Creating Task-Generic Features for Fake News Detection

Alex C. Olivieri
HES-SO Valais-Wallis,
University of Fribourg
alex.olivieri81@gmail.com

Shaban Shabani
HES-SO Valais-Wallis,
University of Basel
shaban.shabani@unibas.ch

Maria Sokhn
HES-SO Valais-Wallis
maria.sokhn@hevs.ch

Philippe Cudré-Mauroux
University of Fribourg
pcm@unifr.ch

Abstract

Information spreads at a pace never seen before on online platforms, even when this information is fake. Fake news can have substantial impact, for instance when it concern politics and influences the results of legislations or elections. Finding a methodology to verify if some piece of news is true or false is hence essential. In this work, we propose a methodology to create task-generic features that are paired with textual features in order to detect fake news. Task-generic features are created by elaborating on metadata attached to answers from Google's search engine, and by using crowdsourcing for missing values. We experimentally validate our method on a dataset for fake news detection based on the PolitiFact website. Our results show an improvement in F1-Score of 3% over the state of the art, which is significant for a 6-class task.

1. Introduction

The Internet provides many opportunities, but when it comes to news publishing, it also creates many issues. The number of communication channels increases over time, and, after the Internet became a widespread technology, new communication channels such as blogs and social networks emerged in addition to mainstream channels such as newspapers. If this change has positive impact on one hand, because it makes it possible to have access to points of view from different chroniclers, it also increases the possibility that the published news can be fake on the other hand, since the trustworthiness of the information reported is not always evaluated by editorial boards. Moreover, Vosoughi et all. [1] showed, in a study on Twitter, that fake news spread faster and reach 100 times more readers than legitimate news.

The fake news problem can create consequences that are not negligible, and in some domains this can lead to outcomes that impact millions of people. Allcott et al. [2] demonstrated that U.S. citizens believe that social media is currently the "most important" source of information. They also showed that the last months before the US election, fake news favoring Trump were shared 4 times more than the ones favoring Clinton. Even if there is no definitive evidence that this was the main reason behind the result of the election, the last 6 months before the election no forecast was giving Trump as the probable winner, and on average, Clinton had an advantage of 40% points. This gives at least a hint that fake news had at least some impact in the process[1] [2].

Automatic fake news detection has already been studied for some years. In [3], Mihalcea and Strapparava show that through Natural Language Processing (NLP), it is possible to discriminate between false and true information to some degree. While older studies were primarily based on exploring lexico-syntactic patterns, Fend and all [4] applied syntactic stylometry to text, which made it possible to find statistical evidence or syntactic patterns helpful to classify deceptive text. Text analysis is the main resource for fake news detection because of the well-established strategies to analyze text [5] [6]. For some types of news publishing such as social networks, text analysis can be combined with the analysis of metadata attached to the news [7]. In this work when we talk about metadata we refer to the following: *metadata is added information about one or more aspects of the data*, and it is used to summarize basic information about data which can make easier to track or to work with specific data. Machine Learning is currently the main tool to create models based on metadata that can be extracted from news, because of its capability to extract and use effective features from the metadata [8]. However, metadata is not always available for news, and in this case the burden of fake news deception stands on the capacity of analyzing text.

In this work, we explore a new way to obtain metadata about news in order to improve the performance of the news detection task. The first thing users do when they want to verify information is to ask Google. Google, when queried about a topic, provides a

[1]https://projects.fivethirtyeight.com/2016-election-forecast/
[2]https://www.270towin.com/2016-election-forecast-predictions/

URI: https://hdl.handle.net/10125/59956
ISBN: 978-0-9981331-2-6

list of webpages as a result, where each item in the list has the same set of metadata attached to it. Our idea is to create a system that behaves like humans; It queries Google-Custom-Search API and uses the metadata in order to create additional features, which can then be combined with features provided by conventional text analytics methods. In order to have a standardized baseline to evaluate our approach, we use a benchmark dataset for fake news detection based on the PolitiFact web site [9]. This dataset provides political news labeled with 6 classes. Our methodology yields an improvement of **3%** in F1 score over the state of the art, which is significant for a classification task on 6 classes.

The rest of this paper is organized as follows: we dive into the state of the art for fake news detection in Section 2; in Section 3, we describe a benchmark dataset created expressly to be used as a baseline for fake news detection in politics; in Section 4 we describe our method for creating additional features using Google-Custom-Search API; we describe the features we obtain from text analytics in Section 5; we introduce our experimental setting in Section 6 before reporting on our experimental results in Section 7; finally, we conclude and discuss future work in Section 8.

