Team Buster.ai at CheckThat! 2020: Insights And Recommendations To Improve Fact-Checking

Mostafa Bouziane, Hugo Perrin,
Aurélien Cluzeau, Julien Mardas, and Amine Sadeq

Buster.AI, France
contact@buster.ai

Abstract. As part of the CheckThat! 2020 Task 2, we investigated sentence similarity using transformer models. In Task 2, the goal was to effectively rank claims based on their relevancy compared to an input tweet. While setting our baseline on sentence similarity for fact-checking, we gathered insights we felt compelled to share in this paper. We learned how multi-modal data utilization could foster significant uplifts in model performance. We also gained knowledge on which hybrid training and strong sampling worked best for fact-checking applications, and wanted to share our interpretation of the results we got. Finally, we want to explain our recommendations on data augmentations. All of the above allowed us to set our baseline in fact-checking in the CLEF Checkthat! 2020 Task 2 competition.

Keywords: Fact-Checking, Veracity, Evidence-based Verification, Fake News Detection, Computational Journalism, Natural Language Processing, Deep Learning, Language Model, Sentence Similarity.

Introduction

In the run-up to elections in many economic powers, being able to discern deception from real news has never been more critical. Moreover, in recent years, the actual cost of inaccurate news has been assessed more thoroughly. With occurrences like the Bloomberg dubious report almost wiping up 6 billion euros of Vinci market value, resulting in a 5 million euros fine for the media conglomerate recently [13] [4], it is also manifest that fake news can have a significant impact on economic and financial markets. As we are observing during the 2019/2020 coronavirus crisis [8], false health information, potentially causes grave harm to public safety and societies as a whole, from the economic to the political stability of entire countries. While most of the effort to combat them is handmade, we believe automation is vital to ensure propelling this fight to a broader impact.

In CheckThat! 2020 Task 2 [10], as descibed in [9], the intended goal was to effectively rank claims based on their relevancy to an input tweet. We had to output a list of numerical scores for sentences in the database, the higher the individual score, the more accurately related the match with the input text is. It became natural that we judged our results with the Mean Average Precision metric (MAP), evaluating if the valid claim is in the top-scored evidences. We will expand in the background section on the several approaches that have been tried. Figure 1 explains our strategy: after preprocessing the data and filtering out potentially unsuitable claims, we can finally estimate the matching probabilities with our neural network on acceptable claims. This filtering enables faster computations while preserving most of the performance that we have when computing scores for all documents of the database. We consequently trained our transformer model to learn sentence similarity on a semantic level, which we will detail in the methodology section.

Solving such a task will assist human fact-checkers in increasing their checking throughput drastically. A desirable algorithm for this task will permit obtaining source candidates automatically, diminishing proof searching times for human fact-checkers, as explained in Figure 1. It is noteworthy that sentence correspondence analysis only serves as a guide for humans, and cannot replace them for the ultimate purpose of fact-checking. Indeed, the necessity for human settlements can further ensure trust from everyone on the exactness of the fact-checking process, confidence, which is also strengthened by the associated facts from cited references so anyone can verify them.

