bigIR at CLEF 2018: Detection and Verification of Check-Worthy Political Claims

Khaled Yasser, Mucahid Kutlu, and Tamer Elsayed

Qatar University, Doha, Qatar
{khaled.yasser, mucahidkutlu, telsayed}@qu.edu.qa

Abstract. With the enormous amount of misinformation spread over the Internet, manual fact-checking is no longer feasible to prevent its negative impact. There is an urgent need for automated systems that can make fact-checking process faster and effectively detect the veracity of claims. In this paper, we present our participation in the two tasks of CLEF-2018 CheckThat! Lab. To rank claims based on their check-worthiness (Task 1), we propose a *learning-to-rank* approach with features extracted by natural language processing such as named entity recognition and sentiment analysis. For veracity prediction (Task 2), we propose using an external Web search engine to retrieve potentially-relevant Web pages and extract features from relevant segments of those pages to predict the veracity. In the official evaluation, our best performing runs for Task 1 are ranked 4^{th} (out of 16 runs from 8 teams) and 1^{st} (out of 5 runs from 2 teams) over the English and Arabic datasets respectively, while our best performing run for Task 2 is ranked 6^{th} (out of 10 runs from 5 teams) over the English datasets.

Keywords: Fact-Checking · Check-Worthiness · Veracity Prediction.

1 Introduction

In the wake of the widespread of misinformation on the Internet and in the news, there emerged a need to combat this phenomenon as effectively as possible. However, despite the advent of technology, human fact-checkers could not keep up with the pace in which misinformation is perpetuated and spread. As a result, there is a need to develop automated systems to assist and help combat false claims and misinformation.

CLEF-2018 CheckThat! Lab [9] introduced two tasks which address two important aspects of fact-checking systems. The main goal of the first task [1] is to detect the check-worthy claims in political debates and prioritize them based on their check-worthiness. This resultant system has two benefits: (1) filtering the claims to be automatically fact-checked, and (2) helping human fact-checkers prioritize the claims to be fact-checked to focus on the most important ones. The second task [2] aims at predicting the veracity of the claims automatically, which is the ultimate goal of fact-checking systems. CheckThat! Lab released

both English and Arabic datasets for the two tasks. In this paper, we present our approaches to tackle these two important problems in the context of the lab.

For the *check-worthiness* task, we use a *learning-to-rank* approach with features extracted for each sentence in the debates. The features include word embeddings, types of named entities, part-of-speech tags, and sentiment and topic of sentences. We evaluated the impact of each feature group on training data and eventually submitted (for the test phase) runs on 3 models where each model has different sets of feature groups. Our best performing model that uses only features of named entities, sentiment and topic of sentences is ranked 4^{th} in the official evaluation. For the Arabic dataset, we automatically translated the dataset into English and used the same models. Our best performing model ranked 1^{st} in the official evaluation, but only 2 groups participated in the Arabic data challenge.

For the *factuality* task, we first retrieve some potentially-relevant Web pages using a commercial Web search engine, excluding the pages not allowed (by the lab organizers) for the task. Next, we detect the relevant segments within the pages and use them to extract features for each claim. Our features include percentages of sentences confirming the claim and contradicting with the claim, and also stances of relevant segments. Our best performing run is ranked 6^{th} in the official evaluation for the English dataset, but we did not participate in the Arabic data challenge in this task.

2 Task 1: Check-Worthiness

In Task 1 [1], the goal is to rank sentences made in presidential debates according to their likelihood of being check-worthy. In this section, we describe our approach and present the evaluation results.

2.1 Proposed Approach

Prioritizing the claims based on their check-worthiness is a ranking problem. Therefore, our approach relies on using learning-to-rank (L2R) methods. There are two versions of the datasets, one in English, and another in Arabic. First, we explain our approach for the English dataset, then how we modify our approach for the Arabic dataset.

Learning-to-Rank Model. We propose an L2R model for this task because the goal is to rank claims based on their check-worthiness instead of classifying them. We focus on pairwise and point-wise L2R models since we do not have a list of queries. We use RankLib L2R toolkit[1] in our implementation. In our experiments on the training data, we observed that the MART algorithm [4] outperforms other L2R algorithms we tried (i.e., RankBoost and RankNet). Therefore we opted for the MART algorithm in the testing phase.

