bigIR at CLEF 2019: Automatic Verification of Arabic Claims over the Web

Fatima Haouari, Zien Sheikh Ali, and Tamer Elsayed

Qatar University, Doha, Qatar
{200159617,zs1407404,telsayed}@qu.edu.qa

Abstract. With the proliferation of fake news and its prevalent impact on democracy, journalism, and public opinions, manual fact-checkers become unscalable to the volume and speed of fake news propagation. Automatic fact-checkers are therefore needed to prevent the negative impact of fake news in a fast and effective way. In this paper, we present our participation in Task 2 of CLEF-2019 CheckThat! Lab, which addresses the problem of finding evidence over the Web for verifying Arabic claims. We participated in all of the four subtasks and adopted a machine learning approach in each with different set of features that are extracted from both the claim and the corresponding retrieved Web search result pages. Our models, trained solely over the provided training data, for the different subtasks exhibited *relatively-good* performance. Our official results, on the testing data, show that our best performing runs achieved the *best* overall performance in subtasks A and B among 7 and 8 participating runs respectively. As for subtasks C and D, our best performing runs achieved the *median* overall performance among 6 and 9 participating runs respectively.

Keywords: Fact Checking · Arabic Retrieval · Learning to Rank · Web Classification.

1 Introduction

Fake news is witnessing an explosion recently, and it is considered as one of the biggest threats to democracy, journalism, and public trust in governments. In combating fake news, the number of manual fact-checking organizations increased by 239% in a period of four years, where it reached 149 fact-checkers in 2018 as apposed to only 44 in 2014[1].

One of the main challenges is that manual fact-checking does not scale with the volume of daily fake news. This mismatch can be attributed to the gap

[1] https://reporterslab.org/fact-checking-triples-over-four-years/

between the time the claim is made and the time the claim is checked and published, as it is very time-consuming for journalists to find check-worthy claims and verify them. Another challenge is that fact-checking requires advanced writing skills in order to convince the readers whether the claim is true or false [6]. In fact, it is estimated that check-worthiness of a claim and writing an article about it can take up to one day [7]. Moreover, manual fact-checkers are outdated [6]. Most of the fact-checking frameworks adopt the old content management systems specialized for traditional blogs and newspapers, but not built for the current modern journalism. A new approach is therefore needed for automated fake news detection and verification.

The industry and academia have shown an overwhelming interest in fake news to address the challenges of its detection and verification. Many pioneering ideas were proposed to address many aspects of fact-checking systems with their focus varies between detecting check-worthy claims [8, 10, 7], checking claims factuality [11, 15, 16, 20], checking news media factuality [2], and proposing full automatic fact-checking systems [9, 14, 22]. There are also some shared tasks proposed and open to the research community interested in the problem such as FEVER-2018 task for fact extraction and verification [18] and CheckThat! 2018 lab on automatic identification and verification in political debates at CLEF [13].

This year, CLEF-2019 CheckThat! Lab [4] introduced two tasks to tackle two main problems of automated fact-checking systems. The main objective of the first is to detect check-worthy-claims to be prioritized for fact-checking [1], while the second focuses on evidence extraction to support fact-checking a claim [5]. In this paper, we present the approach adopted by our bigIR group at Qatar University to address the second task.

Task 2 (Evidence and Factuality) addresses the problem of *finding evidence* over the *Web* for verifying *Arabic* claims. It assumes the system is given an Arabic claim (as a short sentence) and a corresponding ranked list of Web pages that were retrieved by a Web search engine for that claim. The system then needs to address four sub-problems, each is defined as a subtask as follows:

1. **Subtask A**: *Rank* the retrieved pages based on how *useful* they are for verifying the claim.
2. **Subtask B**: *Classify* the Web pages as "very useful" for verification, "useful", "not useful", or "not relevant".
3. **Subtask C**: Within each useful page, *identify* which *passages* are useful for claim verification.
4. **Subtask D**: Determine the true *factuality* of the claim, i.e., whether it is "True" or "False".

We have participated in all of the four subtasks. Since it is the first year of the task (and thus our first attempt), we generally adopted a *simple* machine learning approach, where learning models were trained only on the given training data over hand-crafted features. We applied feature ablation to assess the impact of each feature on the performance of our models.

For subtask A, to re-rank the pages based on their usefulness, we adopted a *pairwise learning-to-rank* approach with features extracted either from the

page as a whole (such as source popularity, URL links, and number of quotes), from the *relevant* segments in the page (such as the similarity score of the most relevant sentence), or from the search results (such as the original rank of the page). Additionally, we extracted claim-dependent features such as the similarity between the claim and the title and the snippet of the page.

For subtask B, we adopted a multi-class classification approach to classify the Web pages. We considered several features including word embeddings, named entities, similarity scores, number of relevant sentences in the page, and URL-based features (such as URL length, URL scheme, and URL domain).

For subtask C, we adopted a binary classification approach to classify the passages within a useful page. Features included Bag-Of-Words (BOW), named entities, number of quotes, score of most relevant sentence from each passage, and the similarity score between the claim and the passage.