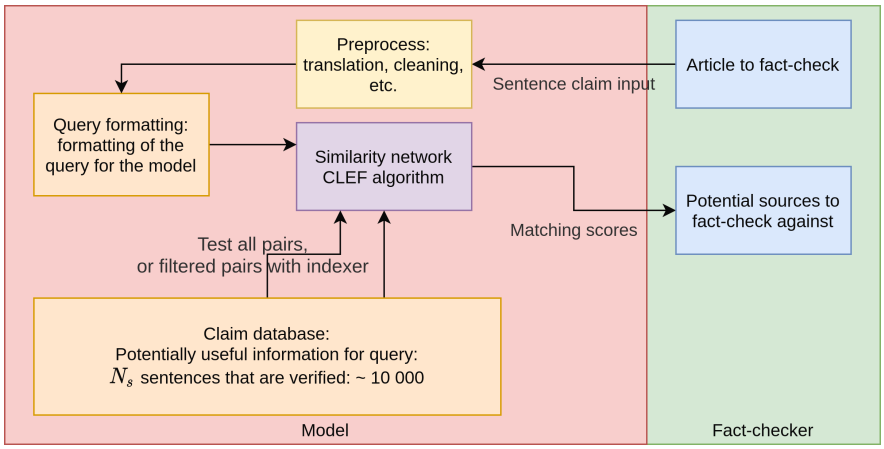

Fig. 1. Fact-checking pipeline

Furthermore, grand strides have been made for general interpretable artificial intelligence, and resolving such a task benefits in achieving this in the fact-checking setting. Being able to find matches on an evidence base allows the

algorithm to provide its sources for individuals to gauge its prediction quality and exhibits where an automatic entailment model built on top would acquire its reference material from.

For all those reasons, this aim has a practical significance and usefulness. After focusing on what has been done already to solve such a problem, we will succinctly present our competition-winning submission. We will outline the underlying architecture and how we trained it, which will be a strong starting point to describe the insights we gained from the competition. The first recommendation we share is how multi-modal knowledge can be leveraged to enhance score quality. Then we want to discuss the use of external datasets, which helped us propelling our algorithm farther and our interpretation of why specific datasets appeared to help more than others. Building on these observations, we want to highlight how proper sampling is required to train models efficiently. Finally, we will give a concise overview of other data augmentations that have impacted our model both for training and inference.

Background

In Task 2, we recall the goal is to rank claims based on their correspondence with the input statement. So, chiefly our goal is to build a sentence similarity model to learn semantics and hidden meanings. The latest advances in Natural Language Processing (NLP) suggest to use Transformer models and attention models. The Transformer [14] idea placed the latest paradigm for solving sequence-to-sequence tasks while handling long-range dependencies with ease. This architecture is based on self-attention, which allows the model to look at the other words in the input sequence to understand a particular word in the sequence better. Such attention is instrumental when the sentence is too long, or it contains a hidden context, this is why research has devised different variants of Transformer models, of which Bidirectional Encoder Representations from Transformers (BERT) is the most notable and constitutes state of the art [2].

BERT's introduction caused a massive surprise in the NLP community, as it outperformed previous models in so many distinct tasks. This considerable success comes from BERT being one of the first models that were non-directional. Previous models based on recurrent neural nets (RNNs) or convolutional neural nets (CNNs) implied a notion of direction, insofar as they implied a specific sequentiality when reading the input. For example, these models will predict a missing word in a sentence using either previous words or the following words, but can solely combine both approaches as an ensembling method. Nevertheless, with the BERT approach based on transformers, this prediction can be made using all available words simultaneously through the attention-based architecture, making BERT much more potent in creating context-based embeddings. These context-aware representations are of interest for the task at hand, since they allow elegant intertwined semantic links between query input and potential candidate evidence. In addition to a new non-directional approach, BERT-like models also introduced a new way of encoding input data, which makes BERT

able to receive a single sentence as input or a two-sentence input, making it suitable for such a large field of different tasks. Moreover, the pre-trained weights on vast amounts of data were released, leading to a sizable interest by the community and a substantial amount of analysis, understanding, and improvement of the architecture, creating a series of BERT based papers with several adjustments and specifications [6].

We also deem noteworthy to draw parallels between the CLEF Task 2 natural pipeline and that of some of the pipelines proposed in Open Domain Question Answering, like has been posted recently by HuggingFace labs [3], since both exhibit the same kind of filtering/classify pipeline. This blog post has not inspired our work, as it just was published at the time of writing, but it is nonetheless an interesting source to reframe our current work.

Methodology: Our submission overview

Task Definition and dataset

Task 2 aims to build tools that verify tweets based on a database of genuine claims. When we feed a tweet to the model, the comparison made with the claims database will produce a ranking from the most to the least relevant claims extracted from the database.

