# Advanced ML - Final Project

Nadav Elyakim, 205702368 and Or Shoham, 305073330

Anchor paper: BUT-FIT at SemEval-2019 Task 7: Determining the Rumour Stance with Pre-Trained BERT

Reichman University

July 2022

## 1. Introduction

We chose the paper named "BUT-FIT at SemEval-2019 Task 7: Determining the Rumour Stance with pre-trained Deep Bidirectional Transformers. Fighting false rumors on the internet is a tedious task. Sometimes, even understanding what an actual rumor is about may prove challenging. And only then one can actually judge its veracity with appropriate evidence. Interest in automated verification of rumors has deepened, as research has demonstrated the potential impact of false claims on important political outcomes. Rumour verification is characterized by the need to consider evolving conversations and news updates to reach a verdict on a rumor's veracity. The paper we chose tries to better classify those Rumours. More precisely, it takes part in a competition called "SemEval 2019 Task 7: RumourEval 2019: Determining Rumour Veracity and Support for Rumours, Subtask A". This competition focused on classifying whether posts from Twitter and Reddit support, deny, query, or comment on a hidden rumor, the truthfulness of which is the topic of an underlying discussion thread. This paper used the most recent BERT architecture (as of 2019) along with ensembling methods to build an end-to-end system. Their system has reached the F1 score of 56.7 % on the provided test data (61.67 % by using Ensembling). They finished in 2nd place in the competition. We reviewed related works done in the field and also go over the implementation of the chosen paper. Then we reconstruct the system and reached the F1 score of 57.56 % even slightly above the anchor paper.

## 2. Related work

### 2.1. Rumour Stance

The goal of rumor stance classification task is to determining the type of stance that each individual post expresses over the disputed veracity of a rumor. This task began to develop in the SemEval 2016-2017. Following the competition, more and more neural networks began to be developed in order to solve the problem while using different methods models. Poddar et al. used CNN to encode the text of each posts, and introduced LSTM and attention mechanism to judge the stance (1). Ma et al. (2) mixed stance classification

and rumor detection tasks into a multi-task model using 2 LSTM layers to model the propagation sequence of two tasks separately, and provided shared parameters through another LSTM layer.

### 2.2. previous competitions

The first competition which related to the competition from 2019 which we discuss in this report is from SemEval-2016 Task 6: Detecting Stance in Tweets (3). This task was focused in detecting stance from tweets: given a tweet and a target entity (person, organization, etc.), automatic natural language systems must determine whether the tweeter is in favor of the given target, against the given target, or whether neither inference is likely. Finally, through the benchmarking that the competition organizers developed, they overcame the other competitors. They developed SVM classifiers using hand-made features and word embeddings.

The second competition is quite similar to the competition this report is about. Subtask A - classifying if a tweet in a conversation threads either supporting, denying, querying or commenting on a rumour.Sub-task B - predict the veracity of a given rumor. The winner paper (4) used a LSTM-based sequential model that, through modelling the conversational structure of tweets, which achieves an accuracy of 0.784 on the RumourEval test set outperforming all other systems in Subtask A.

### 2.3. BERT

The anchor paper we chose is mainly based on a pre-trained BERT model. BERT was created and published by researchers at Google AI Language (5). BERT is at its core a transformer language model with a variable number of encoder layers and self-attention heads. The pre-trained BERT model can be fine-tuned with just one additional output layer.It can be used for question answering and language inference, without substantial task specific architecture modifications. The pre-train BERT first trained on the concatenation of BooksCorpus (800M words) and English Wikipedia (2,500M words).

## 3. Data

### 3.1. Dataset

To understand the model, we must first understand the data structure that used to train the model, since it can be confusing and not so intuitive: First, there are source posts that address certain rumors. Rumors can be true, false, or unverified. There are reactions/replies to each post that indicate whether the user supports (S), denies (D), comments (C) or queries (Q) rumours about the source post. There are tree-shaped relationships between the reactions/replies and the relevant source post as as can be seen in the example below.

The dataset comes from two different sources - Twitter and Reddit and also split to train set, development set and test set. There are 327/38/81 training/development/test tree-structured discussions containing 5217/1485/1827 training/development/test examples (source/reactions/replies)

The Twitter training and the development dataset used for the RumourEval 2019 is the same as used in RumourEval 2017. The new data added in this round is new Twitter test data and new Reddit data (including test, dev and train). More information about

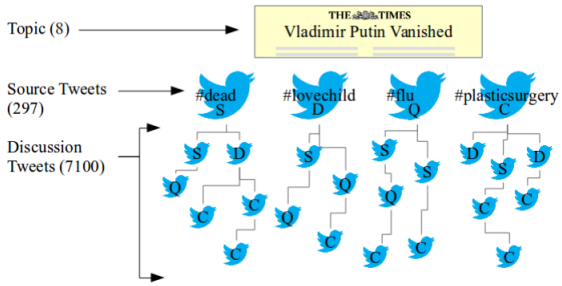


Figura 1: Example of the structure of a rumor.

the data, their origin, and how they were tagged - can be found in articles published for the competition (6).

