Stance detection:
SVM vs BERT vs "ChatGPT"

Simone Baccile, s.baccile@studenti.unipi.it
Project report of HLT course 2021/22, University of Pisa

June 1, 2023

Contents

1 Introduction

In this project, I've compared different classification approaches to solve the stance detection task. In particular, the aim is to compare the "standard" approaches used for this task in literature with the newest popular AI chatbot released a few months ago by OpenAI[1] ChatGPT. The idea is to find out if this new model can be a valid and easy-to-use replacement for standard models, such as supervised classifiers or transformers, especially accessible to non-technical people, without needing developer knowledge to solve the task, but simply chatting with a bot.

The results obtained are not that encouraging. In fact, old-fashioned approaches obtain better results compared to GPT models, even if the latter models gain in explainability of the response because it can be questioned about its classification response.

The report is organized into the following sections: in 2 there is a brief description of the state-of-the-art for what concerns stance classification; in 3, I'll talk about the data used in this project with related evaluation metrics; in 4, there is a list of the models compared for this task; in 5, I'll describe the setting of each experiment; in 6, there is the analysis of the results obtained; in 7 there are the conclusions with possible improvements to this project.

1.1 The task

The stance detection task is a classification of the stance of the author of a piece of text towards a target. The stance is the author's attitude and judgment toward a proposition. The simplest set of targets is {*Favor*, *Against*}, but there can be very different sets for the same task. For example, one could add *Neither*, if the standpoint of the author is neutral, or *Discussed* if the author talks about one argument but doesn't take any position about it. From a sociolinguistic perspective, it has been argued that there is no neutral stance as people tend to position themselves through their texts to be in favor of or against the object of evaluation[4]. This adds complexity to the task because the stance is not obvious from the texts and can depend on different combinations of historical context and social interactions. The stance detection can also be *Cross-targeted* when there are multiple topics for each text and each topic has its own set of targets. Indeed, the stance detection task is one, but then it can be differentiated in multiple ways based on how you define the target labels or on how you want to approach the classification.

What can be misconceived is the difference between stance detection and sentiment analysis, because both aim to classify text into two or more target labels. The difference is in the point of view. While stance detection evaluates a text concerning a topic, sentiment analysis classifies the polarity of a text in itself (the target labels become {*Positive*, *Negative*}). [1] shows that about 35% of the "*Favor*" stance tweets have a positive sentiment, while 67% of "*Against*" stance have negative sentiment (it also

[1] openai.com

shows that the neutral ones are misclassified). So there is no strong correlation between sentiment and stance.

This project focuses on classic stance detection, with two targets {*Favor*, *Against*} on texts with multiple topics. Even if topics are more than one, the set of labels is unique, hence there is not a set of label for each topic.

2 State-of-the-art

Interest in stance detection tasks has grown a lot in the last 6-7 years, probably thanks to the introduction of this task during SemEval-2016. This trend is also shown by different surveys [1] [2] [5]. What can be noticed is that there is a watershed between before and after the introduction of transformers architecture. In fact, until 2017-2018 the most used models for the stance detection task were the supervised classifiers like SVM and Random Forest (with a minority of CNN and RNN), from 2018-2019 the approach to this task has been completely overhauled in favor of transformers like BERT and its derivatives. Another aspect to be taken into account is the features used to train the classifier for supervised technique. From the research papers analyzed in these surveys, the most used features were N-grams, Bag-of-Words, TF-IDF, and POS tagging. In some cases were used extra features like the sentiment of the text or other topic-based features.

Another interesting aspect of the stance detection task is to "measure what counts". In [6] is shown how different evaluation metrics can change the results of a competition (in this case RumourEval task 2017 and 2019[2]). In particular, especially care should be taken when labels are unbalanced.

Last, but not least, [7] is the paper from which I took inspiration for this project. The authors applied ChatGPT to SemEval-2016 and P-Stance[3] datasets, obtaining similar performances compared to previous approaches, like BERT, and in some cases even improving them just with zero shot setup, without the need of pre-training.

3 Data

In this section I'll describe the dataset used for the experiments, the preprocessing of the data, and the metrics used for different model comparisons.

3.1 Dataset

The dataset used in this project is the *IBM Debater - Claim Stance Dataset* [3], which consists of 2394 claims taken from Wikipedia divided into 55 manually identified topics. The stance was annotated by hand for each claim and it can be *PRO* if the claim is in

[2] `https://alt.qcri.org/semeval2019/index.php?id=tasks`

[3] `https://github.com/chuchun8/PStance`

favor of the topic, and *CON* if it is against. Before proceeding with the experiments, I removed all the topics with less than 40 claims because otherwise, it would have been difficult to train the classifier without too many claims. Therefore, I obtained a dataset with 25 topics and 1788 claims, which I divided into training (2/3) and test sets (1/3), balancing the topics in both sets. The distribution of the two datasets is in Figure 1.

3.2 Metrics

According to [5], the most used metric to evaluate stance detection tasks is the F1-score over accuracy. For this project, due to some imbalanced distribution between PRO and CON labels for some topics, I decided to use the F1-score macro averaged, which is the mean between the per-class F1-score. Indeed, all decisions to choose the best model have been made considering this metric. However, accuracy, precision, and recall were also considered for overall performance evaluation.

4 Models

I compared four different models in this project: two supervised models and two transformers. In particular, the former models are taken from *scikit-learn*[4], and are the Multinomial Naive Bayes, used as a baseline to compare with other models, and the Support Vector Machine (SVC) because it is widely used for the stance detection task. The transformer models are from *HuggingFace*[5]. The two transformers used are the BERT model which is the current standard for this type of task, and the GPT2 model to emulate the behavior of ChatGPT.