For subtask D, we also adopted a binary classification approach to discover the claim's factuality given the retrieved Web pages. To classify the claim, we first identify the most similar pages to the claim for feature extraction. For the selected pages, we consider their similarity scores, source popularity, and the sentiment of the page.

Our contribution in this work is two-fold:

1. We participated in *all* of the four subtasks adopting a machine learning approach with relatively-different set of features in each. The features are extracted from both the claims and the retrieved Web pages.
2. Our best performing runs exhibited the best performance in both subtasks A and B among the submitted runs.

The remainder of this paper is organized as follows. Section 2 describes how we processed and extracted features from the claims and retrieved pages. Sections 3, 4, 5, and 6 outline our approach and discuss our experimental evaluation in detail for subtask A, B, C, and D respectively. Finally, Section 7 concludes and discusses possible future work.

2 Preprocessing & Feature Extraction

In our work, we apply common main preprocessing for all subtasks to parse documents, identify relevant segments, and extract features. However, we include or exclude some features in each subtask. In this section, we describe in detail the preprocessing steps and introduce and motivate the features we extracted at all levels. For each page, we extract two types of features: features that depend on the claim/page relationship (**claim-dependent**) and features that depend solely on the page (**page-dependent**).

In what follows, a text *segment* in a page is centered by one sentence, but also includes both the sentence that precedes and the sentence that follows it, as defined by Yasser et al. [21], to consider the context of the sentence.

2.1 HTML Parsing

As the Web pages are in raw HTML format, we parse each page by extracting only the clean version of the textual body discarding images, videos, and scripts using newspaper[2] and BeautifulSoup[3] Python libraries. We removed stopwords using Python NLTK[4] Arabic stopwords. We also discard the sentences containing less than 3 words, motivated by the empirical study done by Zhi et al. [22].

2.2 Text Vector Representations

In extracting our features, we consider two text vector representations:

- **Bag-of-Words (BOW)**: We consider BOW representation to represent full passages (mainly for subtask C). We considered only the terms that appeared at least 7 times in the training data, based on some preliminary experiments.
- **Distributed Representation (W2V)**: We consider word2vec embeddings [12] to represent the claim and the segments of a page; each is represented as the average vector of the embeddings of terms in the claim/segment. We used the pre-trained AraVec embeddings model proposed by Soliman et al. [17].

2.3 Relevant Segments Identification

To identify *relevant segments* in a page for a given claim, we represent the claim and each sentence in the page by their average of term W2V vectors. We then compute the cosine similarity between the vectors of the claim and each segment. Segments are considered relevant if the similarity score is higher than a threshold.

2.4 Page-Dependent Features

We extracted two types of page-dependent features: credibility and content.

Credibility Features To indicate the credibility of the page, we consider the following features:

- **Source Popularity (SrcPop):** This feature may indicate trustworthiness, as it captures how popular a particular website is. We used Amazon Alexa rank[5] motivated by Baly et al. [2] that used this feature to estimate the reliability of media sources. We consider this feature as a categorical feature by binning the ranking values into 10 categories, then we convert it to a one hot encoding vector of 10 binary features.

[2] https://pypi.org/project/newspaper3k/
[3] https://pypi.org/project/bs4/
[4] https://pypi.org/project/nltk/
[5] https://www.alexa.com/

- **URL Features**: these features were used by Baly et al. [2] to detect the reliability of web sources. We used Python URL handling library *urlib*[6] to parse the URL and extract the following orthographic features:
  - **Length (URLLen) and Number of Sections (URLSecs)**: The length of the URL path and the number of sections separated by '/' help indicate whether the website is legitimate, irregular, or a phishing website.
  - **Scheme (URLScheme)**: The URL protocol (https or http) indicates the trustworthiness of the website. We extracted the URL scheme then we used scikit-learn label encoder[7] to encode string values of schemes to integers.
  - **Domain Suffix(URLSfx)**: The suffix of a URL domain determines the source and credibility of the website. For example, a website with domain suffix *.gov* is a federal government site and is more credible than a commercial website with a suffix of *.com*. We used label encoder to encode their string values into integers.

Content Features From page body, we extract the following linguistic and similarity features:

- **Number of Quotes (NQts)**: For each page, we count the number of quotes in all relevant segments. This feature may be very useful to rank web pages and decide how useful they are for claim verification as it may indicate the credibility of the page by quoting sources. In our work, we considered only quotes with five words or more.
- **Number of URL links (NLinks)**: This feature represents the number of URL links in the retrieved page. It may indicate the credibility of the source by giving *references*.
- **Named Entities (NEs)**: Pages mentioning named entities may indicate the truthfulness of the page. We used Python polyglot NLP tool[8] to recognize location, organizations, and persons entities in the most relevant segment of the page. We form a vector of 3 integer values representing the number of occurrences of every entity type in the segment.