### 3.2. Pre-processing

Pre-processing plays an important role in the model the paper suggests and we reconstructed. we followed the steps as detailed in the article - Implementation of this section can be found in our notebook under the chapter "Data loaded and preprocessing"[1]. First We loaded the different set of the data (source/reactions/replies) and extracted from each post only the relevant information (body text, id, source/replies, branches structure, author, set type, label, etc.). We combined all the data into one file in the form of a dictionary that is divided into training, development and test data. Each set type contains source posts, each with its own responses/replies. We saved the data file we created for more convenient use in further development.

Next, we replaced URL's and mentions with special tokens *URL* and *mention* (using tweet-processor). We splitted each post into sentences (using SpaCy) and [EOS] token to indicate the end of each sentence. As tokenizer we used the Hugging Face PyTorch re-implementation of BERT. The tokenizer lower-cases the input and applies the WordPiece encoding to split input words into most frequent n-grams present in the pre-training.

Hand-crafted features - while reconstructing the model we tried to implement many hand-crafted features as the anchor paper writers also did. We tried jointly learned POS, NER, and dependency tag embeddings, as well as the third segment embeddings and ect. As stated in the anchor article we also have come to a conclusion that non of the features led to a statistically significant improvement. We decided to use only the features indicating positive, neutral, or negative sentiment.

## 4. The model and paper reconstruction

As used in the anchor paper we employed the pre-trained BERT model [2] The model is first trained on large corpus of English data in a self-supervised fashion. This means it was pre-trained on the raw texts only, with no humans labelling in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pre-trained with two objectives:

[1] Model reconstruct-https://colab.research.google.com/drive/1_V2BH74Q5xaleOAIt8jJQzAgaHJM-CDS#scrollTo=SLu5QmNZoy41.

[2] Anchor paper model implemention https://github.com/MFajcik/RumourEval2019.

1. Masked language modeling (MLM): taking a sentence, the model randomly masks 15 percentage of the words in the input then run the entire masked sentence through the model and has to predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally mask the future tokens. It allows the model to learn a bidirectional representation of the sentence.

2. Next sentence prediction (NSP): the models concatenates two masked sentences as inputs during pre-training. Sometimes they correspond to sentences that were next to each other in the original text, sometimes not. The model then has to predict if the two sentences were following each other or not.

Model input - as we mentioned, the original input of the pre-train BERT is composed of two documents. Therefore, as $document1$ we use concatenation of the source and the previous post (left empty in case of the source post being the target post) and the target post used for $document2$: $[CLS]source-post+previous-post[SEP]target-post[SEP]$.


```
[{'branch_id': '0.0',
  'id': 0,
  'issource': 1,
  'raw_text': 'Appalled by the attack on Charlie Hebdo in Paris, 10 - probably journalists - now confirmed dead. An attack on free speech everywhere.',
  'raw_text_prev': '',
  'raw_text_src': '',
  'spacy_processed_text': 'Appalled by the attack on Charlie Hebdo in Paris , 10 - probably journalists - now confirmed dead . [EOS]\nAn attack on free speech everywhere . [EOS]',
  'spacy_processed_text_prev': '',
  'spacy_processed_text_src': '',
  'stance_label': 0,
  'tweet_id': '552788945017516032'},
 {'branch_id': '0.1',
  'id': 1,
  'issource': 0,
  'raw_text': '@tnewtondunn You guys should put one of their cartoons - maybe the last one they tweeted - on your front page tomorrow.',
  'raw_text_prev': 'Appalled by the attack on Charlie Hebdo in Paris, 10 - probably journalists - now confirmed dead. An attack on free speech everywhere.',
  'raw_text_src': 'Appalled by the attack on Charlie Hebdo in Paris, 10 - probably journalists - now confirmed dead. An attack on free speech everywhere.',
  'spacy_processed_text': '$MENTION$ You guys should put one of their cartoons - maybe the last one they tweeted - on your front page tomorrow . [EOS]',
  'spacy_processed_text_prev': 'Appalled by the attack on Charlie Hebdo in Paris , 10 - probably journalists - now confirmed dead . [EOS]\nAn attack on free speech everywhere . [EOS]',
  'spacy_processed_text_src': 'Appalled by the attack on Charlie Hebdo in Paris , 10 - probably journalists - now confirmed dead . [EOS]\nAn attack on free speech everywhere . [EOS]',
  'stance_label': 1,
  'tweet_id': '552789083211460608'},
 {'branch_id': '0.2',
  'id': 2,
  'issource': 0,
  'raw_text': "@mjhsinclair @tnewtondunn Sadly, since none of the UK press dared reprint the Jyllands-Posten cartoons, I think that's very unlikely.",
  'raw_text_prev': '@tnewtondunn You guys should put one of their cartoons - maybe the last one they tweeted - on your front page tomorrow.',
  'raw_text_src': 'Appalled by the attack on Charlie Hebdo in Paris, 10 - probably journalists - now confirmed dead. An attack on free speech everywhere.',
  'spacy_processed_text': "$MENTION$ $MENTION$ [EOS]\nSadly , since none of the UK press dared reprint the Jyllands - Posten cartoons , I think that 's very unlikely . [EOS]",
  'spacy_processed_text_prev': '$MENTION$ You guys should put one of their cartoons - maybe the last one they tweeted - on your front page tomorrow . [EOS]',
  'spacy_processed_text_src': 'Appalled by the attack on Charlie Hebdo in Paris , 10 - probably journalists - now confirmed dead . [EOS]\nAn attack on free speech everywhere . [EOS]',
  'stance_label': 1,
  'tweet_id': '552789720540119040'}]
```


