NLP project work report

Michele Luca Contalbo
Yiran Zeng

February 10, 2023

Executive Summary

The project is focused on the "CheckThat!: Enabling Automatic Identification and Verification of Claims in Social Media" challenge. The challenge is composed of three NLP tasks: Check-Worthiness Estimation in Tweets and Political Debates, Detecting Previously Fact-Checked Claims in Tweets and Political Debates, Fake News Detection.
The problem we tackled is a subtask of the second one. More precisely, we focus on creating an NLP system which is able to detect previously fact-checked claims in tweets [1].
We developed several baselines for this challenge on previous projects: in this one, we analyse the contribution of each word to sentence similarity by using SHAP and KernelSHAP.

Background

The dataset used for training the models is composed by *<tweet,vclaim>*, namely the tweet and its corresponding verified claim.

tweet	*A number of fraudulent text messages informing individuals they have been selected for a military draft have circulated throughout the country this week.*
vclaim	*The U.S. Army is sending text messages informing people they've been selected for the military draft.*
tweet	*Fact check: The U.S. Army is NOT contacting anyone regarding the draft. If you are receiving texts, phone calls or direct messages about a military draft, they are not official communications from the U.S. Army pic.twitter.com/3S32De8ekP — U.S. Army CGSC (@USACGSC) January 8, 2020*
vclaim	*The U.S. Army is sending text messages informing people they've been selected for the military draft.*
tweet	*The US drone attack on #Soleimani caught on camera.#IranUsapic.twitter.com/TvRkHvlgby — Olaudah Equiano® (@RealOlaudah) January 6, 2020*
vclaim	*A video shows the U.S.-ordered drone strike that killed Iran Gen. Qassem Soleimani.*

Given these *<tweet,vclaim>* pairs, we want to highlight which words of *tweet* contribute the most to the pairs' similarity. An algorithm that is capable of providing such explanations is **SHAP**.
SHAP is a unified approach to explain the output of any machine learning model. It is based on the idea of Shapley values from cooperative game theory and provides a way to fairly distribute the contribution of each feature to a model's prediction for a specific instance. These values quantify the incremental change in the model's prediction if a particular feature's value was altered. SHAP values account for both the feature's impact on the prediction and its interaction

with other features: a single missing feature may not alter the result, but combined with others, it may change the output in a significant way. SHAP values can be used to understand the behavior of individual predictions, compare the impact of features across instances, and to analyze the entire model globally. In our case, we consider as *feature* each possible word and analyse **locally** and **globally** its contribution.
In this project we also do the same analysis using **KernelSHAP**. KernelSHAP is a kernel-based approximation of the SHAP values which are estimated by applying a weighted linear regression to the predictions of the model: the coefficients for the linear regression represent an approximation of the Shapley values. The *kernel* part of KernelSHAP refers to the use of a kernel function that gives more weight to small and big *coalitions*, where a *coalition* is a possible combination of the input features. The idea is based on the fact that such coalitions provide more insight on the most important features than coalitions that consider half of the input features.

System description

For this project we developed **SHAP** and **Kernel-SHAP** and tested them on **TF-IDF**, **LSTM** and **SBERT** over <*tweet,vclaim*> pairs.
All SHAP-based approaches suffer from high computational time if the number of features is very high. Indeed, it would be necessary to extract all the feature permutations (*coalitions*) and calculate the Shapley values over them. This problem is NP-complete and intractable, so we focus our analysis only on a subset of all the possible coalitions. In detail, we follow the same *kernel* idea explained in the previous section by sampling small and big coalitions, i.e. the ones that should provide the best insights over our input data. For this we define a parameter `level` that determines how many features are selected for each coalition and we base our sampling on that. Moreover, we consider also their negation, i.e. the coalitions considering all the features except the ones chosen before: in this way we ensure that our data follows the same intuition of the *kernel* trick.
Usually in SHAP-based methods the features that have not been selected are given a random value. This is due to the fact that, most of the times, it is not possible to change the input dimension due to how the model is structured. If we change the non-selected feature with a fixed value we might be altering our results if the inserted values are not independent from the other features. Thus, usually random values are inserted, since with a very high number of coalitions the randomness of the feature will not alter the results. Another problem arises in our case, i.e. with a very small amount of coalitions. Indeed, random values would probably change the variance of our data and be harmful for the explanations.
Due to all these considerations, since in our case features are the words inside the tweets, we opt to replace non-selected features with the empty string: we believe that it should not change the variance of our data and it should not provide great insights to the other features.
For example, considering the following input sentence taken from our dataset

Here's how @BernieSanders took Nevada from @HillaryClinton

Our approach would generate the following coalitions if `level` = 1

[1,0,0,0,0,0,0] [0,1,1,1,1,1,1] [0,1,0,0,0,0,0] [1,0,1,1,1,1,1] ...

generating the following outputs

Here's
how @BernieSanders took Nevada from @HillaryClinton
how
Here's @BernieSanders took Nevada from @HillaryClinton
...

which are used to estimate the Shapley values.