2.5 Claim-Dependent Features

We extracted the following features based on the claim-page interaction:

- **Original Rank (Rank)**: This feature is available from the search results and it represents how the page is *potentially-relevant* to the claim according to the search engine.
- **Similarity**: This includes cosine similarity between claim and title (**ClmTtlSim**), claim and snippet(**ClmSnptSim**), and claim and a passage (**ClmPsgSim**).

[6] https://pypi.org/project/urllib3/
[7] scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[8] https://github.com/aboSamoor/polyglot

- **Number of Relevant Sentences (NRelSent)**: For every page, we compute the similarity between the claim and each sentence. We count the number of relevant sentences in each page as it might indicate the relevance of the page.
- **Number of relevant webpages (NRelPages)**: For every claim, we count the number of webpages with a similarity score between claim and most relevant sentence higher than a certain threshold.
- **Score of the most Relevant Segment (MostRelSeg)**: This feature indicates how similar the most relevant segment is to the claim.
- **Sentiment (SntCnt)**: Sentiment analysis can help identify if the stance of the page is positive, negative, or neutral. This may help in identifying whether the page agrees with the claim or not. We use polyGlots Sentiment model[9] to extract sentiments. From the most relevant segment, we get two values, the number of words with positive polarity and the number of words with negative polarity.

3 Subtask A: Reranking Retrieved Pages

In this subtask [3], the goal is to rerank the retrieved pages based on their usefulness for verifying a specific claim. In this section, we present our proposed approach, experimental setup and results, our selected runs for CLEF submissions, and finally we will present the CLEF results.

3.1 Approach

Our approach is based on learning-to-rank (L2R). We propose a pairwise L2R model considering three different L2R classifiers, namely, SVM C-Support Vector Classification (SVC), which is implemented based on libsvm[10], Gaussian Naïve Bayes (Gaussian NB), and the ensemble classifier Random Forest (RF), using Scikit-learn Python library.[11] We consider the following features (discussed in Section 2):

- Basic features: Rank, SrcPop, and MostRelSeg.
- Similarity features: ClmTtlSim and ClmSnptSim.
- NLinks.
- NQts.

3.2 Experimental Setup

Parameters We experimented with the three different classifiers mentioned in 3.1. We set the kernel for SVC to linear, and set the number of estimators for the RF models to 100 (based on preliminary experiments). For the NB models, we did not tune any hyper-parameters and used the default settings.

[9] https://polyglot.readthedocs.io/en/latest/Sentiment.html
[10] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[11] https://scikit-learn.org/stable/index.html

Baselines We compare our models against a baseline that returns the pages ranked in their original ranks (i.e., based on relevance scores of the search engine, not on usefulness for fact-checking).

3.3 Evaluation on Training

As we were constrained by the size of the training data, containing only 10 claims, we adopted leave-one-claim-out (LOO) cross validation to evaluate the trained models. We optimized our models using the graded relevance measure NDCG@20.

We first experimented with different values of the cosine similarity threshold (0.4, 0.5, 0.6, and 0.7) when extracting relevant segments. In our unreported preliminary experiments, we observed that the best performing models were the ones trained with features extracted using a similarity threshold of 0.4 and 0.7, presented in Fig. 1 and Fig. 2 respectively. We also tried different combinations of features as shown in both figures.

The results show that our models could not beat the baseline with only the basic features. However, NB models outperformed the baseline when other features were introduced. We also notice that introducing the ClmTtlSim and ClmSnptSim to the basic features improved the performance of our models, while excluding the SrcPop feature improved the performance. Moreover, our proposed NLinks and NQts features did not have a noticeable impact on the performance of the models.

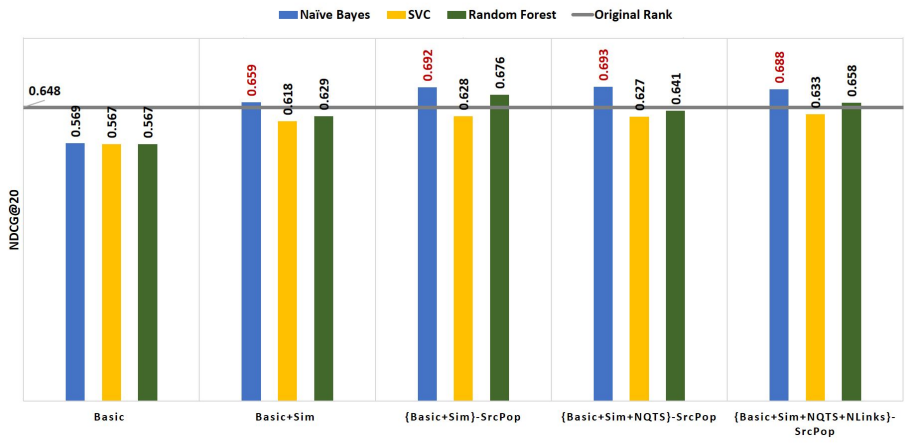

Fig. 1. Subtask A: Performance of L2R models on training data with combinations of features (cosine similarity threshold set to 0.4).