4.1 Multinomial Naive Bayes

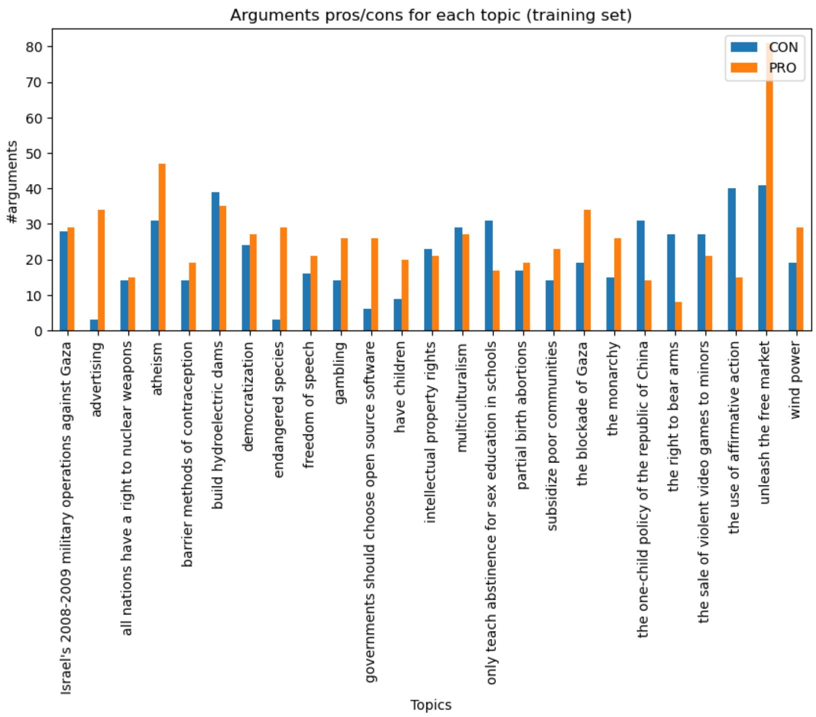
The Multinomial Naive Bayes is a variant of the Naive Bayes classifier used in text classification. It takes in input a vector of data features, in this project TF-IDF vectors, and applies the Bayes theorem to predict the probability of a given feature appearing in a specific class. It is a simple model but can be very effective for an easy text classification task.

4.2 SVM

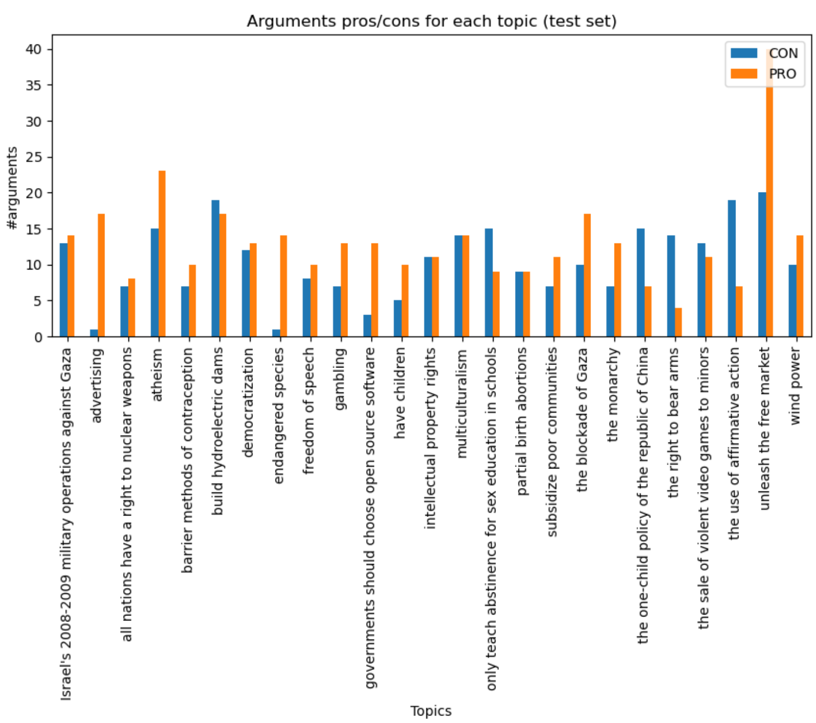
A support vector machine constructs a hyper-plane in a high dimensional space, which can be used for classification tasks. A good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. In this project, the prediction is given by $sign(w^T\phi(x)+b)$ where

[4] `https://scikit-learn.org/stable/`

[5] `https://huggingface.co/`



(a) Training set



(b) Test set

Figure 1: Splitting of the IBM Debater dataset in training and test set

w is the margin, $\phi(\cdot)$ is the kernel function, and b is the intercept, found by the SVM algorithm.

4.3 Fine-tuned BERT

The BERT model used in this project is the *bert-base-uncased*[6], which is a transformer model pre-trained on English corpus with almost 3300 million of words (Toronto BookCorpus and English Wikipedia). It was pre-trained using Masked Language Modeling as objective, which means that taking a sentence, it randomly masks some words and then tries to predict the masked words. The BERT model consists of 12 encoders with 12 bidirectional self-attention and has a total of 110 million parameters. To do the classification task, a linear layer is added on top of the pooled output. Because BERT learns the latent space of words in context, I've fine-tuned it with data for the stance detection task in order to optimize its performance.