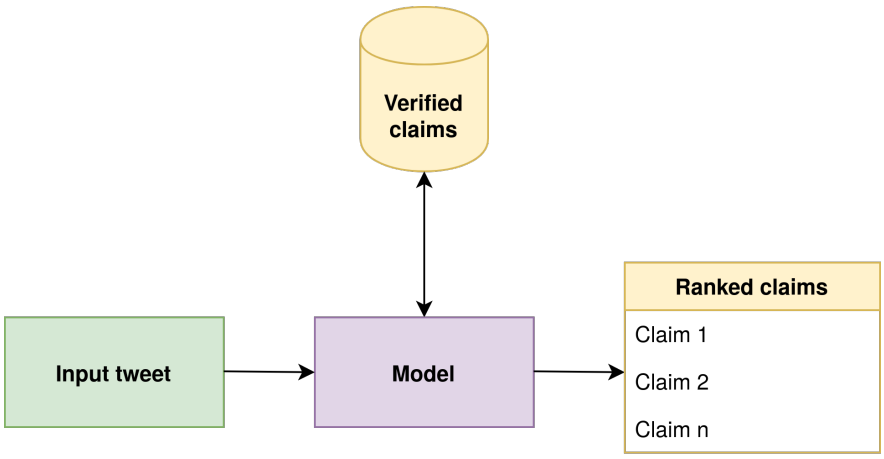

Fig. 2. Task 2 definition

Pairs of tweet and related claim constitute the samples of the available dataset; the tweet is the input that we want to verify, and the related claim can be retrieved from the database of verified claims.

Proposed approach

A rank-based method is an obvious thinking starting point to solve this task: a method, where the model will have to learn patterns to rank pairs based on how related they are. Nevertheless, for this specific task, we want to exploit the fact that a single claim can be related to the input, simplifying the ranking task to either being related, i.e., top-ranked, or not related, i.e., ranked second or more. A rank-based model can be the right approach if the rank scores are smoothed and well distributed over all claims, which is hard to achieve, since we do not have this information in the provided dataset.

Training: Therefore, we instead opted for a binary classification approach, as follows:

- Using genuine examples (pair tweet/claim) provided in the dataset.
- Generating wrong examples using several sampling strategies.
- Using a binary cross-entropy loss.

Inference: For all our submissions, once the model is trained properly, we pair each tweet in the test set with all the verified claims in the database, and we then apply the neural network with a composite score s function for ranking, that we defined as in Equation 1, with p_p being the probability of being related, and p_n being the probability of not being related to the query. It is important to mention that we replace the last sigmoid activation by a softmax activation for the last layer, even though in most cases sigmoid and softmax perform similarly, since the values sum up to roughly one already on the right data. Nevertheless, softmax will prove a more reliable activation on unknown data.

$$s\left(p_n, p_p\right) = \begin{cases} p_p & \text{if } p_p \geq p_n \\ 1 - p_n & \text{otherwise} \end{cases} \tag{1}$$

Network architecture

Our network relies upon on a RoBERTa checkpoint [7] to begin training. Therefore, the underlying architecture is a basic BERT architecture with a 12-layer, 768-hidden, 12-heads configuration, which amounts to 125 million parameters [17]. Those parameters result from the need to have both robust and fast inference, which it both achieved in practice, as it achieved the top score of around 93% MAP at depth 5, with an inference time below 0.1 seconds per pair. In absolute fairness, we shall point out that the complexity of inference, with m being the number of facts in our database, and n being the number of queries, is in $O(nm)$, which means our approach requires a pre-filtering to scale with a broader knowledge base.

Augmentations and dataset

We must now disclose the external data we tried to provide as a supplement for training. We first tried using SciFact [15], a scientific claim verification corpus, comprising 1 409 claims fact-checked against 5 183 abstracts with entailment labels. Secondly, we tried augmenting our corpus with FEVER [12], a quite substantially large dataset with 185 445 short claims, a few orders of magnitude larger than SciFact and CLEF datasets. Finally, we also used Liar [16], which constituted around 12 800 claims, placing it as a middle-ground in terms of size and featuring short claims and evidence.

