NLP project report

Michele Luca Contalbo
Yiran Zeng

January 20, 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 contribute by providing several syntactical and semantical baselines: BM25, TF-IDF, LSTM, non-finetuned SBERT and SGPT. Furthermore, we finetune SBERT with 2 different ranking losses and apply data augmentation by using GPT.
We show that, while GPT seems to be worse than BERT when it comes to sentence embeddings, a combination of the 2 can be beneficial, by using GPT with few-shot prompting to create synthetic data, and then train the BERT model on the augmented dataset.
Furthermore, we analyse the co2 consumption of each model. During inference time, the difference between syntactical and semantical approaches tends to favour syntactical models.
Several approaches have tried to combine syntactical and semantical models, by creating a multi-stage NLP system. In our case, we combine SBERT trained on the augmented dataset with TF-IDF using Lambda-MART, following the algorithm used by A.Chernyavskiy [3].
In the end, we test whether the model improves by using *<vclaim>*, *<vclaim,title>*, *<vclaim,title,subtitle>* combinations. More details on this in the Background section

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.*

The dataset provides more information about the vclaim, which can be concatenated to get richer sentence embeddings.

title	*Doctor Saves Baby, Baby Saves Doctor*
subtitle	*An incredible story about a doctor and a paramedic who saved each other's lives 30 years apart.*
vclaim	*A paramedic saved the doctor who had saved his own life as a baby.*

Since our task is to find vclaims that verify a set of tweets, we can assemble the inputs into a triplet as `<tweet, positive_claim, negative_claim>` and use a siamese neural network architecture to train a baseline model. We want the triplet loss to force the tweet and its corresponding claims together and push the irrelevant claim apart in the latent space. For the processing of text data, where text at different locations is correlated, traditional neural network algorithms can only process the input on a one-to-one basis and cannot take care of the different impact of individual inputs on the entire text. Therefore, to better match our task requirements, we will use recurrent neural networks because they allow for the persistence of information. To further avoid the problem of gradient disappearance or explosion during the training process, we use an LSTM layer. For this base model, the initial embeddings for each word are taken from `glove`.

Most of the approaches tend to use BERT as pre-trained model for sentence embeddings and ranking.
A.Chernyavskiy et al [3] developed a pipeline using TF-IDF, Sentence-BERT (`stsb-bert-large`) fine-tuned and LambdaMART [6]. Their architecture follows a two-stage approach, by feeding to the LambdaMART ranker the top 20 vclaims obtained from sentence-BERT and TF-IDF. Mixing syntactical and semantical models by using MART proved to be beneficial for the overall score,

specially by using LambdaRank gradient computation, which can be set to optimize specific metrics.
Recently, there have been several attempts to use GPT models for tasks in which, generally, BERT excels. N.Muenninghoff [2] has tried to train generative models for sentence similarity tasks, proving that these models can reach good results, almost like the ones obtained by BERT. However, they need to use a larger amount of parameters: hence, BERT is still preferred.
There have been attempts to use generative models to create new synthetic data [4]. Indeed, they are very useful in scenarios where the data is too few. When generating new data, we tried to test the behaviour of the ranking model when using data generated in a zero-shot fashion or in a few-shot. Lately, there have been debates regarding the difference between the two approaches: some experiments have shown that the gain of using few-shot prompting is reduced when more context is given in the zero-shot prompts [5]. In our experiments, we used the template `A tweet is provided:{} This is the claim that verifies the tweet:{}` for zero-shot, while `Tweet:{} Response:{}` for few-shot prompting. The latter still gives better results, but this could be a matter of further study.

System description

In this project, we try to recreate A.Chernyavskiy et al [3] model, also by using better weights pre-trained in ranking tasks (`all-mpnet-base-v2`), and apply data augmentation with GPT.
The dataset has been split in train, validation and test dataset, with respectively 800,200 and 200 samples.
We apply some preprocessing on the input text data: tweets are often full of urls, emojis and escaped characters that are noise and do not add any semantic meaning to our models. We decided, instead, to keep tags and hashtags, since they are often used as subjects or objects of the sentences. Indeed, their deletion decreases the performance of BERT's model.
No basic text preprocessing has been applied, like stemming, lemmization or normalization, because `all-mpnet-base-v2` has been pre-trained on data without the use of these techniques.
The training data is then fed to two different models: TF-IDF and SBERT with GPT augmented data. The synthetic data obtained from GPT was, most of the times, non consistent wrt tweets and verification claims, meaning that the vclaims were not actually verifying their corresponding tweets, thus adding noise to our dataset. For this reason, we used a finetuned lighter GPT model (`EleutherAI/gpt-neo-125M`) to create tweets and a heavier, not finetuned one (`EleutherAI/gpt-neo-1.3B`) to create verification claims that are consistent with their tweets. Hence, the generation of the verification claims is dependent on the tweets created previously, by creating 16-shot prompts and then asking the model to create a new vclaim based on the synthetic tweet.
Then, the model selects the top-50 vclaims from each tweet in the validation set: we take the corresponding scores given by tf-idf and train the LambdaMART model by providing the score and ranking position for each one of the two models. While the positions for SBERT are bound between 1 and 50, the positions