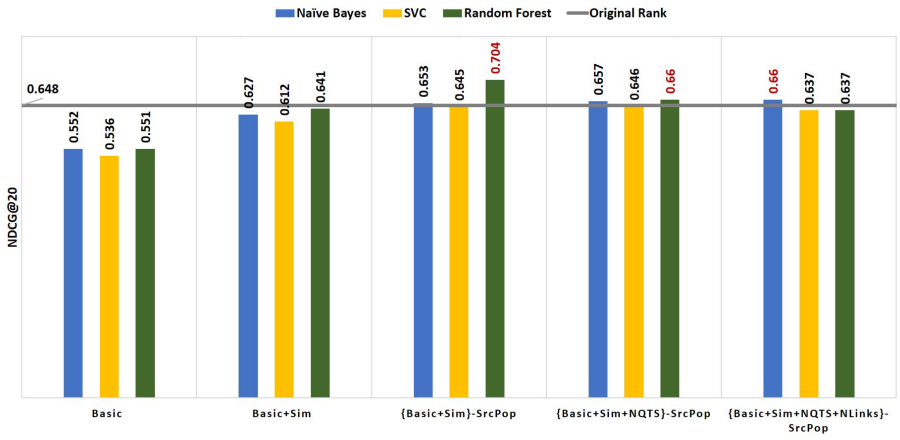

Fig. 2. Subtask A: Performance of L2R models on training data with combinations of features (cosine similarity threshold set to 0.7).

3.4 CLEF Evaluation

Runs As shown in Fig. 1, NB models outperform other L2R models over the training data, therefore we picked the 3 best NB models to submit to CLEF:.

1. NB trained with Basic, ClmTtlSim, and ClmSnptSim features, and excluding SrcPop.
2. NB trained with Basic, ClmTtlSim, ClmSnptSim, and NQts features, and excluding SrcPop.
3. NB trained with Basic, ClmTtlSim, ClmSnptSim, NQts, and NLinks features, and excluding SrcPop.

Moreover, when the cosine similarity threshold was set to 0.7, RF outperformed other models, as shown in Fig. 2, so we also picked its best performing model:

4. RF trained with Basic, ClmTtlSim, and ClmSnptSim features, and excluding SrcPop.

Results As shown in Table 1, the official CLEF evaluation shows that our best performing model on the test data was the NB model trained with basic, ClmTtlSim, and ClmSnptSim features (excluding SrcPop) which achieved NDCG@20 value of 0.55. This was the maximum score achieved among 7 runs submitted for this subtask. We observed that the performance of our models on training data was better than on testing data; this can be attributed to the small size of the training dataset, containing only 395 pages from 10 claims, which could be insufficient and not a good representative to train the models.

Table 1. Subtask A: Performance of CLEF submitted runs.

Features	Classifier	NDCG@20 on train	NDCG@20 on test
{Basic+Sim} -SrcPop	RF	**0.704**	0.47
{Basic+Sim+NQts} -SrcPop	NB	0.693	0.52
{Basic+Sim} -SrcPop	NB	0.692	**0.55**
{Basic+Sim +NQts+NLinks} -SrcPop	NB	0.688	0.51

4 Subtask B: Classifying Retrieved Pages

The main goal of this subtask [3] is to classify all retrieved Web pages based on how useful they are in detecting the claim's veracity. A webpage is useful if it has enough evidence to verify the claim and if its source is trustworthy. In this section, we present our approach, experimental setup, training results, and CLEF results for our submitted runs.

4.1 Approach

In our approach for this subtask, we use different machine learning algorithms to perform multi-class classification. We consider SVC as it shows to learn well from small datasets. We also include Gradient Boosting (GB) and RF as an ensemble model. As mentioned in 3.1, we use Scikit-learn Python library for our implementation. We consider the following features:

- Basic features: Rank, SrcPop and MostRelSeg.
- NEs in the relevant segment.
- NQts.
- URL features.
- W2V representation of both the claim and the relevant segment.

4.2 Experimental Setup

Parameters For SVC, we used an RBF kernel with regularization parameter C = 15 and L2 penalty, and we set γ to 0.01 to avoid over-fitting. For GB and RF models, we set the number of estimators to 100 and 150 respectively (based on preliminary experiments).

Baselines As a baseline we adopted Wang et al. [19] method for feature extraction and classification. Their dataset consists of short passages where passages are classified into five different categories. This baseline was selected because

the feature extraction methods are implemented on short passages similar to the size of our extracted relevant segments. Moreover, they are working on fine-grain classification.

Since our training data is highly imbalanced, we also used the Zero Rule algorithm as a baseline for this subtask. Zero Rule algorithm predicts the majority class in the dataset. In our training data, class -1 (non-relevant) is the majority class with 65% of the labels.