4.4 P-tuned GPT2

The GPT model used for in this project is *gpt2*[7], which is a pre-trained model on English corpus with 8 million of web pages (Common Crawl and BookCorpus), but differently from BERT it is trained with the objective to guess the next word in a sentence. The main difference from BERT is that GPT is an auto-regressive transformer decoder which means that each token is predicted and conditioned on the previous token. The encoder is not needed because the decoder receives the previous token by itself. This makes GPT models very good for tasks like language generation, but less good for classification. The GPT2 model consists of 12 layers and 117 million parameters. Classification tasks are performed by GPT models using the last generated token, for this reason any padding should be done on the left and not on the right. This project aimed to highlight the differences between the standard approach to stance detection with ChatGPT, but because the access to the model behind ChatGPT (GPT3) is paid, I decided to emulate its behavior. ChatGPT is nothing more than a prompt completion model, so I can emulate it with a prompt-engineering technique. This is what I've done using the P-tuning. P-tuning is a prompt-engineering technique which uses a prompt encoder to insert anywhere in the sequence prompt tokens. It is more efficient than classical prompt tuning because the prompts are not manually crafted (which is the main difficulty when applying prompt-engineering techniques). To be fair it must be said that the GPT2 model I've used is not the same as ChatGPT, but it's its predecessor. In fact, GPT3 has far more parameters (175 billion!) and many more training data.

[6] `https://huggingface.co/bert-base-uncased`

[7] `https://huggingface.co/gpt2`

5 Experiments

The experiments were done in three phases. The first phase is the preprocessing phase, where I preprocessed the inputs of the training set to be fed to the models. The second phase is the training part in which the models have been trained or tuned using the training set. In the end, the models were evaluated on the test set with the metrics already described in Section 3, but more details about the results are postponed to Section 6. All the models were trained, tuned, and tested on 2,5 GHz Intel Core i5 dual-core with 4 GB 1600 MHz DDR3 of RAM. Due to time limitations and computational constraints, I haven't had a chance to do an exhaustive search for all the parameters.

5.1 Preprocessing

The preprocessing part can be divided into two different types based on the trained models.

- Supervised models: each argument of the training has been lowercase, tokenized in words using the *nltk* library [8], and spell corrected using *autocorrect* package[9]. Then I removed the punctuation and the stopwords, and finally, I used the Porter stemmer to obtain the word stem. Then I built the matrix of TF-IDF features to train the Multinomial Naive Bayes and the SVM. The details of the TF-IDF parameters are discussed in the next subsection.

- Transformers: for both models, I didn't manually preprocess the arguments, but used the corresponding tokenizers with padding. Because the GPT2 model hadn't a PAD token, I have added it assigning the same token id as the EOS token. The main difference in padding between BERT and GPT2 is that for the former the padding is on the right and for the latter it is on the left, due to their different classification methods.

5.2 Training

Also the training part follows two different approaches based on the trained model.

- Supervised learning: I used a grid search to find the best parameters for the TF-IDF and for the model itself. The grid search on the training set with 3-fold cross-validation. In the following, I have reported a list of parameters used in the grid search with a brief description.

  - **TF-IDF**:

    * *min_df*: threshold to ignore terms with document frequency less than it.

[8] `https://www.nltk.org/index.html`
[9] `https://github.com/filyp/autocorrect`

    - *max_features*: consider only top max_features terms when constructing the vocabulary.
    - *ngram_range*: what n-grams to extract.
  - **MNB**:
    - *alpha*: smoothing parameter.
  - **SVM**:
    - *C*: regularization parameter.
    - *kernel*: kernel type.
    - *degree*: degree of the polynomial kernel.
    - *gamma*: kernel coefficient.
    - *shrinking*: shrink heuristic flag.

- <u>Transformers</u>: for these models, I didn't do a grid search of the parameters, but I have just done the fine-tuning or the p-tuning with almost the default ones because they are a lot and it is really difficult to understand everyone's meaning. I split the training set into train (80%) and validation (20%) for tuning. A few parameters I changed are (in brackets the values used during the training):
  - *Batch size*: the batch size per CPU (16).
  - *Evaluation strategy*: when to do the evaluation ("epoch" means at the end of each epoch).
  - *Epochs*: number of epochs (5).
  - *Learning rate*: learning rate of the optimizer (2e-5).
  - *Weight decay*: weight decay to apply to layers (0.01).

5.3 Ablation study

What I wanted to investigate was also how important are the topics in the stance detection task. In literature, there is no unanimous opinion that decides whether it is better or not to use the topics with the arguments. So what I do is a sort of reverse ablation study. Firstly, I have done all the experiments only with the arguments as training input. Then I have done again the same experiments but with topics added to training input. I just concatenated each argument on its respective topic. For example, if one argument is "the most immoral acts in human history were performed by atheists" and its topic is "atheism", the new training argument will be "atheism the most immoral acts in human history were performed by atheists". From now on, I will refer to the first approach with the symbol **(A)** and to the one with topic+argument with **(TA)**.

6 Results and Analysis

In this section, there are the results obtained from the experiments evaluated on the test set. In the second part, there is an analysis part in which I tried to explain the results of those models using SHAP[10], a game theoretic approach to explain the output of machine learning models.

6.1 Results

In Tables 1 and 2, there are the results obtained from the experiments. In Table 1, there are the results of the supervised classifiers, the Multinomial Naive Bayes (MNB) and the Support Vector Machine (SVM). The column 'Best params' shows what are the best parameters selected by grid search for each model. The evaluation score is the macro F1 score, the grid search time is the time to find the best parameters, while eval time is the time to evaluate the model on the test set. In Table 2, there are the same results but for the transformer models. In this case, the column 'Tuning time' shows the time to fine-tune or P-tune the models. In Figures 2 and 3, there are the confusion matrices for the two types of experiments (A) and (TA). Detailed figures on the distribution of predictions through the different topics can be found in Appendix.