for tf-idf can span the whole length of the verification claim dataset.
For the standard SBERT fine-tuning **without** data augmentation, we used a learning rate of 1e-2 for Mega Batch Margin Loss and Multiple Negative Ranking Loss, with a weight decay of 1e-2 in both cases. Using higher learning rates with lower weight decays led the models to overfit and to achieve bad results. The batch sizes used are of size 50 and 64 respectively.
The GPT model for tweet creation has been trained with a learning rate of 1e-5 and number of epochs equal to 40.
The SBERT data augmented model has been trained with a batch size of 32, 50 epochs and learning rate and weight decay equal to 1e-2. The choice of using different batch sizes for different models is due to computational constraints.
The LambdaMART model has been trained with a maximum depth of 3, eta equal to 0.2 and with a pairwise ranking objective.
Other than this, we also trained several neural baselines to obtain the sentence embeddings: one approach does not involve learning and it is simply the mean of Glove embeddings (`Glove mean`), while another follows the same principle but has two dense layers with ReLU in between (`Glove dense`). Then we test with a simple LSTM layer, bidirectional LSTM (`biLSTM`) and two bidirectional LSTMs with last embedding as sentence embedding (`2biLSTM-last`) and mean of intermediate outputs (`2biLSTM-intermediate`). Finally, we also test bidirectional GRU with batch normalization (`2biGRUbn`) and without (`2biGRU`). All the neural baselines have been trained with triplet loss with semi-hard negative mining.

Analysis and results

In Table 1, it can be seen the different models' MAP and MRR scores by only using `<tweet,vclaim>` pairs.

Baselines

Between syntactical models, it can be seen a huge difference between `BM25` and `TF-IDF`: after applying stemming, we have seen that BM25 has significantly improved while TF-IDF did not (Table 2). Apparently, `scikit-learn`'s `TfidfVectorizer` function already applies some preprocessing, so the comparison of the algorithms is biased.

Model	**MAP@1**	**MAP@2**	**MAP@5**	**MRR**
`BM25`	0.5477	0.5955	0.5995	0.6107
`TF-IDF`	0.7085	0.7559	0.7601	0.7672

Table 2: Syntactical models after stemming

Model	MAP@1	MAP@2	MAP@5	MRR
Glove mean	0.1709	0.1951	0.1999	0.2080
Glove dense	0.2563	0.2839	0.2912	0.3022
LSTM	0.2814	0.3166	0.3254	0.3379
biLSTM	0.3467	0.3886	0.3998	0.4078
2biLSTM	0.2990	0.3346	0.3411	0.3463
2biLSTM-mean	0.1809	0.2228	0.2278	0.2489
2biGRU	0.2714	0.3099	0.3177	0.3343
2biGRUbn	0.2814	0.3216	0.3337	0.3450
BM25	0.3668	0.3932	0.4020	0.4148
TF-IDF	0.7010	0.7471	0.7544	0.7594
SGPT	0.8618	0.8932	0.8980	0.8896
SBERT	0.9020	0.9384	0.9394	0.9300
SBERT-MBML	0.9020	0.9242	0.9300	0.9207
SBERT-MNRL	0.9271	0.9485	0.9533	0.9425
SBERT-MNRL-GPT zero-shot	0.9322	0.9569	0.9581	0.9480
SBERT-MNRL-GPT few-shot	0.9523	0.9686	0.9696	0.9603
LambdaMART	**0.9623**	**0.9795**	**0.9817**	**0.9720**

Table 1: Comparison between models

As expected, applying stopwords elimination did not bring any improvement due to the IDF part in both algorithms, but stemming and other preprocessing procedures seem to be the driving factor for the difference in scores.