Results: Our insights

For all comparisons here, we used our checkpoints obtained at early training to avoid overfitting issues to falsify any results, and we will mention whenever we use a model checkpoint for our submissions for reference. We used Whoosh [1] as a baseline instead of elastic search, and it provides a clean Python API, and obtained decent metrics compared to ElasticSearch in our experiments.

Multi-modal data utilization

We define multi-modal information in this context as any information coming from either external links, pictures, videos, or audio information contained in the textual data; information, which we eventually translated into text data that can be fed to the model alongside the original tweet, as described in Figure 3. The extraction process mentioned can be applied on many kinds of data; from image reverse search to get a title for pictures, to fancier image/video description networks, or even speech-to-text algorithms, the possibilities are varied. For our submission, we mainly converted links to the page name associated with the linked page, as well as getting a title for images referred to in tweets, with an image reverse search. Table 1 sums up the results we have in terms of inference, and we can notice a significant uplift in performance, since, with the same model checkpoint, and no retraining, we achieve around 3% better performance, which amounts to more than half of the potential performance left to gain in this case. The importance of such preprocess arises from the inherent multimedia essence of Twitter data, which can also be found in most news outlets nowadays. Hence, we preconize easing up the learning and inference process by adding those additional pieces of information.

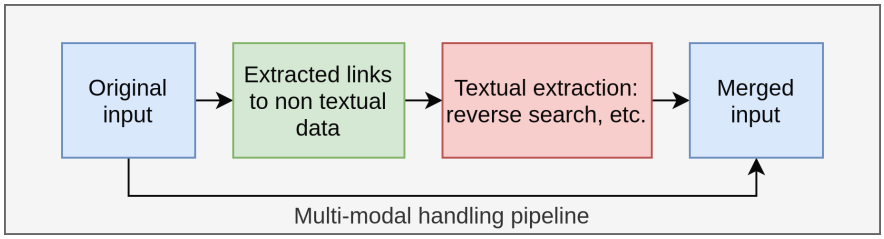

Fig. 3. Multi-modal pipeline

Metric	Depth	Whoosh baseline score	Without multi-modal	With multi-modal
MAP	1	0.804	0.96	0.99
MAP	3	0.838	0.967	0.992
MAP	5	0.843	0.97	0.992
MAP	10	0.846	0.97	0.992
MAP	20	0.847	0.97	0.992
MAP	all	0.847	0.97	0.992

Table 1. Metrics table on dev set. Results comparison between baseline, with and without multi-modal information.

Hybrid training

This section will explain how we used hybrid training with different external data and their effect on the overall results.

FEVER: Sampling pairs from the FEVER dataset during training seems to help the model improve its performance. One reason is that those pairs have similar syntactic structures and lengths to the ones available in Task 2. The performance was stable on the dev set; the results on the test set, where we had better MAP at depth k values, confirmed this.

SciFact: Sampling pairs from the SciFact dataset during training did not improve the model performance. One reason is that examples from Scifact have varying numbers of tokens, which are ampler compared to our core data, making it hard for the model to converge and generalize well.

Liar: Same as SciFact, the Liar dataset did not improve the performance of the model, since the structure of the examples does not match with the ones we have in Task 2.

Metric	Depth	without FEVER	with FEVER
MAP	1	0.8970	0.9070
MAP	3	0.9260	0.9370
MAP	5	0.9290	0.9380
MAP	10	0.9290	0.9380
MAP	20	0.9290	0.9380
MAP	all	0.9290	0.9380

Table 2. Metrics table on test set. Results comparison between baseline, with and without sampling from FEVER dataset.