[1] https://sourceforge.net/p/lemur/wiki/RankLib/

Features. The sentences in the dataset can be a full or half sentence. Because the data is extracted from debates, there exist sentences used in daily language, sentences without any verb because of being interrupted by another person, or sentences that are just a continuation of a previously interrupted sentence. While this kind of challenge suggests to consider sentences before and/or after each sentence in feature extraction, we chose to ignore these contextual data and consider only the individual sentences because the features extracted from the context would overlap with other statements, which could potentially introduce noise. The features we extract for each sentence can be divided into 5 categories: (1) vector representation of sentences, (2) named entities, (3) part-of-speech tags, (4) sentiment, and (5) topic features. We explain each of these features next.

- **Vector Representation of Sentence:** We use Word2Vec to represent sentences. Due to the small sample set in our training data, we use embedding-based vector representation instead of term-based representation. This allows us to capture similar sentences, and not just ones with the exact terms used. It also has lower dimensions as opposed to that of a term-based representation. We used a model pre-trained on Google News[2] where each vector has 300 components, and represented each sentence by the average vector of the vectors of its terms.
- **Types of Named Entities:** Check-worthy claims usually contain some named entities such as organizations, countries, and people. However, not all types of entities will be helpful for identifying check-worthy claims. For example, a claim which mentions the name of an international company is more likely to be check-worthy than a claim which mentions the name of a local music group, both of which are valid named entity types. We represent the types of named entities as an n-dimensional vector where each dimension corresponds to a type. The vector contains binary features, reflecting the existence or absence of a certain entity type in a sentence. In order to detect the named entity types, we use IBM Watson API for Natural Language Understanding[3] which yields a vector of 26 features. Some of the available entity tags are: *person*, *organization*, *country*, and *geographical entity*.
- **Part-of-Speech (POS) Tags:** There can be structural aspects of sentences that can help distinguish check-worthiness. For example, a sentence written in future tense is less likely to be check-worthy than a sentence written in past tense. Therefore, we use POS tags to capture the sentence structure of claims. We represent POS tags as a vector of binary features where each feature corresponds to the existence or absence of a certain tag. We use Stanford CoreNLP [7] to extract the POS tags. The POS feature vector consists of 36 features.
- **Sentiment:** We use sentiment of sentences (i.e., whether the sentence is presenting a positive, negative, or neutral attitude) as a feature. We extract sentiment labels using Stanford CoreNLP [7], but we collapse different

[2] https://code.google.com/archive/p/word2vec/
[3] https://www.ibm.com/watson/services/natural-language-understanding/

grades of positive and negative labels (e.g., *very positive* and *very negative*) to *positive* and *negative*, respectively.

- **Topic:** The topic of the sentence can indicate whether it is worth being checked or not. For example, a sentence about the use of nuclear weapons is more check-worthy than one about the release date of a movie. Therefore, we extracted topics of sentences using IBM Watson API in which a sentence could be classified into multiple topics where each topic could be fine-grained up to three levels. In our implementation, we consider only topics which have a confidence score of 0.5 or higher and used only the first two levels; after manual inspection of the topics available we found the third level to be too fine-grained for such a small dataset. The topics, just like types of named entities and POS tags, are represented as a vector of binary features but with a dimensionality of 348.

Feature Selection. After constructing the feature vectors for all sentences, we performed two-tier feature selection to choose the most discriminant features. The first tier is group-level selection, in which we evaluate the impact of each feature group previously mentioned in a leave-one-out fashion. For example, we test a model with and without topic features to see their impact on the results. The first tier was done manually by trying different possible combinations of feature groups (See Section 2.2). The second tier adopts a variance-threshold feature selection where we removed features which have variance less than a certain threshold across the samples. We used a threshold of 0 to eliminate features which have the same value across all samples.

Handling Class Imbalance. We observed that the training dataset given for Task 1 is highly imbalanced such that only 3% of the statements have been judged as check-worthy. This could potentially cause problems on the test data due to lack of enough samples for check-worthy claims. Therefore, we oversample the positive samples by duplication. In our experiments with the training data, we tried multiple oversampling values and observed that an oversample factor of 400% is the most ideal one for the training data.