Figura 2: Three examples of the final input that enters the model (from the first branch).

The above is based on the assumption that the stance of discussion's post depends only on it-self, on the source thread post and on the previous thread post.

Input tokens are represented by jointly learned token embeddings $E_t$, segment embeddings $E_s$, capturing whether the word belongs into document1 or document2, and positional embeddings $E_p$. The input representation is obtained by summing input embedding matrices $E = E_t + E_s + E_p \in R^{Lxd}$ (see figure 3), $L$ being the input length and $d$ the input dimensionality.

The fine-tuning of BERT is done using only the [CLS] token level output and passing it through two dense layers (dense layer - a layer that is deeply connected with its preceding layer) yielding posterior probabilities (see figure 4). A weighted cross-entropy loss is used to ensure a flat prior over the classes.

The model implemented in PyTorch with Hugging Face re-implementation and "BERT-large-uncased" setting, pre-trained using 24 transformer layers, having the hidden unit size of $d = 1024$, 16 attention heads, and 335,145,988 parameters. Batch size of 32 exam-

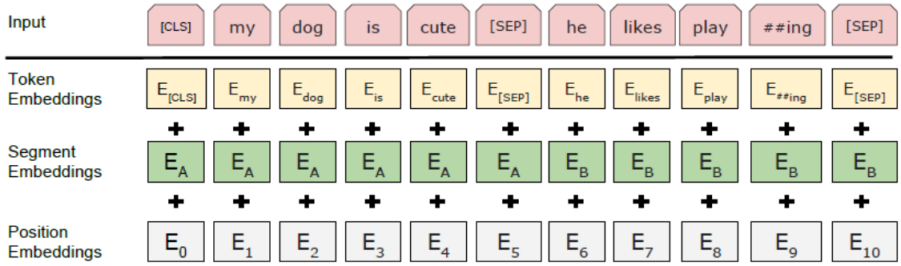


Figura 3: Example of input embeddings.

ples and other hyperparameters as same as in the BERT paper. We trained the reconstruct model on google colab on GPU with HIGH-RAM.

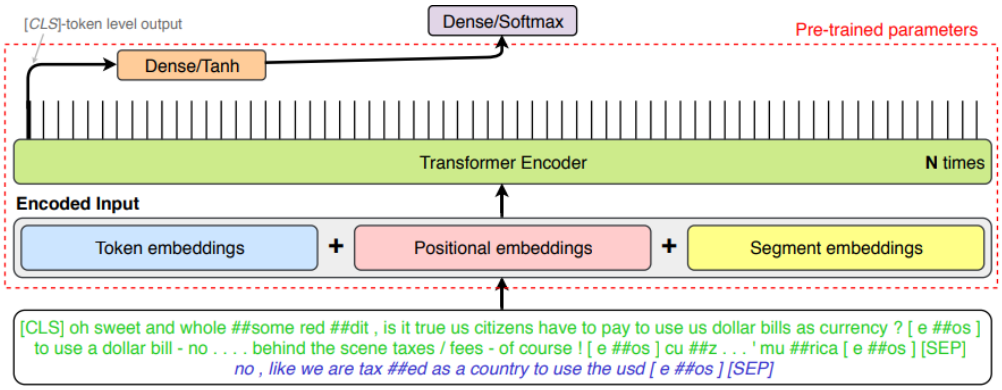


Figura 4: An architecture of BUT-FIT's system.