Sampling scheme

We also advocate for sensible sampling schemes to ensure training on the optimal information signal. When scrutinizing the CLEF dataset, we observe that naive sampling will often sample what we call negative samples, i.e., samples not related to the query we pair it with. Those negative samples are utterly unrelated to the said query, leading to learning a more trivial solution of not pairing sentences which do not share anything meaningful. However, this naive approach leads to subpar MAP, as it will have a hard time distinguishing close evidence and rank them in an unsatisfiable manner as a consequence. We can illustrate this effect with the following example: with the query "The Pope is blue with yellow stripes." and most evidence talking about age and place of birth of various people, plus a few sentences about the Pope, a non-optimal sampling could lead to the algorithm only giving a high score to anything with "The Pope" contained in it, and therefore, poorly scoring the other sentences about the Pope potentially. We solved this, inspiring ourselves from a reinforcement learning

principle called experience replay [5], in which we use past predictions to learn more efficiently. In our case, we went towards this idea by sampling according to indexed searches, which produced relatively good results that could fool our algorithm trained with naive sampling, forcing it to learn the proper semantics to distinguish between syntactically and lexically close sentences. Ideally, we would take the top ranked results of our model to generate hard examples, however we chose to avoid the incurred computationnal cost, while using ElasticSearch results as a proxy for our model ranking. This sampling is a fast and accurate enough approximation of the reinforced mechanism of sampling new data based on our algorithm's error.

We used a balanced sampling, and we fully acknowledge that doing this will introduce a bias in the distribution of our dataset, which can have negative impact on the learning; nevertheless, we advocate that it is worth leveraging towards the fact-checking goal. In this context, we sample pairs of query and evidence, therefore balancing means we sample as much matching pairs than non-matchihng pairs. In Table 3, we have evidence it can be hugely beneficial to the training process, as it helps to gain a tremendous amount of performance, using the same network as our submission, making the difference between a usable and unusable solution.

Metric	Depth	With proposed sampling	With naive sampling
MAP	1	0.96	0.1
MAP	3	0.967	0.1
MAP	5	0.97	0.1
MAP	10	0.97	0.1
MAP	20	0.97	0.1
MAP	all	0.97	0.1

Table 3. Metrics table on dev set. Results comparison between with and without proper sampling.

Additional augmentations

In this section, we will present some other strategies that we tried but did not seem to impact the improvement of the model's performance significantly.

Named entity recognition (NER) based augmentation : Given that Task 2 data contains a lot of named entities, which are a crucial indicator of similarity between sentences, we tried to run a named entity recognition model on pairs and augment them with the detected labels. The model is a fine-tuned version of the bert-base multilingual cased model weights from HuggingFace's transformers [17]. The results did not improve the overall performance, maybe because our augmentation strategy was not robust since we only add a separator and the detected labels at the end of each sentence, which is not very efficient since many sentences will have common labels and this will prevent the model

from a better convergence. A better strategy to try would be to add the label next to the detected entity so that the model fits well on this kind of augmentation, and will make more sense of the relation between the words and the added labels. Another strategy would be to use the one-hot encoding of the named entities and stack them with BERT embeddings before the classifier layers; that way, the model will learn how to use those added encodings without having much negative impact on the semantics learned from the original sentence.

ConceptNet [11] based augmentation : ConceptNet, a general knowledge graph, treats words and phrases of natural language as nodes that it connects with labeled edges, therein representing the general knowledge involved in understanding language, facilitating comprehension of the meanings hidden behind everyday use of a word. The intuition behind using ConceptNet for our task is that combining it with words/sentence embeddings empowers an understanding that is not accessible solely from distributional semantics. As with NER, we augmented pairs using ConceptNet API, with semantic relations it produces. Whenever we detect a part of speech in a sentence, i.e., verbs, nouns, or adjectives, we append to sentences the relations retrieved from the graph (e.g., "chanting is synonym of shouting"). When adding those augmentations, the results did not vary significantly, since we gained around 0.5% in MAP at depth 1 on dev set, and almost stable for the other depths. Again, the strategy of adding a separator and the augmentation at the end of the sentence may not be the most pertinent. Indeed, ConceptNet supplies plenty of distinct relations such as "synonym of", "antonym of", "related to", "can be done with"..., so choosing the right relation is of great importance, and cannot be done straightforwardly, as the correct relation to use may depend on context.