Arabic. All the steps we described so far have been applied on the English dataset. One of the main challenge for Arabic dataset is that there is no suitable NLP tools that can be used to extract features. Therefore, we opted for using machine translation methods. Mohammad et al. [8] discussed the effect of machine translation on sentiment analysis and showed that models trained on a different language yield results comparable to those trained on the same language. Therefore, we first translate the Arabic dataset to English using Google Translate API [4] and then apply the same model trained on the English one.

[4] https://cloud.google.com/translate/docs/

2.2 Evaluation Results

In order to pick the models to use in the test phase, we performed k-fold cross validation for evaluating different models. Testing the models on data from the same debates they are trained on might not give proper insights on how they would perform on the unseen data; they could share similar topics, talk about the same entities, or have similar syntactic styles. Therefore, we set the folds such that each has the data from a separate debate.

In our unreported initial experiments, we observed that MART outperforms other L2R algorithms when the maximum number of trees and learning rate are set to 100 and 0.1 respectively. Subsequently, we evaluated the impact of each feature group using the MART algorithm while also changing the number of leaves parameter. The values for the number of leaves parameter we tried are from 5 to 12 inclusive. Table 2.2 shows the Average Precision (AP) score of the two best-performing models for each group of features on the English dataset.

Features	**Number of leaves**	**AP on test**
All	10	0.196
All	12	0.195
All - {entity types}	6	0.241
All - {entity types}	7	0.250
All - {topics}	5	0.247
All - {topics}	6	0.241
All - {POS tags}	5	**0.332**
All - {POS tags}	6	0.281
All - {sentiment}	5	0.257
All - {sentiment}	8	0.216
All - {Word2Vec}	5	**0.420**
All - {Word2Vec}	6	0.337
All - {POS tags, Word2Vec}	5	**0.399**
All - {POS tags, Word2Vec}	6	0.347

Table 1. Results of training MART models with different groups of features and number of leaves for Task 1. The maximum number of trees in the training stage is set to 100. The ones in bold represent the models selected for the test phase.

As shown in Table 2.2, the best performing model is the one trained without using Word2Vec vectors as features using trees with only 5 leaves. Aside from Word2Vec, models trained without POS features also outperform the models that do not use a particular set of features. We select the top 3 models (shown in bold in the table) for the submission of both English and Arabic datasets without picking two models with the same feature groups.

In the official evaluation for the English dataset, our selected models *All - {Word2Vec}*, *All - {POS tags, Word2Vec}*, and *All - {POS tags}* achieved

MAP values of 0.1120 (10^{th} in the ranking), 0.1319 (4^{th} in the ranking), and 0.1117 (11^{th} in the ranking) respectively. As on the Arabic dataset, our models achieved 0.0899 (4^{th} in the ranking), 0.1498 (1^{st} in the ranking), and 0.0962 (3^{rd} in the ranking) MAP scores respectively. That clearly shows the second model (i.e., *All* - {*POS tags, Word2Vec*}) outperforms the other two models in both datasets. Interestingly, it performed better on the Arabic dataset than it did on the English one (0.1498 vs. 0.1319).

3 Task 2: Factuality

After finding which claims are more important to be checked, Task 2 [2] focuses on the second stage of automated fact-checking which is detecting the veracity of check-worthy claims.

3.1 Proposed Approach

We approach this task as a classification problem. For a given claim, we first find potentially-relevant Web pages using an external Web search engine. Then we detect the relevant segments in each of those pages and extract features from these segments. Finally, we predict the veracity of the claim using a learning model based on the extracted features. We discuss our approach in detail next.

Web Search. The first step of our approach is retrieving search results for the claim. We follow the approach described by Karadzhov et al. [5] to generate a query from a given claim. We use Google Custom Search[5] to retrieve the results but with custom settings to filter out websites which are not allowed for the task. We retrieve 10 results for each claim.

Relevant Segments Detection. The goal of this step is to find which parts of a Web page are relevant to the claim of interest. We compute the cosine similarity between the Word2Vec vectors of the claim and each sentence in the page. We consider a sentence as relevant if the cosine similarity score is higher than 0.5. Next, for each relevant sentence, we add a sentence before and a sentence after in order to capture the contextual information. We call these three-sentence structures *relevant segments*.