Ensembling - In order to increase model performance and yet avoid overfitting the authors of the anchor paper trained 100 models with different learning rates. They created 4 fusion mechanism (ensembles) each was built by 17 models. The best results fusion, "$TOP - N_s$çhooses 1 model randomly and add it to the ensemble. Then, it randomly shuffles the rest of the models and tries to add them into the ensemble one at the time, while iteratively calculating ensemble's F1 by averaging the output pre-softmax scores. If a model increases the total F1 score, the model is permanently added to the ensemble. The process is repeated until no further model improving the ensemble's F1 score can be found. The Ensembling method requires strong processing abilities, a lot of memory and days of training. We implemented the model itself as we described above, but we didn't proceed with Ensembling implementation due to the reasons outlined above. Our approach was to use "fit multiplerather than Ensambling - we trained models until we reached at least the F1 score of the anchor paper or until RAM finished. Each trail of training model was trained with a different seed and learning rate.

## 5. Results and conclusion

In order to evaluate the model performance macro-averaged F1 was used since "comment" class is very dominates, as well as being the least helpful type of data in establishing

rumour veracity. In The paper the results are compared to 3 baselines - "branch-LSTM", "FeaturesNN", and "BiLSTM+SelfAtt". The model achieved better performance than all three with macro F1 of 56.70 % on the test set without using Ensembling method and 61.67 % with. In our reconstruct we achieve macro F1 of 57.56 % on the test - slightly higher than the results shown in the paper.

| | **Acc-test** | **F1-dev** | **F1-test** |
|---|---|---|---|
| **Anchor paper** | 84.08 | 56.24 ± 9e3 | 56.70 ± 3e2 |
| **Our results** | 81.68 | 83.305 | **57.569** |

The "Win trail"with $lr = 4e - 06$ which was the first to achieve at least the results of the anchor paper gets his pick after 2-3 ephoch. We assume that this is because we used a pre-trained model and only doing a fine-tuning.

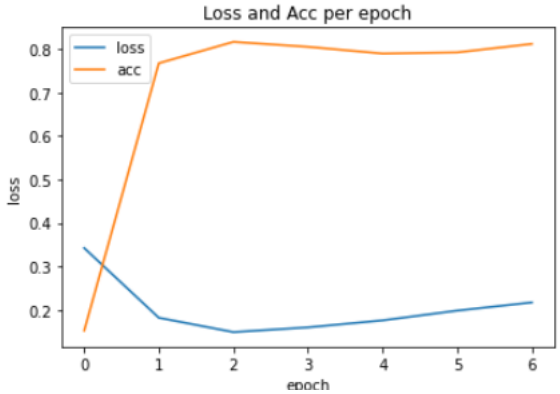


Figura 5: Loss and Acc per epoch for the best trail

To summarize the first part, we managed to get the results anchor article. The final implementation is quite clear and not complex. Only the source post of discussion, the previous post and the target post are used and it is not required to extract and invent additional features. The model is based on a pre-trained model with minor changes and not very long training. However the results are not so high. This may be caused by data - The label distribution for each of the datasets.The next section will expand on this topic.

# 6. Innovative part

## 6.1. Introduction

In this part, we will try to improve the paper's best result. The paper we chose uses Bert which is a very advanced transformer architecture. Moreover, the article details additional techniques that push the results to a very high level. Despite these results, the paper didn't delve into the label distribution for each of the datasets (train, validation, and test). One major challenge, which wasn't addressed in our paper, is that there are considerably more non-relevant replies (comments) than informative ones (supports and denies), making the task highly imbalanced (see figure 7). It is possible that by trying to improve the balance we can get better results.