4.3 Evaluation on Training

We conducted multiple experiments in attempt to find which features combination will result in the best F_1 score. We split our dataset into 70% for training and 30% for testing. From our experiments, we noticed that varying the similarity threshold when extracting relevant segments had a significant impact on the overall score. We concluded that our best performing models were the ones trained with features extracted with similarity thresholds of 0.4 and 0.7. Fig. 3 and Fig. 4 show the results obtained from our experiments using similarity thresholds 0.4 and 0.7 respectively.

We observed that when training the classifiers with basic features and NEs the performance improved. On the other hand, incorporating some content features like URL features and W2V vectors had a negative impact on the performance of the classifiers. We also note that ensemble classifiers (GB and RF) outperformed the baselines and other classifiers all the time.

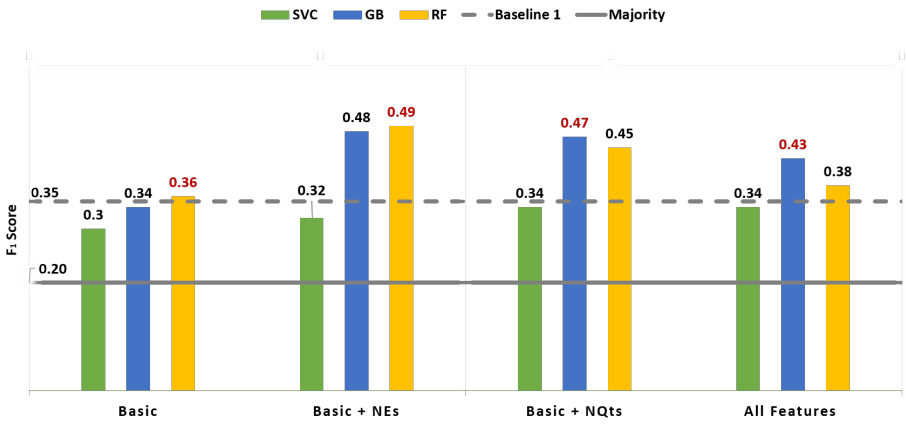

Fig. 3. Subtask B: Performance of classifiers on training data with combinations of features (cosine similarity threshold set to 0.4).

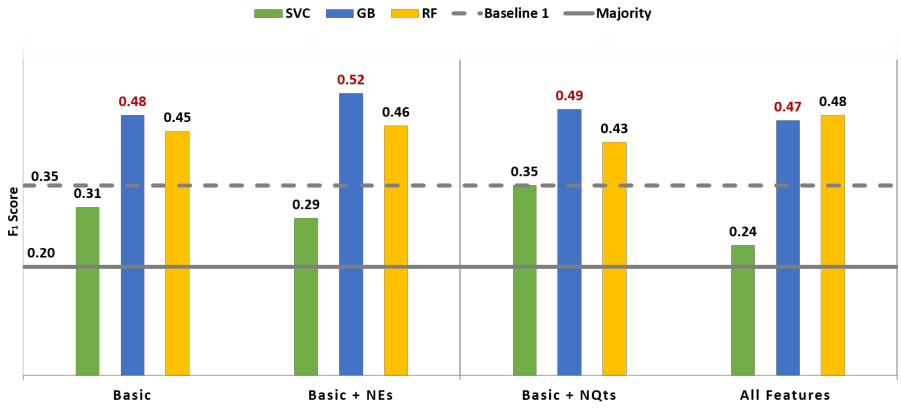

Fig. 4. Subtask B: Performance of classifiers on training data with combinations of features (cosine similarity threshold set to 0.7).

4.4 CLEF Evaluation

Runs As concluded in section 4.3, ensemble classifiers have outperformed SVC classifiers. So, for our runs we picked the GB and RF models. We selected the following models with cosine similarity threshold of 0.7:

1. GB Classifier trained with basic features.
2. GB Classifier trained with basic features and NEs.

We also picked the following models when cosine similarity threshold is set to 0.4:

3. GB Classifier trained with basic features and NQts.
4. RF Classifier trained with basic features and NQts.

Results Table 2 shows our training results compared to the official CLEF testing results. We notice that our best validation model with F_1 score of 0.52 that combines basic features with NEs has achieved lower testing score. Meanwhile, our model that combines basic features with NQts has scored a testing F_1 score of 0.31. The inconsistency between train and test F_1 scores can be justified due to the small training dataset of only 395 webpages. Also, the imbalance in the classes of the dataset could have caused the models to overfit. Our best model that achieved F_1 score value of 0.31 is the highest among all submitted runs for this subtask.

5 Subtask C: Classifying Passages

In this subtask [3], the goal is to extract useful passages for claim verification within the useful retrieved pages. In this section, we present our proposed

Table 2. Subtask B: Performance of CLEF submitted runs.

Features	**Classifier**	F_1 **on Train**	F_1 **on Test**
Basic Features	GB	0.48	0.16
Basic Features + NEs	GB	**0.52**	0.22
Basic Features + NQts	GB	0.47	**0.31**
Basic Features + NQts	RF	0.45	0.30

methodology, experimental evaluation, selected runs for this subtask, and CLEF results.