In Table 3, the results are resumed altogether considering only the evaluation metrics. All the metrics are macro averaged except the accuracy. What is highlighted from these results is that as expected SVM and BERT models work very well because they are state-of-the-art for the stance detection task. What surprised me is that the GPT2 model works very badly, even worse than the Multinomial Naive Bayes. Looking at the Figure 2 it can be seen that the GPT2 model tends to classify all the arguments with the PRO label. Probably the meaning is to find out how it predicts the label of the arguments, just looking at the last token not may be enough. For what regards the use of topics, they increase the score for the best models, but worse models like MNB and GPT2. The F1 score of SVM increases from 0.73 to 0.76, while for BERT it increases only by 0.1. You can notice that in this table there are two extra rows regarding ChatGPT. I took 50 arguments from the test set, and I asked manually to ChatGPT to detect the stance of those arguments, with and without the topics. Firstly, I sent only the arguments and the results are very bad, but I was still hopeful because ChatGPT suggested inserting also the topics, because otherwise, the stance detection is difficult. So I wrote again all the arguments with the topics but... the results were the same. I can reason about these bad predictions by saying that I didn't fine-tune ChatGPT for this task with data, so it wasn't able to correctly predict correctly these arguments, even though the results were in line with the P-tuned GPT2.

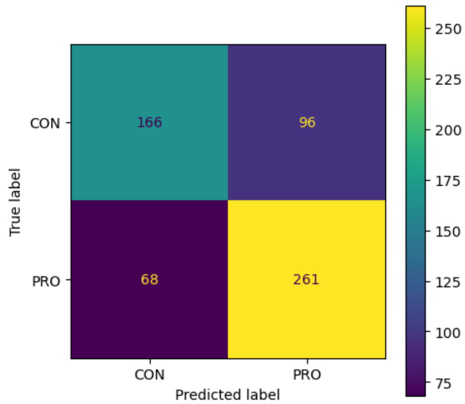
[10] `https://github.com/slundberg/shap`

Model	Best params	F1 train	F1 test	Grid search time	Eval time
MNB (A)	min_df=1 ngram=(1,3) alpha=0.14	0.64	0.72	≈ 25min	< 1sec
SVM (A)	min_df=5 ngram=(1,1) C=1.7 gamma='scale' kernel='rbf'	0.67	0.73	≈ 40min	< 2sec
MNB (TA)	min_df=2 ngram=(1,1) alpha=0.17	0.66	0.67	≈ 30min	< 1sec
SVM (TA)	min_df=1 ngram=(1,1) C=1.54 gamma='scale' kernel='poly' degree=2	**0.72**	**0.76**	≈ 40min	< 2sec

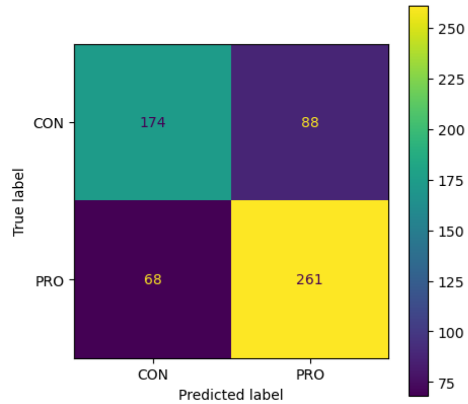
Table 1: Supervised classifier experiments. The best parameters comes from the grid search and the F1 score is the macro F1 score.

Model	F1 train	F1 test	Tuning time	Eval time
BERT (A)	**0.73**	0.71	≈ 30min	≈ 2min
GPT2 (A)	0.71	0.52	≈ 35min	≈ 4min
BERT (TA)	0.72	**0.72**	≈ 50min	≈ 2min
GPT2 (TA)	0.67	0.50	≈ 40min	≈ 4min

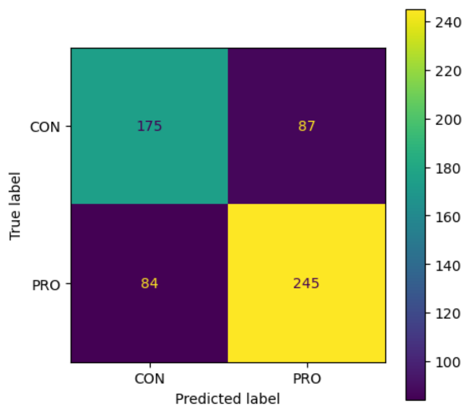
Table 2: Transformers experiments. Tuning time is intended a time of fine-tune the BERT model and P-tune the GPT2 model. F1 score is the macro F1 score.



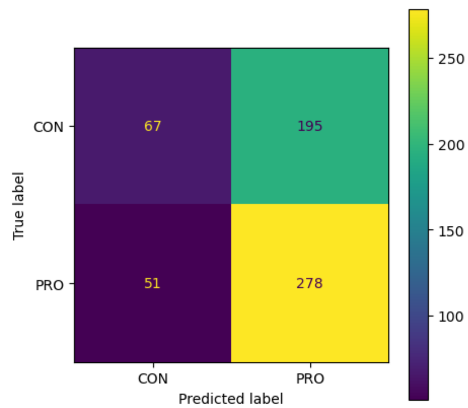
(a) MNB



(b) SVM



(c) BERT

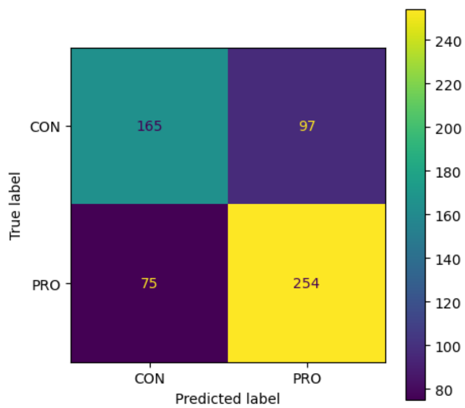


(d) GPT2

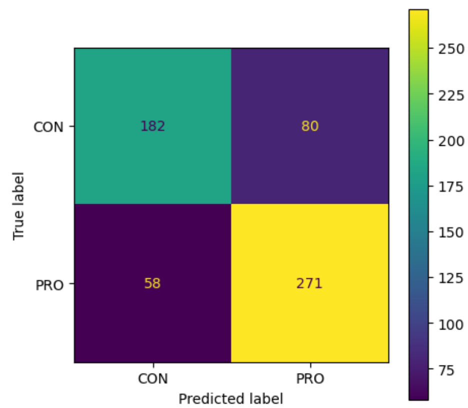
Figure 2: Confusion matrices of all (A) experiments.