Analysis and results

Local Interpretability

In this section we compare the scores obtained by **TF-IDF**, **LSTM** and **SBERT** on two samples The patterns shown in these two cases extend to the other samples of the test set. Moreover, we show the results obtained by SHAP and the ones obtained by KernelSHAP.
The `<tweet,vclaim>` pair considered in the first case is

Tweet: *Is @jacindaardern willing to denounce this legislation of child sexual abuse?*
Vclaim: *In August 2018, French politicians passed a law which stated that a child is capable of consenting to having sex with an adult.*

SHAP

Figure 2 shows the SHAP values obtained for the tweet. As expected, TF-IDF gave higher scores to words appearing in vclaim due to the *term frequency* of its formula. Indeed, it is a syntactical model and does not take into account the semantics of the words. For example, *legislation* and *law* appear in tweet and vclaim respectively: even though they have similar meaning, *legislation* has a close to zero SHAP value due to the fact that it does not syntactically match *law*. In these cases, words that are used in various contexts like *is* and *to* may receive an higher SHAP value, even though they are very common and thus should not be too useful. Apparently, the *inverse document frequency* part of the algorithm did not manage to significantly reduce those values, even though it can be noted its effect by comparing the values with the one obtained by *child*.
With reference to LSTM and SBERT, even though the second is way better then the first for sentence similarity, they give very similar explanations. Indeed, they do not suffer from the same drawbacks of TF-IDF: this can be noted by the fact that they are able to give higher scores to words even though they do not appear

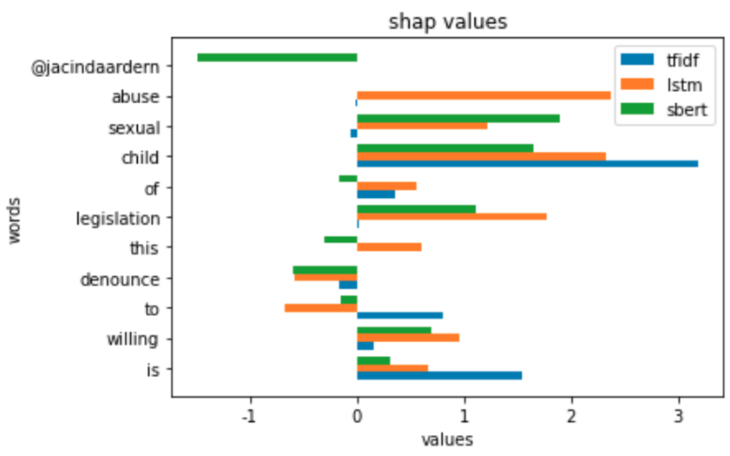

Figure 1: Comparison between TF-IDF, LSTM and SBERT

in vclaim, hence the scores are given based on semantics. LSTM, compared to SBERT, is still giving too much weight to stopwords like *is* and *this*. This can be symptom of their perfomance differences.
Note that *@jacindaardern* does not have scores for TF-IDF and LSTM. This is due to the fact that TF-IDF and LSTM have used a different tokenization technique than the one used by SBERT

KernelSHAP

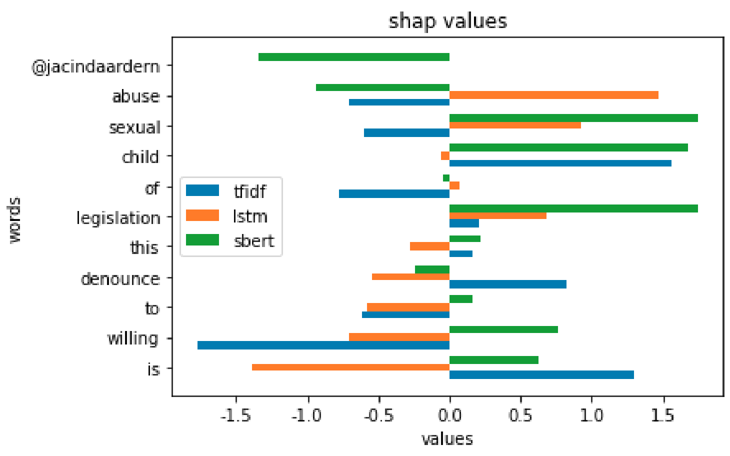

Figure 2: Comparison between TF-IDF, LSTM and SBERT for KernelSHAP

KernelSHAP tends to obtain more negative contributions compared to SHAP. The difference is due to the fact that KernelSHAP tends to obtain more accurate estimates with fewer evaluation of the original model [7], but at the same time it remains an approximation method based on linear regression.
TF-IDF still seems to provide more weight to words that appear in both sen-

tences. We can see that, for both SBERT and LSTM, the results are similar to the ones obtained with the other approach.