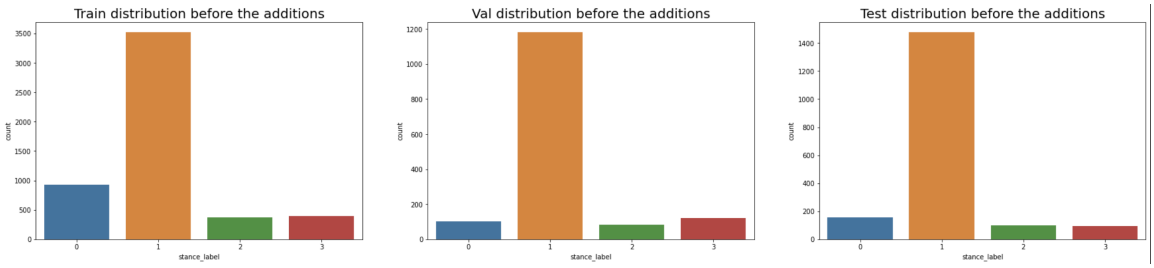


Figura 6: The anchor datasets distribution - train, validation, and test. highly imbalance datasets. Support = 0, Comment = 1, Deny = 2, Query = 3

The following table shows the f1 score for each label for the best results we achieved by reconstructing the anchor paper:

| | **Support** | **Comment** | **Deny** | **Query** |
|---|---|---|---|---|
| **Precision** | 0.41 | 0.90 | 0.0 | 0.43 |
| **Recall** | 0.48 | 0.86 | 0.0 | 0.47 |
| **F1 score** | 0.36 | 0.95 | 0.0 | 0.39 |

It can be seen that the model predicts the samples classified as 'comment' very well, but in the other labels, the results are poorly classified. In particular, the model doesn't predict any tweets that their label is 'deny' even though these appear a significant number of times in the test set. In this task, we are aiming to improve the performance of the informative minority classes ('support' and 'deny'). We experiment with traditional methods for imbalanced data treatment and used some external resources. In addition, we created another BERT-based model with a different pre-trained network and more fully connected linear layer that tries to improve the results even higher - our framework architecture is presented in figure 8.

## 6.2. Related works

The paper we chose finished second in the competition, so one of the first references we researched was the paper that finished in first place (7). This paper proposes an inference chain-based system, which fully utilizes conversation structure-based knowledge. They also expanded the training data in minority categories to alleviate class imbalance like we are aiming to do. In fact, we used the same additional data they used to improve our article results.

Another interesting paper that also relates to our competition (8) is experimenting with some methods for our imbalanced datasets with feature- and BERT-based classifiers. They show that by balancing the classes their model not only has higher performance for the minority classes but also has higher overall performance (in terms of macro-F1 and geometric mean recall – GMR). In our research, we used RUS/ROS techniques since they show a significant improvement in the results of this article.
RUS - randomly discards samples in the majority class so that the class proportions can be balanced. Generally, it is computationally more efficient than over-sampling because it reduces the training data, although it may lead to under-fitting.

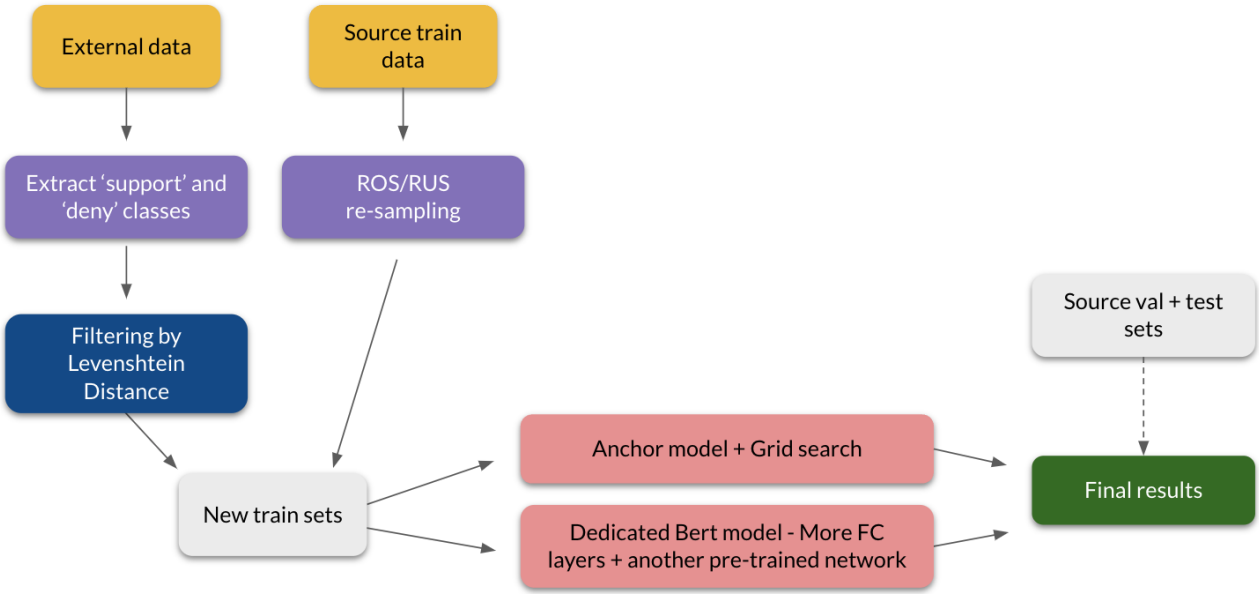


Figura 7: Our framework research architecture

ROS - re-balances the class proportions through randomly replicating samples in the minority classes. However, these replications can increase the possibility of over-fitting.