Model	F1	Accuracy	Precision	Recall
MNB (A)	0.72	0.72	0.72	0.72
SVM (A)	0.73	0.74	0.73	0.73
BERT (A)	0.71	0.71	0.71	0.71
GPT2 (A)	0.52	0.58	0.58	0.55
MNB (TA)	0.67	0.67	0.67	0.66
SVM (TA)	**0.76**	**0.77**	**0.77**	**0.76**
BERT (TA)	0.72	0.72	0.72	0.72
GPT2 (TA)	0.50	0.50	0.52	0.52
ChatGPT (A)	0.39	0.44	0.40	0.43
ChatGPT (TA)	0.39	0.44	0.40	0.43

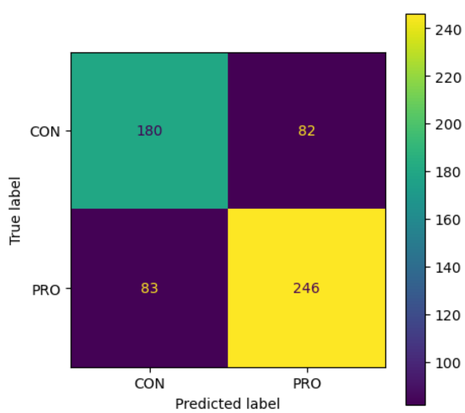
Table 3: Metrics evaluated on test set for all experiments. Except the accuracy all the metrics are macro averaged. ChatGPT wasn't evaluated on all the test set.



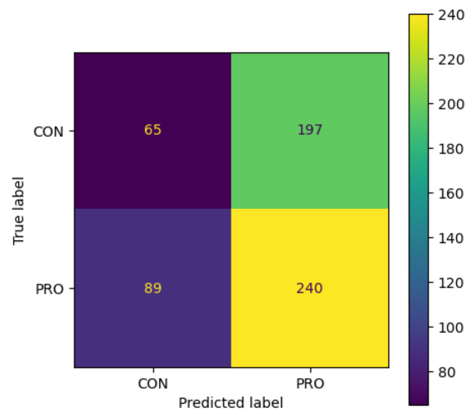
(a) MNB



(b) SVM



(c) BERT

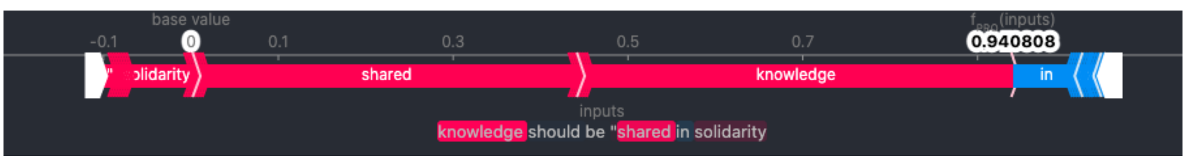


(d) GPT2

Figure 3: Confusion matrices of all (TA) experiments.

6.2 SHAP analysis

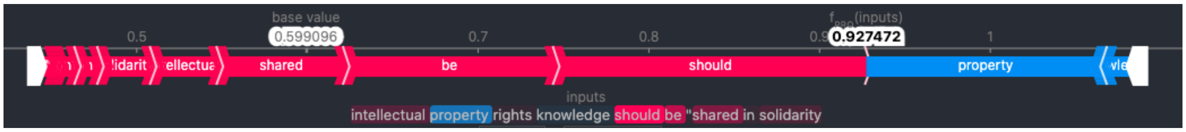
SHAP is the acronym for SHapley Additive exPlanations and uses the Shapley values to explain the outputs of machine learning models. The Shapley values is a concept used in game theory that involves gains and costs from several actors working in coalition to achieve a desired outcome. In this case, a Shapley value is assigned to each token (word) to understand how much it contributes to the final output (0=CON, 1=PRO). The Shapley values are not so expensive to compute when there are only a few or hundreds of features to compute the output. But when the number of features grows and is in the order of thousands or tens of thousands, this computation is pretty expensive. Ideally, one would like to compute the Shapley values with all the test sets, but since for this project I am in the second case with thousands of features, I computed the Shapley values only for 20 arguments from the test. That means that the results are not so reliable, but in any case, they could be useful to try to understand what is happening behind these models. In Figure 4, there are two arguments from (A) experiments classified by BERT. Shapley values assigned to these two arguments show that the words ‘knowledge’ and ‘shared’ tend to classify the sentence as ‘In favor’ while words like ‘self’ and ‘something’ weigh more towards ‘Against’. In the first case, it is more obvious that those words help to classify a PRO sentence, while the second case is less obvious. But, again, this is because the Shapley values should be computed on more arguments. In Figure 5, there are similar examples but with arguments from (TA) experiments. The figure highlights when the topics are too long they are not taken into consideration if not for a few first words. But in general what is pointed out by this analysis is that the words that weigh more in the classification are the verbs. Extra figures about the SHAP analysis can be found in Appendix.



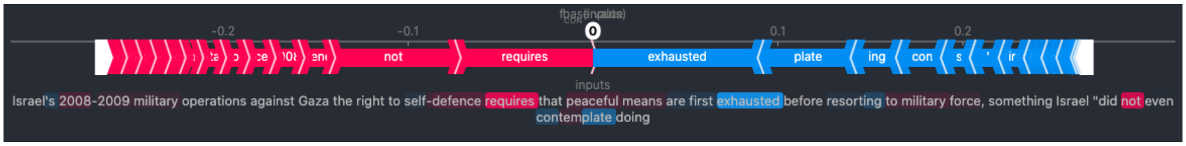
(a) PRO

(b) CON

Figure 4: BERT Shapley values for two arguments from (A) experiments.