Global Interpretability

The SBERT model was chosen due to its better performance in calculating similarity compared to other models. The goal of the analysis was to gain global interpretability of the model.
The SHAP values were normalized using the z-score method. A counter was implemented to select feature words that appeared at least a certain number of times for analysis. The reasoning behind this was that feature words with high SHAP values in multiple samples and a high average SHAP value across the samples can be considered positive features that contribute to the model's similarity calculation. On the other hand, negative features that appeared in multiple samples were also selected for analysis.

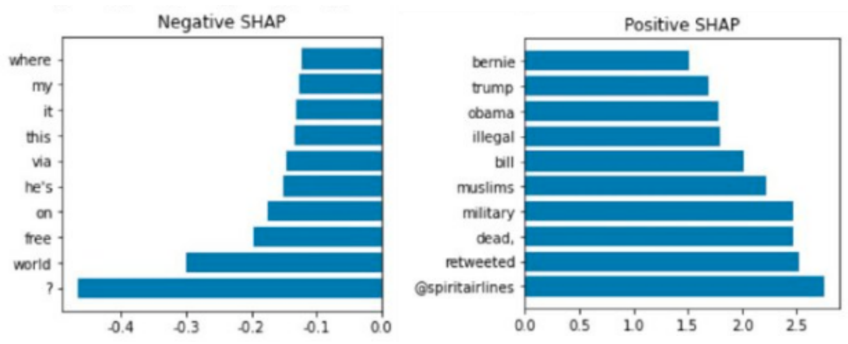

Figure 3: Most negative and positive contributions

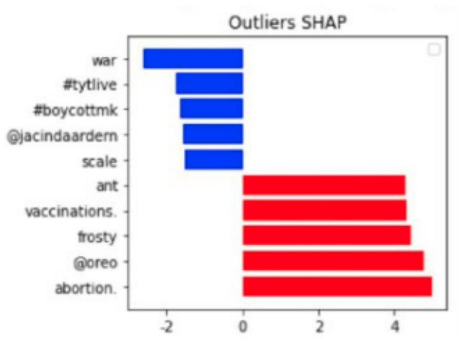

Figure 4: Outliers

The analysis showed that positive feature words, which appeared at least three times in the dataset, were mostly nouns and may represent the subject of the tweet, which helps the model match the corresponding vclaim. The negative feature words, which also appeared at least three times in the dataset, were mostly prepositions and pronouns, which may confuse the model's calculations by matching both correct and wrong vclaims with the tweet. The analysis also showed that the "?" symbol had the greatest negative impact, which makes sense as most of the vclaims were based on fact-checked declarative sentences. Outliers, which were the maximum and minimum values filtered by sorting the SHAP mean value, were words that appeared only once in the whole dataset and may not contribute significantly to the overall analysis. However, for a single sample, the appearance of these feature words can often be a decisive factor in the calculation. These outliers were typically @ and hashtags, which are commonly considered to be the subject and focus of a tweet.

Bibliography

[1] S. Shaar, F. Haouari, W. Mansour, M. Hasanain, N. Babulkov, F. Alam, G. Da San Martino, T. Elsayed, P. Nakov (2021) *Overview of the CLEF-2021 CheckThat! Lab Task 2 on Detecting Previously Fact-Checked Claims in Tweets and Political Debates*, CEUR Workshop Proceedings.

[2] N. Muennighoff (2022) *SGPT: GPT Sentence Embeddings for Semantic Search* Cornell University.

[3] A. Chernyavskiy, D. Ilvovsky, P. Nakov (2021) *Aschern at CheckThat! 2021: Lambda-Calculus of Fact-Checked Claims* CEUR Workshop Proceedings.

[4] D. Whitfield (2021) *Using GPT-2 to Create Synthetic Data to Improve the Prediction Performance of NLP Machine Learning Classification Models* Cornell University.

[5] L. Reynolds, K. McDonell (2021) *Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm* Cornell University.

[6] C.J.C. Burges (2010) *From RankNet to LambdaRank to LambdaMART: An Overview* Microsoft Research.

[7] S.M.Lundberg, S.I.Lee (2017) *A unified approach to interpreting model predictions* CoRR.