purely consider macro F1-score as a performance metric, as we do not know the ultimate goal of the model.

5 Conclusion

We have presented a DistilBERT model for stance detection in Twitter conversations based on the dataset from Subtask A in RumorEval 2019. In line with the results from RumorEval 2019 we find that the model performs best when we concatenate the tweet replied to unto the tweet which label we attempt to predict. This results in a macro F1-score of 0.58 for the test data. For future implementations of this model we suggest extending underrepresented classes in the training data in order to deal with imbalances.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2020. *Speech and Language Processing*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

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

Ruoyao Yang, Wanying Xie, Chunhua Liu, and Dong Yu. 2019. BLCU_NLP at SemEval-2019 task 7: An inference chain-based GPT model for rumour evaluation. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1090–1096, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2015. Analysing how people orient to and spread rumours in social media by looking at conversational threads.

A Data distribution

Table 4: Distributions: Training, validation and test set

Dataset	**Comment (%)**	**Query (%)**	**Deny (%)**	**Support (%)**
Training set	2907 (64.3 %)	358 (7.9 %)	344 (7.6 %)	910 (20.1 %)
Validation set	778 (74.2 %)	71 (6.8 %)	106 (10.1 %)	94 (9.0 %)
Test set	771 (72.3 %)	92 (8.6 %)	62 (5.8 %)	141 (13.2 %)