(a) PRO



(b) CON

Figure 5: BERT Shapley values for two arguments from (TA) experiments.

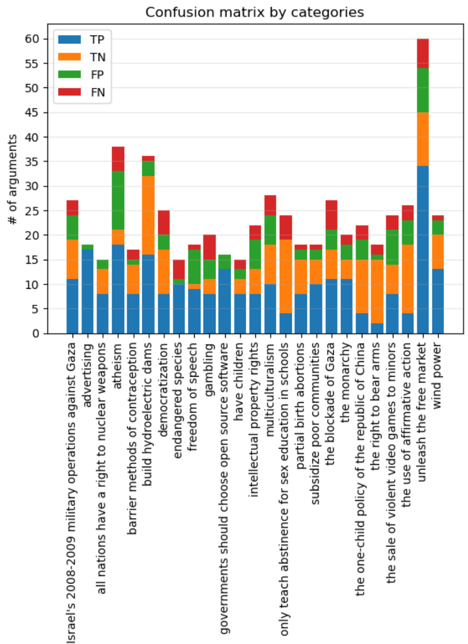
7 Conclusions

In this project, I've compared different models for the stance detection task on *IBM Debater - Claim stance dataset*. The models compared are Multinomial Naive Bayes, SVM, BERT, and GPT2. The results obtained from my experiments confirm the advantages of using SVM and BERT models for the classification tasks. In particular, the SVM model has a final macro F1 score of 0.73 which increases to 0.76 when the training is done with topics. The BERT model has a macro F1 score of 0.71 which increases to 0.72 with topics. Of course, I know that there are some limits with my approach. Above all, the lack of more in-depth tuning of the models due to computational power limitations. The second limit is due to the GPT2 model used because it is a predecessor of the current best models of OpenAI GPT3/GPT3.5. Probably the results obtained from these newer models would have been better, as proved by the paper which inspired this project. Strictly related to the first problem, an in-depth analysis of the Shapley values with more arguments would have helped to better explain the behavior of the models. Another interesting improvement would be to add extra features, like the sentiment of the arguments, and see if the score improves. But as the wise wizard Gandalf said: *"All we have to decide is what to do with the time that is given us"*.

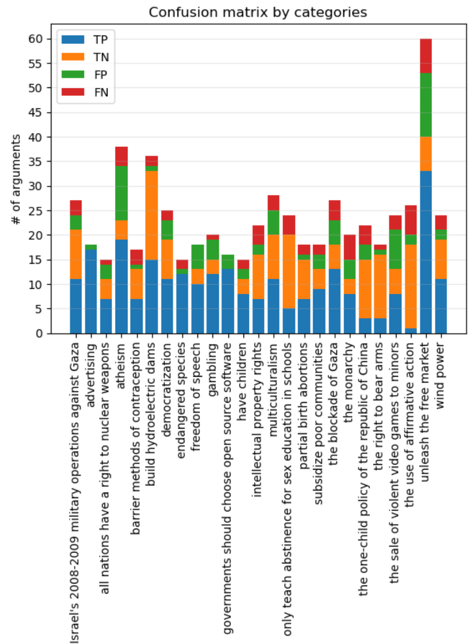
8 Code

The code of this project is freely available on GitHub in the following repository https://github.com/Simoniuss/HLT-project.