5.1 Approach

Deciding whether a passage within a useful page is useful or not is a classification problem. Therefore, our methodology is based on using different machine learning classifiers namely SVC, NB, and RF. We consider the following features for this subtask:

- BOW of the passage.
- MostRelSeg in the passage.
- ClmPsgSim.
- NQts in the passage.
- NEs in the passage.

5.2 Experimental Setup

Parameters The three different classifiers mentioned in section. 5.1 were used in our experiments. We set the kernel for SVC to linear, and the number of estimators for the RF models to 100 in all the experiments. For the Gaussian NB, we did not tune any hyperparameters and we based our experiments on the default settings.

Baselines We compare our models against the majority baseline.

5.3 Evaluation on Training

Since we have only 6 claims in the dataset provided for subtask C, which contains only 167 passages from 31 different pages, we considered LOO cross validation in our experiments. We used F_1 score as our evaluation metric. As shown in Fig. 5, SVC outperformed all other models with all groups of features. However, when the BOW features were excluded, the Gaussian NB achieved the best among all. We also observed that the two best performing models are the SVC model when the NEs features were excluded, and the SVC model when the NQts feature was excluded achieving an F_1 score of 0.444 and 0.43 respectively. We also noticed that the performance of the SVC model trained with all features improved compared to when trained with BOW features only, achieving an F_1 score of 0.427 as apposed to 0.387.

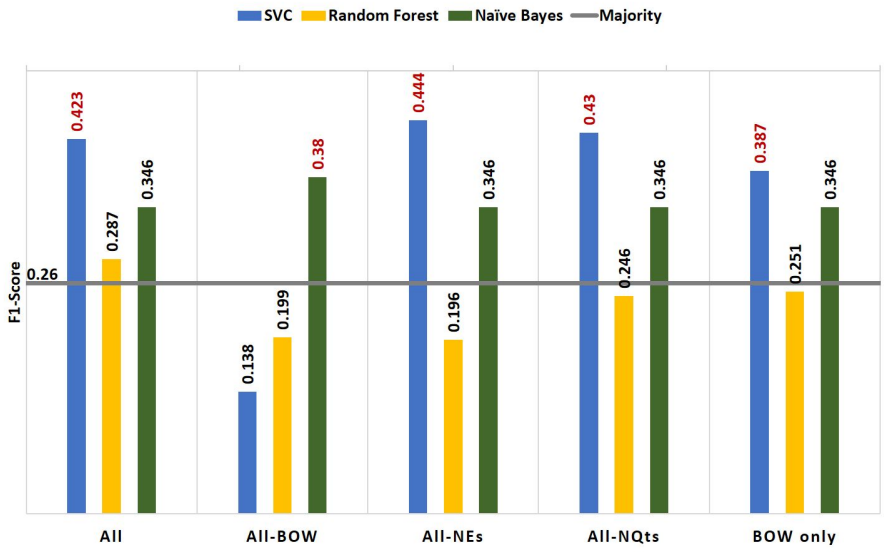

Fig. 5. Subtask C: Performance of classifiers models on training data with combinations of features.

5.4 CLEF Evaluation

Runs As shown in Fig. 5, SVC models outperformed other classifiers except when the BOW features were excluded, in which case the NB model achieved the best F_1 score. Therefore, we picked the 3 best SVC models and the best NB model to submit:

1. SVC trained with all features.
2. SVC trained with all features excluding the NQts feature.
3. SVC trained with all features excluding NEs features.
4. NB trained with all features excluding BOW features.

Results As shown in Table 3, in the official CLEF evaluation, our best performing model in the test phase was the SVC model trained with all features excluding the NQts features, which achieved F_1 score value of 0.4. The low F_1 of our models can be attributed to the big difference in training and testing data including passages from 6 claims and 59 claims respectively. Our highest scoring model is ranked 3^{rd} out of the six runs submitted to the lab, and the maximum score achieved among all runs submitted for this subtask was 0.56.

6 Subtask D: Verifying Claims

The goal of this subtask is to identify whether the claim is "True" or "False". For a claim to be true, it should have supporting evidence that verifies its factuality.

Table 3. SubTask C. Performance of CLEF submitted runs.

Features	**Classifier**	F_1 **on train**	F_1 **on test**
All	SVC	0.423	0.39
All-NQts	SVC	0.43	**0.4**
All-NEs	SVC	**0.44**	0.19
All-BOW	NB	0.38	0.37

In this section, we present our approach, experimental setup, and training results for verifying the claims. Then, we discuss CLEF results for our submitted runs.

6.1 Approach

Deciding the factuality of a claim is a binary classification problem. Therefore, we propose a supervised learning approach using different classifiers: GB, RF and Linear Discriminant Analysis (LDA).

For this subtask, we select the most significant features from webpages to classify the claim. Unlike previous tasks, we consider SntCnt features to find the polarity of the webpage. In addition, we consider the usefulness of the article by using the most *relevant segment* extracted as explained in Section 2 to represent the webpage. In our experiments, we consider the following features for our binary classifiers:

- Similarity Scores: out of all webpages associated with a claim, we only consider three different scores: maximum ClmTtlSim, ClmSnptSim, and MostRelSeg.
- NRelPages.
- For every claim, we select the webpage with maximum MostRelSeg value and extract the following features from it: SrcPop and SntCnt.