Instead, for neural baselines, we obtain an expected improvement over Glove mean and Glove dense by using RNN-based models. Between these models, biLSTM provides the best performance: adding another LSTM layer with more weights (2biLSTM) seems to add unnecessary complexity to the learning process, while taking the average of the intermediate results (2biLSTM-mean) plummets the scores as expected (the last output has already taken the context of the sentence since it has been processed *left-to-right* and *right-to-left* two times each). The tests on the GRU layers show that batch normalization between GRU layers seems to be beneficial to the output embedding.

Pre-trained models

It can be seen that semantical pretrained models outscore syntactical ones by a large margin. The usage of generative models for data augmentation seems beneficial, considering also that the SBERT-MNRL-GPT model has been trained on a **smaller batch size**. Hence, better values could be achieved by using triplet based losses with a larger range of samples, or by using hard and semi-hard negative mining.
The chosen hyperparameters avoid overfitting. Figure 1 shows the benefits of

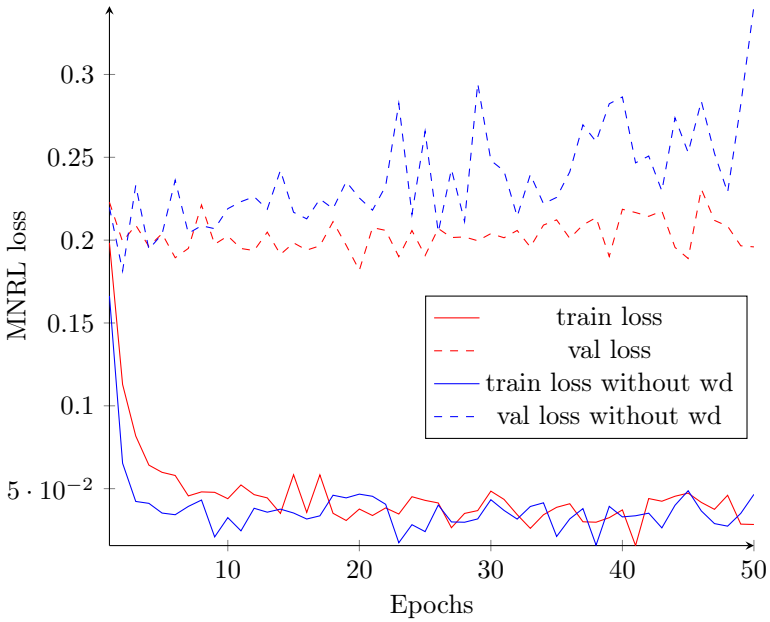


Figure 1: Effects of using weight decay on loss

using weight decay (*wd*) in `SBERT-MNRL`, pattern which can be seen in all the other models. The increase of the validation loss impacts the overall MAP scores, which, in the last epochs, are lower than the scores of the non finetuned `SBERT` model.

The combination of `SBERT-MNRL-GPT few-shot` and `TF-IDF` results seem to improve the scores in the `LambdaMART` model, providing, as far as we know, a new **state of the art** result on the dataset. The two models, indeed, seem to make different mistakes and predictions, which can be beneficial for a Gradient Boosted algorithm.

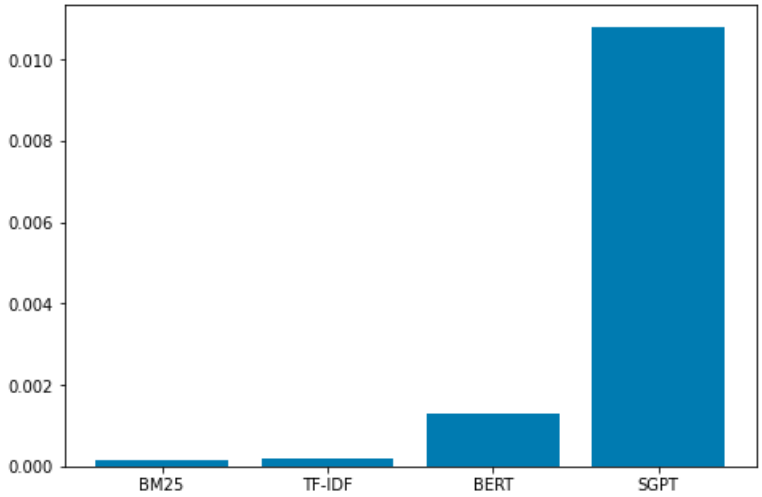
Analysis of vclaim augmentation

Table 3 shows that the best results on the training dataset are achieved with just `vclaim`. Interestingly, `title` and `subtitle` improve `TF-IDF` (due to having more repetitions of important words), but does the opposite for `SBERT`. We think that this is probably due to instability of the learning process, thus there is some variance in the scores between different training attempts, but also because the added information may be redundant or even noisy for neural models.
Regarding the neural baselines, it is possible to note the same pattern obtained from pre-trained models: the improvement is very low and sometimes augmentation makes the ranking worse. RNN-based models seem to gain less from