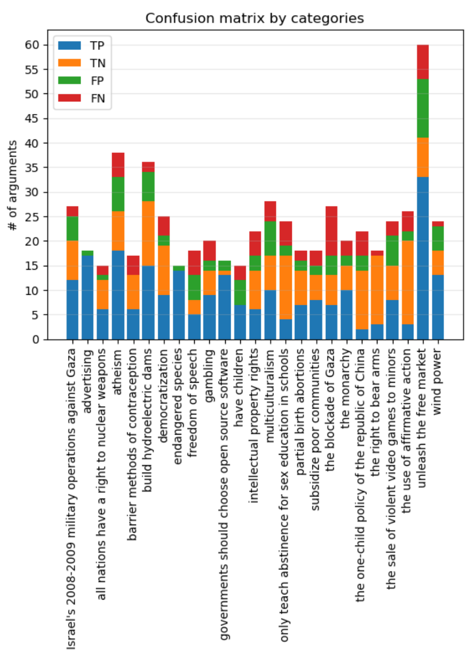
9 Appendix



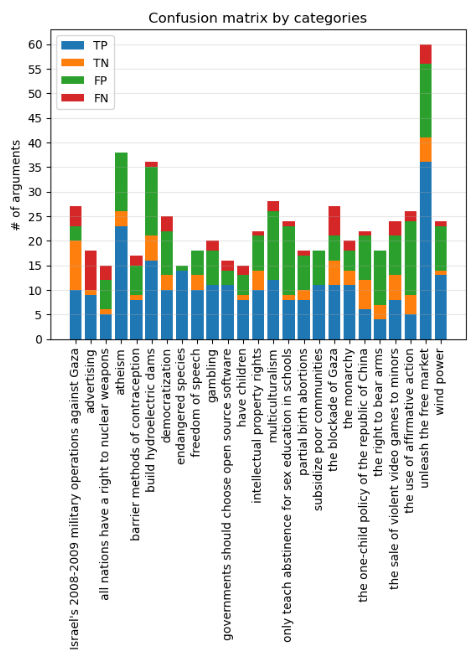
(a) MNB



(b) SVM

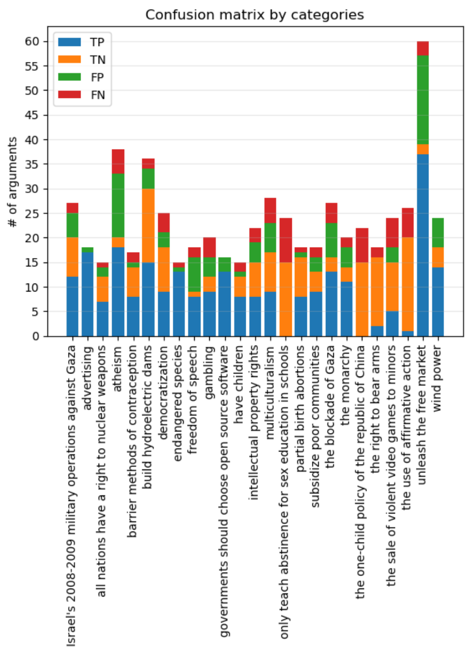


(c) BERT

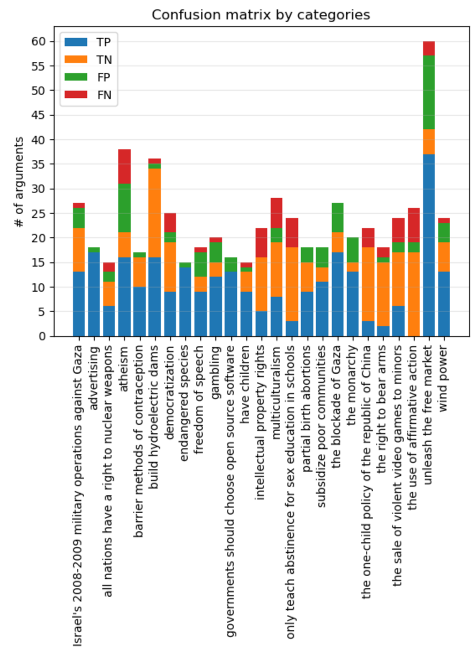


(d) GPT2

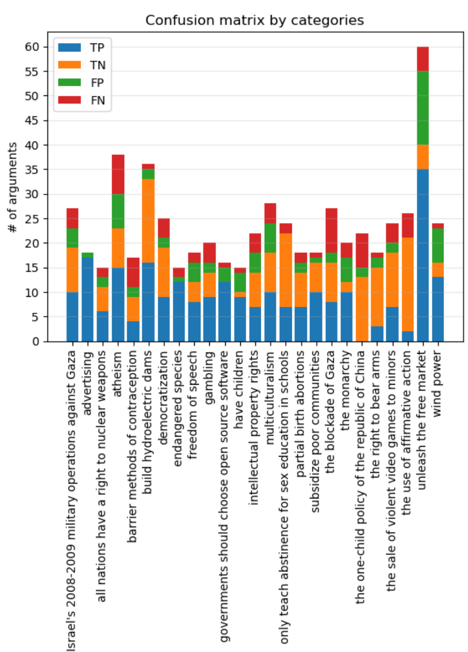
Figure 6: Classification results for each topic for (A) experiments.



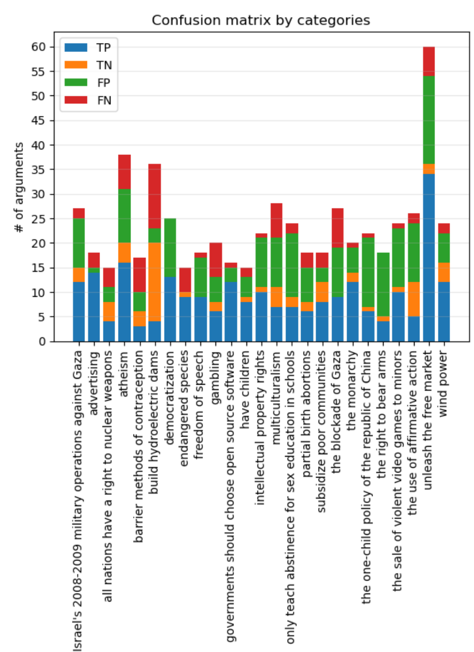
(a) MNB



(b) SVM

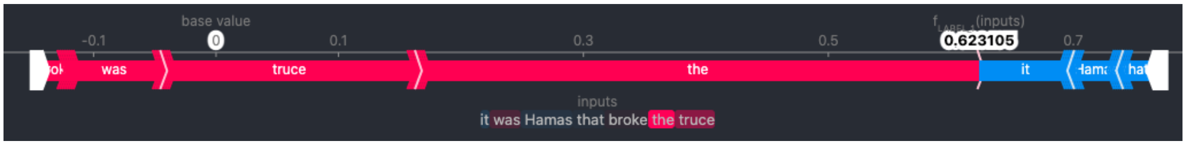


(c) BERT



(d) GPT2

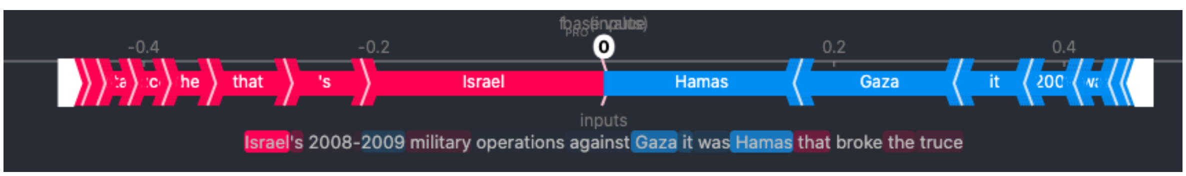
Figure 7: Classification results for each topic for (TA) experiments.