Features. We extract features from each relevant segment. We first explain the features and how they are extracted, then we explain how we aggregate the features from different pages. The features are as follows:

- **Stance Detection:** Stance detection is the process of finding whether a piece of text is for, against, or unrelated to another piece of text. Following

[5] https://cse.google.com/cse/

the Fake News Challenge[6], where the challenge was to check if an article supports or denies a claim using stance detection, we incorporate stance detection into our method. We use the implementation provided by [10] and train it on the entire Emergent dataset [3]. We first classify each relevant segment in all pages separately and then calculate the percentage of each label, yielding a feature vector of size 3.

– **Contradiction in Predicates:** A stance can be a subjective statement regarding a claim. For instance, let the claim be "He bought weapons" and a sentence extracted from a document be "I do not want to believe that he bought weapons." While the sentence is against the claim, it does not state whether the claim is actually true or not. On the other hand, consider the following sentence: "He forgot to buy weapons". The predicate of this claim (*"buy"*) contradicts the one in this sentence. In this feature, we try to detect if the predicate of a relevant segment contradicts with the predicate of the given claim. In order to extract such relation, we use TruthTeller[6] in combination with WordNet[7].
  In particular, we first generate a relation matrix, R, which represents the relation between two sentences. $R_{i,j}$ indicates the relation between term i in the first sentence and term j in the other, where the relation can be either *synonyms*, *antonyms*, or *unrelated*. We compare the lemmas of the terms and assign the label *synonyms* if the two terms are the same. Furthermore, we classify each term in a sentence using TruthTeller where a term can either be *positive* (i.e., the term is affirmed), *negative* (i.e., the term is negated in some way), *uncertain* (i.e., there is a doubt around the term), or *unknown* (i.e., a decision could not be made). We detect the relative truth of term j in sentence B based on a term i in sentence A, given that they have a relation, using the following rules:
  - If $R_{i,j} = synonyms$ and $T(A_i) = T(B_j) \rightarrow confirms$
  - If $R_{i,j} = synonyms$ and $T(A_i) \neq T(B_j) \rightarrow contradicts$
  - If $R_{i,j} = antonyms$ and $T(A_i) = T(B_j) \rightarrow contradicts$
  - If $R_{i,j} = antonyms$ and $T(A_i) \neq T(B_j) \rightarrow confirms$

  Figure 1 and 2 show an example of a relation matrix and truth predicates for the sentences *"He bought weapons"* and *"He forgot to buy weapons"* respectively. The terms "buy" and "bought" share the same lemma; their relation is set to *synonyms*. The terms "buy" and "bought" are synonyms with opposite truth values, *positive* and *negative*, respectively. Following the aforementioned rules, the two sentences contradict one another.
  The feature vector for predicates consists of two features, one for the percentage of confirming predicates, and one for the percentage of contradicting ones. We do exclude uncertain and unknown truth values to simplify the process since they do not offer any value.

The results of both stages, stance detection and contradiction in predicates, from all pages are aggregated into a single vector of features. This final vec-

[6] http://www.fakenewschallenge.org/
[7] https://wordnet.princeton.edu/

$$\mathrm{R(A,\ B)} = \begin{bmatrix} \textbf{synonyms} & \textit{unrelated} & \textit{unrelated} \\ \textit{unrelated} & \textit{unrelated} & \textit{unrelated} \\ \textit{unrelated} & \textit{unrelated} & \textit{unrelated} \\ \textit{unrelated} & \textbf{synonyms} & \textit{unrelated} \\ \textit{unrelated} & \textit{unrelated} & \textbf{synonyms} \end{bmatrix}$$

Fig. 1. The relation matrix R for the sentences A = *"He bought weapons"* and B = *"He forgot to buy weapons"*. Each cell $R_{i,j}$ represents the relation between term i in A and term j in B.

$$T(A) = [\textit{unknown}, \textbf{positive}, \textit{unknown}]$$
$$T(B) = [\textit{unknown}, \textbf{positive}, \textit{unknown}, \textbf{negative}, \textit{unknown}]$$

Fig. 2. The truth vectors for the sentences A = *"He bought weapons"* and B = *"He forgot to buy weapons"*

.

tor carries the following 5 features: the percentage of segments supporting the claim, the percentage of segments which are against the claim, the percentage of segments unrelated to the claim, the percentage of sentences with contradicting predicates, and the percentage of sentences with confirming predicates.