### 6.3. Part 1 - Adding new data and resampling the anchor model

First, we downloaded the data of the SemEval 2019 competition (3). this data contains tweets and it is divided into 4 different classes (see figure 9). We are only interested in samples that contain 'support'/'deny' since these are the labels we lack to balance our dataset (label 'query' did not exist in this data).

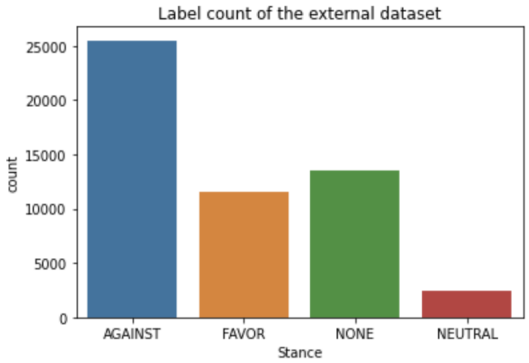


Figura 8: Class distribution of the SemEval-2016 Stance Dataset

After that, we did some pre-processing in the same way as in the first part in order to convert the new data into the necessary format (e.g. replace tokens, split sentences). This

dataset contains 53,000 samples while our training dataset contains only 5,216. This is a significant difference that could easily trigger an imbalance on the other side. Moreover, not all the existing tweets in the new data are similar in characteristics to the tweets in our data and we can conclude they were not taken from the same distribution. To overcome that challenge, we took only the most similar tweets by using the Levenshtein Distance method (9). The Levenshtein distance between two strings $a, b$ is given by $lev_{a,b}(|a|, |b|)$ where:

$$lev_{i,j} = \begin{cases} max(i,j), & \text{if } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1, \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{a_i \neq b_j} \end{cases} & , \quad \text{otherwise} \end{cases} \tag{1}$$

We divided the data into two sets - the 'support' set and he 'deny' set. We then iterated on each sample from the 'support' set and compared it to all the samples from the dev set that only contain the 'support' label (we used dev-set for efficiency reasons) and then took the average for each sample. We performed the same process for the 'deny' set and combined the most similar samples from each set (the samples with the shortest distance) with the original data.
Needless to say, the whole process was performed only on the train set, since we wanted to keep the validation set in the original distribution to check if we brought an improvement, which we will then expect to see on the test set as well. As we mentioned in section 6.2 we also used ROS/RUS methods because we saw that they bring significant improvement.
Since the test set is also unbalanced and tends toward the 'comment' class, our goal was not to achieve a perfect balance between the classes but to give a more significant representation to the minor classes (see figure 10). In conclusion, the different train sets we created (together with different learning rates and dropout) in order to try to improve the anchor results are those:

- Only RUS
- Only ROS
- RUS + New set
- ROS + New set

Since we intervene aggressively in the original dataset, it is very important to adjust the hyper-parameters. The learning rate and the dropout are important parameters that may contribute to under/over fitting problems. We changed and tested these parameters in order to improve the results.

### 6.4. Part 2 - Adding new data and resampling another Bert model with a new pre-trained network

In this section, we tried to use a bit different Bert network. We used 'roberta-large' (10) pretrained model which was trained on significant more data and outperformed 'bert-large-uncased' on several known datasets. We also added 2 more fully connected classification layers to predict the labels to improve the fine-tuning. The classification layers take as input the last hidden vector of the [CLS] token after the transformers layers.