6.2 Experimental Setup

Parameters For GB and RF classifiers, we found that the default parameters are the best (based on preliminary experiments). For LDA classifier, we found that using 5 components for linear discrimination is most effective in terms of accuracy.

Baseline As a baseline for this subtask, we implemented Karadzhov et al. [11] method. They classify claims as "True" or "False" based on the top returned search results from several engines. They used an SVC classifier with RBF kernel in their experiments. The inputs to the classifier are word embeddings of the most relevant segment in the webpage, webpage snippet, and the claim. In addition to the word embeddings, the average and maximum similarity scores of the segments and snippets are included as features. We also adopt their method of segment extraction to compare with our approach.

6.3 Evaluation on Training

We conducted multiple experiments to find which features combination will result in the best factuality classification. Due to the limitation in the size of training dataset, we used 8-fold cross validation on all our models for this subtask.

We first experimented with different values of the cosine similarity threshold (0.4, 0.5, 0.6, and 0.7) when extracting relevant segments. In our unreported preliminary experiments, we observed that the best performing models were the ones trained with features extracted using a similarity threshold of 0.6 presented in Fig. 6. We noticed that the GB model trained with all features outperformed all other models. We also observed that our models outperformed the baseline score most of the time except when the NRelPages were excluded from the features. Furthermore, we conclude that NRelPages and SntCnt features are useful in classification of a claim.

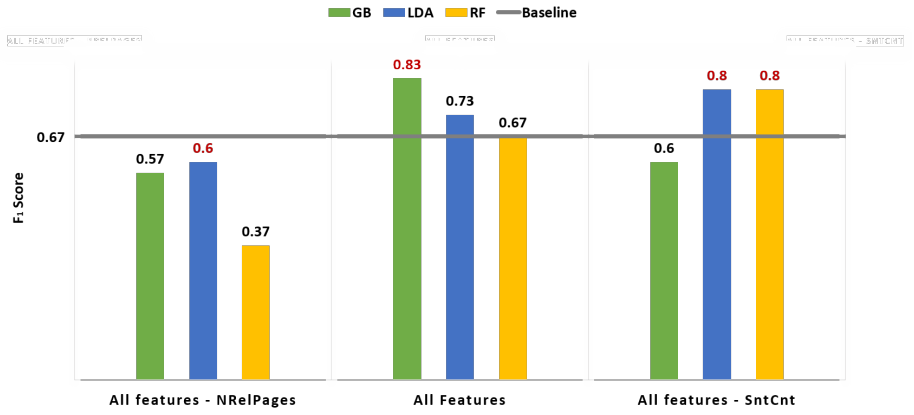

Fig. 6. Subtask D: Performance of classification models on training data with combinations of features (cosine similarity threshold set to 0.6).

6.4 CLEF Evaluation

Runs Based on our training results presented in section 6.3, we decided to use the models trained on all features to classify the claims factuality on testing data. We selected the best ensemble classifiers with two different similarity thresholds.

1. GB classifier, with similarity threshold 0.7.
2. GB classifier, with similarity threshold 0.4.
3. RF classifier, with similarity threshold 0.4.
4. RF classifier, with similarity threshold 0.6.

Results Table 4 shows our training results compared to the official CLEF testing results. Runs for subtask D were submitted over two cycles. In the first cycle, we classify the claims factuality using all webpages provided. In the second cycle, we classify the claims factuality using only useful webpages. We present the results for the second cycle in this section.

As presented in Table 4, we notice that all models achieved very similar F_1 test scores. However, our GB model trained with all features has the highest training and testing scores, achieving F_1 score of 0.91 and 0.53 for training and testing respectively. Our highest scoring model is ranked 4^{th} out of the nine runs submitted to the lab, and the maximum score achieved among all runs submitted for this subtask was 0.62.

Table 4. Subtask D: Performance of CLEF submitted runs.

Features	**Classifier**	F_1 **on Train**	F_1 **on Test**
All	GB	**0.91**	**0.53**
All	GB	0.83	0.51
All	RF	0.80	**0.53**
All	RF	0.66	0.51

7 Conclusion

In this paper, we present our approach for task 2 of CLEF-2019 CheckThat! Lab. For subtask A, we proposed pairwise learning-to-rank approach using different learning models to rank the retrieved pages based on their usefulness. Our best performing model trained using the basic and similarity features (excluding source popularity) achieved an NDCG@20 of 0.55, which is the highest score among 7 runs submitted for this subtask. For subtask B, we proposed a classification model incorporating source popularity feature along with named entities. Our best performing model achieved an F_1 score of 0.31, which is the highest score achieved among the 8 runs submitted for this subtask. For subtask C, we proposed a classification model considering BOW, named entities, and the number of quotes features extracted from passages. Our best performing model trained with all features (excluding the number of quotes) achieved an F_1 score of 0.4 and got 3^{rd} place. For subtask D, we proposed a classification model using sentiment features to find the polarity of the page, in addition to the number of potentially-relevant pages. Our best model trained with all features achieved an F_1 score of 0.53 and got 4^{th} place.