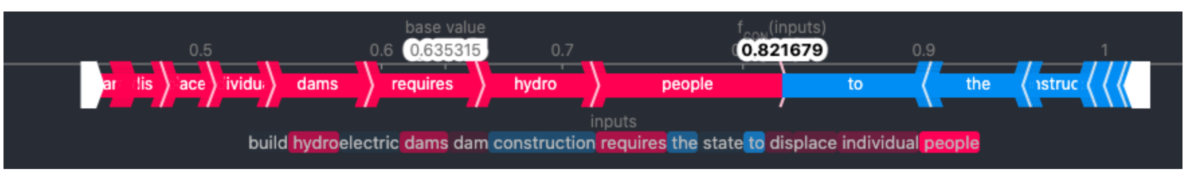
(a) PRO

(b) CON

Figure 8: GPT2 Shapley values for two arguments from (A) experiments.

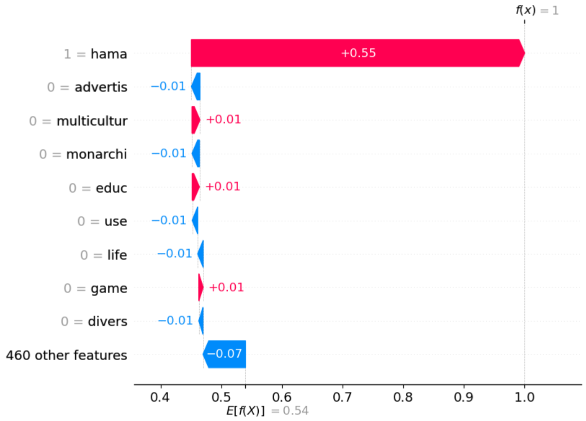


(a) PRO

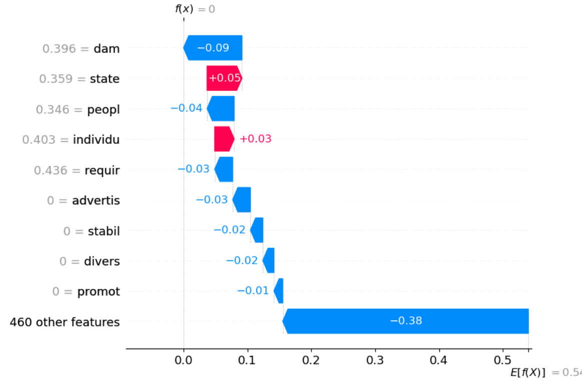


(b) CON

Figure 9: GPT2 Shapley values for two arguments from (TA) experiments.

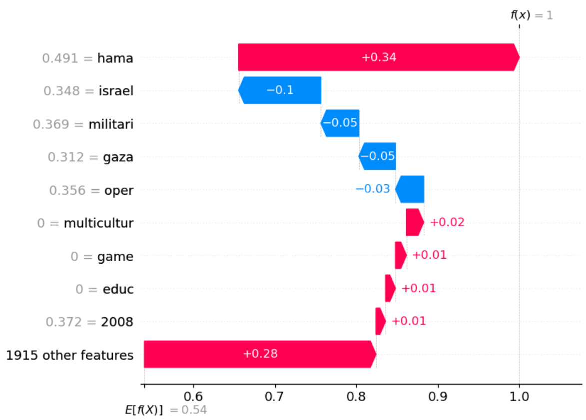


(a) PRO

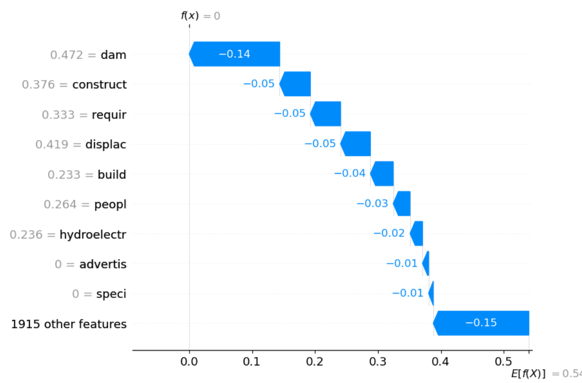


(b) CON

Figure 10: SVM Shapley values for two arguments from (A) experiments.



(a) PRO



(b) CON

Figure 11: SVM Shapley values for two arguments from (TA) experiments.

References

[1] Abeer AlDayel and Walid Magdy. "Stance Detection on Social Media: State of the Art and Trends". In: *CoRR* abs/2006.03644 (2020). arXiv: 2006.03644. URL: https://arxiv.org/abs/2006.03644.

[2] Nora Alturayeif, Hamzah Luqman, and Moataz Ahmed. "A Systematic Review of Machine Learning Techniques for Stance Detection and Its Applications". In: *Neural Comput. Appl.* 35.7 (Jan. 2023), pp. 5113–5144. ISSN: 0941-0643. DOI: 10.1007/s00521-023-08285-7. URL: https://doi.org/10.1007/s00521-023-08285-7.

[3] Roy Bar-Haim et al. "Stance Classification of Context-Dependent Claims". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 251–261. URL: https://aclanthology.org/E17-1024.

[4] Alexandra Jaffe. *Stance: Sociolinguistic Perspectives*. Oxford University Press, June 2009. ISBN: 9780195331646. DOI: 10.1093/acprof:oso/9780195331646.001.0001. URL: https://doi.org/10.1093/acprof:oso/9780195331646.001.0001.

[5] Dilek Küçük and Fazli Can. "Stance Detection: A Survey". In: *ACM Comput. Surv.* 53.1 (Feb. 2020). ISSN: 0360-0300. DOI: 10.1145/3369026. URL: https://doi.org/10.1145/3369026.

[6] Carolina Scarton, Diego Silva, and Kalina Bontcheva. "Measuring What Counts: The Case of Rumour Stance Classification". In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 925–932. URL: https://aclanthology.org/2020.aacl-main.92.

[7] Bowen Zhang, Daijun Ding, and Liwen Jing. *How would Stance Detection Techniques Evolve after the Launch of ChatGPT?* 2023. arXiv: 2212.14548 [cs.CL].