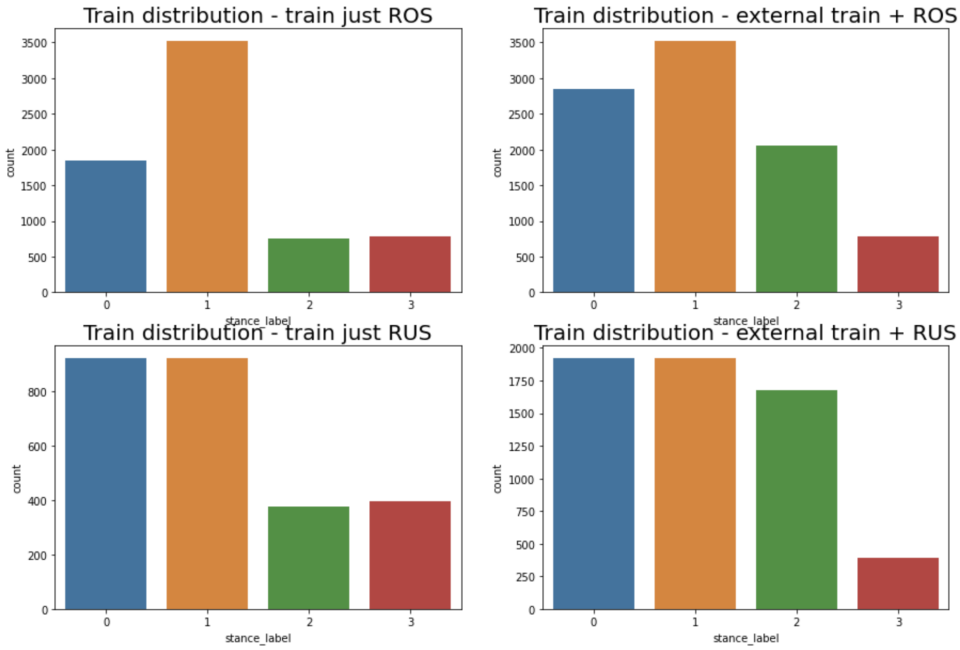


Figura 9: Labels distribution of each one of our new train dataset

The classification layers use dropout to prevent the model from over-fitting. In addition, we used the AdamW optimizer which yields better training loss and generalizes much better than models trained with Adam. We called this model 'BertFT' as it's doing more fine-tuning than the anchor model is doing.

The classification layers are:
1. ReLU and Dropout on Linear layer $1024 \rightarrow 512$
2. Linear layer $512 \rightarrow 4$ (the number of labels)

We did the same experiments as we did on the anchor model and trained the model with the 4 combinations we created as our training sets.

## 6.5. Results and conclusion

After running several experiments the next tables shows the main results we got after 7 epochs for each run (efficiency reasons).
Results of the total F1 score for part 1 - Adding new data and re-sampling the anchor model:

| **Sampling method / lr** | **9e-07** | **2e-06** | **4e-06** | **6e-06** |
|---|---|---|---|---|
| **Only ROS** | 0.462 | **0.592** | 0.584 | 0.039 |
| **ROS & New data** | 0.403 | 0.502 | **0.567** | 0.039 |
| **Only RUS** | 0.426 | 0.552 | **0.591** | 0.577 |
| **RUS & New data** | 0.435 | 0.430 | **0.503** | 0.215 |

Results of the total F1 score for part 2 - Adding new data and re-sampling another

Bert model with a new pre-trained network:

| **Sampling method / lr** | **1e-04** | **1e-05** | **1e-06** | **1e-07** |
|---|---|---|---|---|
| **Only ROS** | **0.379** | 0.271 | 0.221 | 0.221 |
| **ROS & New data** | 0.222 | **0.235** | 0.226 | 0.221 |
| **Only RUS** | 0.211 | **0.264** | 0.135 | 0.171 |
| **RUS & New data** | 0.221 | 0.223 | 0.227 | **0.246** |

As we can observe, we managed to improve the anchor best results. The combination of 'Only ROS' and learning rate of 2e-06 achieved an F1 score 0.592 compared to 0.567 which was achieved by the anchor paper. If we delve deeper we can observe the Precision, Recall, and F1 for each class in our results:

| | **Support** | **Comment** | **Deny** | **Query** |
|---|---|---|---|---|
| **Precision** | 0.77 | 0.88 | 0.42 | 0.54 |
| **Recall** | 0.33 | 0.93 | 0.35 | 0.68 |
| **F1 score** | 0.46 | 0.90 | 0.38 | 0.60 |

We can compare it to the anchor paper results (showen in section 6.1) and conclude the improvements came from all the informative labels! ('support', 'deny', and 'query') without damaging the results of the 'comment' label which still kept being the label with the highest score.
For example, label 'deny' achieved F1 score of 0 in the anchor results, and achieved F1 score of 0.38 in our results. That shows a significant improvement.
We can also observe that the anchor model (the first table) achieved better F1 overall compared to the new model we built (the second table). At first, we suspected it might be because of the difference of the pre-trained networks, so we trained each model (anchor model and the new model) with each of the pre-trained networks but those end up with no significant improvements. Then, we trained the best results we got from each model on more epochs and indeed, the new model achieved an F1 score of 0.494 (no change in the anchor model) which is still far from the anchor's best result although it's far closer than before. The differences may be because the anchor model implements connections between replies of the same thread which wasn't implemented in our new model, but more analysis should be done in order to figure out the reason.