the augmentation, while `Glove mean` and `Glove dense` show some slight improvement. The only case in which RNN-based models improve with the vclaim augmentation is with `2biGRUbn`, mainly because it makes the learning process less delicate with reference to the choice of hyperparameters. In general, augmenting the vclaim sentences seems to enhance the performance of syntactical models more than semantical ones.

CO2 emissions

The following histogram shows the co2 emissions for different models at inference time.



We can see that, during inference, semantical models seem to be worse and produce more emissions. SGPT has a much worse performance due to the size of its network. Since training and data augmentation procedures may be very demanding with regards to co2 emissions and computational costs, it would be advised to give up on precision and fall back to easier models, like TF-IDF or not fine-tuned SBERT, which already achieve good scores.
LambdaMART does not significantly change the co2 emissions.

Model	MAP@1	MAP@2	MAP@5	MRR
Glove mean	0.1709	0.1951	0.1999	0.2080
Glove mean vclaim+title	0.1809	0.1985	0.2043	0.2125
Glove mean vclaim+title+subtitle	0.1859	0.2085	0.2146	0.2229
Glove dense	0.2563	0.2839	0.2912	0.3022
Glove dense vclaim+title	0.3015	0.3250	0.3320	0.3450
Glove dense vclaim+title+subtitle	0.2889	0.3287	0.3380	0.3395
LSTM	0.2814	0.3166	0.3254	0.3379
LSTM vclaim+title	0.2261	0.2638	0.2696	0.2871
LSTM vclaim+title+subtitle	0.2638	0.3019	0.3097	0.3125
biLSTM	0.3467	0.3886	0.3998	0.4078
biLSTM vclaim+title	0.3317	0.3744	0.3857	0.3994
biLSTM vclaim+title+subtitle	0.3316	0.3827	0.3908	0.4012
2biLSTM	0.2990	0.3346	0.3411	0.3463
2biLSTM vclaim+title	0.1709	0.2102	0.2157	0.2280
2biLSTM vclaim+title+subtitle	0.2312	0.2797	0.2903	0.3027
2biLSTM-mean	0.1809	0.2228	0.2278	0.2489
2biLSTM-mean vclaim+title	0.2211	0.2529	0.2602	0.2696
2biLSTM-mean vclaim+title+subtitle	0.2110	0.2529	0.2554	0.2709
2biGRU	0.2714	0.3099	0.3177	0.3343
2biGRU vclaim+title	0.2513	0.2814	0.2935	0.3072
2biGRU vclaim+title+subtitle	0.2462	0.2822	0.2880	0.3023
2biGRUbn	0.2814	0.3216	0.3337	0.3450
2biGRUbn vclaim+title	0.2864	0.3132	0.3245	0.3384
2biGRUbn vclaim+title+subtitle	0.3015	0.3283	0.3381	0.3522
TF-IDF	0.7010	0.7471	0.7544	0.7594
TF-IDF vclaim+title	0.7915	0.8233	0.8308	0.8312
TF-IDF vclaim+title+subtitle	0.8166	0.8434	0.8459	0.8480
SBERT-MNRL	0.9271	0.9485	0.9533	0.9425
SBERT-MNRL vclaim+title	0.9372	0.9577	0.9602	0.9500
SBERT-MNRL-GPT	0.9523	0.9686	0.9696	0.9603
SBERT-MNRL-GPT vclaim+title	0.9372	0.9594	0.9606	0.9508
SBERT-MNRL-GPT vclaim+title+subtitle	0.9422	0.9619	0.9631	0.9534
LambdaMART	**0.9623**	**0.9795**	**0.9817**	**0.9720**
LambdaMART vclaim+title	0.9422	0.9619	0.9644	0.9546
LambdaMART vclaim+title+subtitle	0.9472	0.9719	0.9742	0.9632

Table 3: Comparison of vclaim augmented models

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.