3.2 Evaluation Results

The evaluation process for Task 2 is similar to that of Task 1. We performed k-fold cross-validation where each fold corresponds to the claims from one debate. The optimization and evaluation for Task 2 was done on model prediction accuracy. We tried three common classifiers for this problem and performed grid search optimization for each classifier on all of its parameters. The results are reported in Table 2.

Features	Classifier	Accuracy
Stance & Contradiction in Predicates	SVM	0.523
	Random Forest	0.476
	Logistic Regression	**0.602**
Stance	SVM	0.533
	Random Forest	**0.571**
	Logistic Regression	0.476
Contradiction in Predicates	**SVM**	**0.619**
	Random Forest	0.427
	Logistic Regression	0.428

Table 2. Results of training multiple models with stance and contradiction in predicates features for Task 2.

Although stance is more common, the results in Table 2 show that using contradiction in predicate, by itself or in addition to stance, has the potential to improve classification accuracy.

The selected models, highlighted in bold in Table 2, are SVM with contradiction-in-predicates features, logistic regression with all features, and random forest with stance features. The metric in the official evaluation was Mean Squared Error (MSE), and our models received scores of 0.9640, 0.9640, and 0.9425, respectively. The first two models landed the last place, while the last one landed the 6^{th} place (out of 10).

4 Conclusion

In this paper, we present our methods for the tasks of CLEF-2018 CheckThat! Lab. For the task of detecting check-worthy claims, we proposed a learning-to-rank based approach which uses natural language processing methods for extracting the features. Our best performing model achieved the 4^{th} place on the English dataset. The same model, when used in conjunction with machine translation, got the 1^{st} place in the evaluation on the Arabic datasets.

Regarding the task of verifying the check-worthy claims, we proposed a system which uses an external Web search engine to collect evidence and then predicts the veracity of the claims by detecting the stance of relevant statements and contradicting or confirming sentences in the retrieved pages. Our best performing model got 6^{th} place. However, we observed promising results for the contradiction in predicates feature, suggesting that it can be a useful feature for fact-checking with more enhanced methods. However, it needs further experiments on larger datasets.

Acknowledgments

This work was made possible by NPRP grant# NPRP 7-1313-1-245 and NPRP grant# 7-1330-2-483 from the Qatar National Research Fund (a member of Qatar Foundation). Statements made herein are solely the responsibility of the authors.

References

1. Atanasova, P., Màrquez, L., Barrón-Cedeño, A., Elsayed, T., Suwaileh, R., Zaghouani, W., Kyuchukov, S., Da San Martino, G., Nakov, P.: Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims. Task 1: Check-worthiness
2. Barrón-Cedeño, A., Elsayed, T., Suwaileh, R., Màrquez, L., Atanasova, P., Zaghouani, W., Kyuchukov, S., Da San Martino, G., Nakov, P.: Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims. Task 2: Factuality

3. Ferreira, W., Vlachos, A.: Emergent: a novel data-set for stance classification. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies. pp. 1163–1168 (2016)
4. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
5. Karadzhov, G., Nakov, P., Marquez, L., Barron-Cedeno, A., Koychev, I.: Fully automated fact checking using external sources. arXiv preprint arXiv:1710.00341 (2017)
6. Lotan, A., Stern, A., Dagan, I.: Truthteller: Annotating predicate truth. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 752–757 (2013)
7. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014), http://www.aclweb.org/anthology/P/P14/P14-5010
8. Mohammad, S.M., Salameh, M., Kiritchenko, S.: How translation alters sentiment. Journal of Artificial Intelligence Research **55**, 95–130 (2016)
9. Nakov, P., Barrón-Cedeño, A., Elsayed, T., Suwaileh, R., Màrquez, L., Zaghouani, W., Atanasova, P., Kyuchukov, S., Da San Martino, G.: Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims. In: Bellot, P., Trabelsi, C., Mothe, J., Murtagh, F., Nie, J., Soulier, L., Sanjuan, E., Cappellato, L., Ferro, N. (eds.) Proceedings of the Ninth International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Lecture Notes in Computer Science, Springer, Avignon, France (September 2018)
10. Riedel, B., Augenstein, I., Spithourakis, G.P., Riedel, S.: A simple but tough-to-beat baseline for the fake news challenge stance detection task. arXiv preprint arXiv:1707.03264 (2017)