In conclusion, we found our paper didn't address the classes balancing distribution and we tried to improve that area by adding new data and using re-sampling techniques. We managed to improve our paper's best result by better balancing the given training set. We still can achieve even higher results by mainly, conducting more experiments in order to get an accurate balance between each of the classes and also, by searching for more external datasets with similar characteristics to the dataset given in the competition.

We learned a lot from the process with an emphasis on different implementations that exists in NLP, techniques for balancing an unbalanced dataset, and methods for conducting correct and accurate research.

# Referencias


[1] Poddar, Lahari, Wynne Hsu, Mong Li Lee y Shruti Subramaniyam: *Predicting Stances in Twitter Conversations for Detecting Veracity of Rumors: A Neural Approach*. En *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, páginas 65–72, 2018.

[2] Ma, Jing, Wei Gao y Kam Fai Wong: *Detect Rumor and Stance Jointly by Neural Multi-Task Learning*. En *Companion Proceedings of the The Web Conference 2018*, WWW '18, página 585–593, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee, ISBN 9781450356404. https://doi-org.ezprimo1.runi.ac.il/10.1145/3184558.3188729.

[3] Mohammad, Saif, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu y Colin Cherry: *SemEval-2016 Task 6: Detecting Stance in Tweets*. En *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, páginas 31–41, San Diego, California, Junio 2016. Association for Computational Linguistics. https://aclanthology.org/S16-1003.

[4] Kochkina, Elena, Maria Liakata y Isabelle Augenstein: *Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM*. CoRR, abs/1704.07221, 2017. http://arxiv.org/abs/1704.07221.

[5] Devlin, Jacob, Ming-Wei Chang, Kenton Lee y Kristina Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. CoRR, abs/1810.04805, 2018. http://arxiv.org/abs/1810.04805.

[6] Gorrell, Genevieve, Kalina Bontcheva, Leon Derczynski, Elena Kochkina, Maria Liakata y Arkaitz Zubiaga: *RumourEval 2019: Determining Rumour Veracity and Support for Rumours*. CoRR, abs/1809.06683, 2018. http://arxiv.org/abs/1809.06683.

[7] Yang, Ruoyao, Wanying Xie, Chunhua Liu y Dong Yu: *BLCU_NLP at SemEval-2019 Task 7: An Inference Chain-based GPT Model for Rumour Evaluation*. En *Proceedings of the 13th International Workshop on Semantic Evaluation*, páginas 1090–1096, Minneapolis, Minnesota, USA, Junio 2019. Association for Computational Linguistics. https://aclanthology.org/S19-2191.

[8] Li, Yue y Carolina Scarton: *Revisiting Rumour Stance Classification: Dealing with Imbalanced Data*. En *Proceedings of the 3rd International Workshop on Rumours and Deception in Social Media (RDSM)*, páginas 38–44, Barcelona, Spain (Online), Diciembre 2020. Association for Computational Linguistics. https://aclanthology.org/2020.rdsm-1.4.

[9] Yujian, Li y Liu Bo: *A Normalized Levenshtein Distance Metric*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6):1091–1095, 2007.

[10] Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer y Veselin Stoyanov: *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. CoRR, abs/1907.11692, 2019. http://arxiv.org/abs/1907.11692.

[11] Fajcik, Martin, Pavel Smrz y Lukas Burget: *BUT-FIT at SemEval-2019 Task 7: Determining the Rumour Stance with Pre-Trained Deep Bidirectional Transformers*. En *Proceedings of the 13th International Workshop on Semantic Evaluation*, páginas 1097–1104, Minneapolis, Minnesota, USA, Junio 2019. Association for Computational Linguistics. https://aclanthology.org/S19-2192.

[12] Hochreiter, Sepp y Jürgen Schmidhuber: *Long Short-Term Memory*. Neural Computation, 9(8):1735–1780, 1997.

[13] Seo, Min Joon, Aniruddha Kembhavi, Ali Farhadi y Hannaneh Hajishirzi: *Bidirectional Attention Flow for Machine Comprehension*. CoRR, abs/1611.01603, 2016. http://arxiv.org/abs/1611.01603.