That was our first attempt using a very small training data that was provided by the track organizers. With larger datasets, we plan to improve our classification models with more features including word embeddings, that are trained specifically for this task, and probably with deep learning models as well.

References

1. Atanasova, P., Nakov, P., Karadzhov, G., Mohtarami, M., Da San Martino, G.: Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 1: Check-Worthiness
2. Baly, R., Karadzhov, G., Alexandrov, D., Glass, J.R., Nakov, P.: Predicting Factuality of Reporting and Bias of News Media Sources. CoRR **abs/1810.01765** (2018), http://arxiv.org/abs/1810.01765
3. Elsayed, T., Nakov, P., Barrón-Cedeño, A., Hasanain, M., Suwaileh, R., Da San Martino, G., Atanasova, P.: Checkthat! at clef 2019: Automatic identification and verification of claims. In: European Conference on Information Retrieval. pp. 309–315. Springer (2019)
4. Elsayed, T., Nakov, P., Barrón-Cedeño, A., Hasanain, M., Suwaileh, R., Da San Martino, G., Atanasova, P.: Overview of the CLEF-2019 CheckThat!: Automatic Identification and Verification of Claims. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. LNCS, Lugano, Switzerland (September 2019)
5. Hasanain, M., Suwaileh, R., Elsayed, T., Barrón-Cedeño, A., Nakov, P.: Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 2: Evidence and Factuality
6. Hassan, N., Adair, B., Hamilton, J.T., Li, C., Tremayne, M., Yang, J., Yu, C.: The quest to automate fact-checking. In: Proceedings of the 2015 Computation+ Journalism Symposium (2015)
7. Hassan, N., Arslan, F., Li, C., Tremayne, M.: Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1803–1812. ACM (2017)
8. Hassan, N., Li, C., Tremayne, M.: Detecting check-worthy factual claims in presidential debates. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. pp. 1835–1838. ACM (2015)
9. Hassan, N., Zhang, G., Arslan, F., Caraballo, J., Jimenez, D., Gawsane, S., Hasan, S., Joseph, M., Kulkarni, A., Nayak, A.K., et al.: Claimbuster: The first-ever end-to-end fact-checking system. Proceedings of the VLDB Endowment **10**(12), 1945–1948 (2017)
10. Jaradat, I., Gencheva, P., Barrón-Cedeno, A., Màrquez, L., Nakov, P.: Claimrank: Detecting check-worthy claims in arabic and english. arXiv preprint arXiv:1804.07587 (2018)
11. Karadzhov, G., Nakov, P., Màrquez, L., Barrón-Cedeño, A., Koychev, I.: Fully automated fact checking using external sources. arXiv preprint arXiv:1710.00341 (2017)
12. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
13. 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: International Conference of the Cross-Language Evaluation Forum for European Languages. pp. 372–387. Springer (2018)
14. Popat, K., Mukherjee, S., Strötgen, J., Weikum, G.: Credeye: A credibility lens for analyzing and explaining misinformation. In: Companion of the The Web Conference 2018 on The Web Conference 2018. pp. 155–158. International World Wide Web Conferences Steering Committee (2018)

15. Rashkin, H., Choi, E., Jang, J.Y., Volkova, S., Choi, Y.: Truth of varying shades: Analyzing language in fake news and political fact-checking. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 2931–2937 (2017)
16. Ruchansky, N., Seo, S., Liu, Y.: Csi: A hybrid deep model for fake news detection. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 797–806. ACM (2017)
17. Soliman, A.B., Eissa, K., El-Beltagy, S.R.: Aravec: A set of arabic word embedding models for use in arabic nlp. Procedia Computer Science **117**, 256–265 (2017)
18. Thorne, J., Vlachos, A., Christodoulopoulos, C., Mittal, A.: Fever: a large-scale dataset for fact extraction and verification. arXiv preprint arXiv:1803.05355 (2018)
19. Wang, L., Wang, Y., de Melo, G., Weikum, G.: Five shades of untruth: Finer-grained classification of fake news. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). pp. 593–594. IEEE (2018)
20. Wang, W.Y.: "Liar, liar pants on fire": A new benchmark dataset for fake news detection. arXiv preprint arXiv:1705.00648 (2017)
21. Yasser, K., Kutlu, M., Elsayed, T.: Re-ranking Web Search Results for Better Fact-Checking: A Preliminary Study. In: Proceedings of 27th ACM International Conference on Information and Knowledge Management (CIKM). pp. 1783–1786. ACM, Turin, Italy (2018)
22. Zhi, S., Sun, Y., Liu, J., Zhang, C., Han, J.: Claimverif: a real-time claim verification system using the web and fact databases. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 2555–2558. ACM (2017)