Metric	Depth	without ConceptNet	with ConceptNet
MAP	1	0.96	0.965
MAP	3	0.967	0.967
MAP	5	0.967	0.968
MAP	10	0.967	0.969
MAP	20	0.967	0.970
MAP	all	0.967	0.970

Table 4. Metrics table on dev set. Results comparison between with and without using ConceptNet.

Back and forth translation : In order to augment the positive examples in the dataset, one can think of using back and forth translation strategy, which means : given a source S (English in our case) and target T (anything but English) language, we want to feed the example to translation algorithms following two steps:

- Step one: the example will be fed to a translation algorithm from S to T, that way we will get a new sentence in the target language.

– Step two: the new sentence will be fed to a translation algorithm from T to S, that way we will get a new sentence in the source language, semantically the same as the first one but with a different formulation.

Metric	Depth	without translation augmentation	with augmentation
MAP	1	0.96	0.95
MAP	3	0.967	0.96
MAP	5	0.967	0.965
MAP	10	0.967	0.965
MAP	20	0.967	0.965
MAP	all	0.967	0.965

Table 5. Metrics table on dev set. Results comparison between with and without using translation augmentations.

Doing so, the dataset will be more diverse in terms of syntax, and will help the model to generalize better. The model are MarianMT models from Hugging-Face's transformers [17].

This strategy did not help the baseline model to improve because the results were stable (Figure 5). We believe that this is due to the fact that the dataset is not big enough, so the augmentations did not have much effect on the overall performance.

Conclusion

In conclusion, our first recommendation for fact-checking was to ensure any piece of information, no matter the medium, is leveraged, to avoid as much as possible, relying on hidden information/context or implicit knowledge. Then, we wanted to stress the importance of consistent extra datasets, and how they helped training, even when dealing with unrelated subjects, as it was the case for us with FEVER, but also how they would impair learning if their size are not consistent. We also explained the importance of proper sampling. Last but not least, we presented a few augmentations that we experimented with, primarily semantic and lexical related augmentations, and hopefully gave inspiration for more augmentations to help fact-checking efforts in the future.

References

1. Chaput, M. Whoosh documentation, 2020.
2. Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805* (2018).
3. Jernite, Y. Explain anything like i'm five: A model for open domain long form question answering, 2020.

4. Kadri, F., and de Presse, A. F. Le faux vinci ou l'éloge de la prudence., 2016.
5. Lin, L.-J. Reinforcement learning for robots using neural networks. Tech. rep., Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.
6. Liu, Q., Kusner, M. J., and Blunsom, P. A survey on contextual embeddings. *ArXiv abs/2003.07278* (2020).
7. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR abs/1907.11692* (2019).
8. Nations, U. During this coronavirus pandemic, 'fake news' is putting lives at risk: Unesco, 2020.
9. Shaar, S., Babulkov, N., Da San Martino, G., and Nakov, P. That is a known lie: Detecting previously fact-checked claims. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), ACL '20, Association for Computational Linguistics, pp. 3607–3618.
10. Shaar, S., Nikolov, A., Babulkov, N., Alam, F., Barrón-Cedeño, A., Elsayed, T., Hasanain, M., Suwaileh, R., Haouari, F., Da San Martino, G., and Nakov, P. Overview of the CLEF-2020 CheckThat! lab on automatic identification and verification of claims in social media: English tasks. In *Working Notes of CLEF 2020—Conference and Labs of the Evaluation Forum* (Thessaloniki, Greece, 2020), CLEF '2020.
11. Speer, R., Chin, J., and Havasi, C. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR abs/1612.03975* (2016).
12. Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 809–819.
13. Times, B. French hoax costs bloomberg 5m euros in fines., 2019.
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *CoRR abs/1706.03762* (2017).
15. Wadden, D., Lin, S., Lo, K., Wang, L. L., van Zuylen, M., Cohan, A., and Hajishirzi, H. Fact or fiction: Verifying scientific claims, 2020.
16. Wang, W. Y. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Vancouver, Canada, July 2017), Association for Computational Linguistics, pp. 422–426.
17. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv abs/1910.03771* (2019).