2. Related Work

Fake news detection is defined as the task of classifying news by their veracity. With the advent of online news publication, the context of veracity analysis has evolved - what once could be demonstrated by providing external evidence for the claimed statements is not possible anymore. Fake news is created intentionally and often contains alongside it fake references that compromise previous detection approaches. This issue raises even more the challenge of fake news detection, on which we as humans are always good at as explained in [10], where the authors show with extensive studies that we just perform 4% better than random guesses. Over time, more and more automatic techniques for fake news detection have been developed, and since news is sometimes made of text and metadata that accompanies it, hybrid approaches that analyze both have showed the most promising results [7]. Shu et al. [11] provide a comprehensive review of fake news detection on social media, focusing on the characterization of fake news as well as detection approaches.

The idea behind linguistic approaches for fake news detection based on text is to find predictive deception cues which can help to detect the fakeness of news [12]. Basic linguistic approaches consider only the syntax of the sentences, such as “shallow syntax” for parts of speech [13] or location-based analysis of words [14]. These basic approaches are too simplistic because they rely on isolated n-grams; they are limited as they do not consider the context of the news and the words senses [15]. In [16] and [17] the authors showed that adding semantics to the data analysis improves the capability to properly classify information, but they also show that this methodology is hardly generalizable as it requires a deep knowledge of the domain of interest.

A more promising approach is to complement the linguistic analysis of news with methodologies that use metadata attached to the news. In the domain of social network, a well-established use of metadata is to analyze the social network of interactions such as the behavior of the sources [18] and the patterns in the contents and metadata [19]. Unfortunately, these analysis are possible only in a social media context, where the flow of information is clear and it is possible to have a timeline of the news spreading. In news publishing, it is challenging if not impossible to rebuild such flows of information. In [20] the authors demonstrated that source information is meaningful when trying to solve data conflict, but in the news publishing field often the data about the information flows are missing, which in turn leaves this problem unsolved and makes it impossible to use network analyses. In news publishing, one should instead rely on the metadata already present in the news and on other metadata obtainable using other means.

In [9] the authors describe another major problem when trying to create systems capable of detecting fake news — i.e., it is impossible to compare approaches because there are no readily available benchmark. In their work, they created the first benchmark dataset for fake news detection about politics. They also made experiments on this dataset and, by using a Convolutional Neural Network (CNN) model, achieved a top accuracy of 27% that can be used as a baseline for following experiments. More researchers used this dataset and a recent work [21] reports an accuracy of 41.5% using a Long-Short Term Memory (LSTM) based model that incorporates features extracted from information about the speakers' profiles, which were provided a priori from data annotators, i.e. journalists. However, this kind of information is not normally available within the benchmark dataset, and moreover, the speaker' profile data does not correspond statistically to the dataset itself. In their work, they also state that without considering this information, their model does just 0.5% better than the baseline. To date, no contribution made significant improvement on the aforementioned baseline.

3. Dataset Description

When researches want to inspect unlabeled datasets, crowdsourcing is usually a good choice to label the data in order to obtain training, validation and test sets. The same approach has been used for fake news detections studies. Wang et al. however state that this approach, even if useful, is often unsuitable in the context of fake news because of the mismatch between the training and test datasets, which are created on completely different and simulated platforms [9].

Following this observation, Wang et al. contributed a benchmark dataset called LIAR, which includes 12,836 statements about politics taken from political debates, interviews, news releases, blogs, tweets, etc. The source from the statements is the *POLITIFACT* website[3], winner of the Pulitzer prize, where experts label the statements as belonging to one of six defined classes. The classes, starting from the fakest one to the truest one, are the following: *PantsOnFire, False, MostlyFalse, HalfTrue, MostlyTrue* and *True*. The classification results depend on the presence of evidence to support the statements. An important fact to emphasize is that statements are usually short, on average consisting of 17 words, with the longest statement consisting of 66 words and the shortest one with only 2 words (e.g., speaker: ”Donald-Trump”, statement: ”on immigration”), which makes fake news detection based on text analysis very challenging. The LIAR dataset contains 12,836 statements with metadata. Table 1 shows the number of records for each set.

Source	**N records**	**Percentage %**
Training Set	10,269	80%
Validation Set	1,284	10%
Testing Set	1,283	10%

Table 1. LIAR dataset

The authors then performed some experiments using machine learning classifiers in order to automatically detect fake news. Table 2 shows the results they obtained. The split datasets they created and the results of the six classes are the starting point of our work.

4. Using Google-Custom-Search API to obtain Task Generics Features

In this section, we describe how we used Google's Custom-Search API to create task-generic features that are then used for fake news detection.

[3]http://www.politifact.com/

Models	Valid.	Test
Majority	0.204	0.208
SVMs	0.258	0.255
Logistic Regression	0.257	0.247
Bi-LSTMs	0.223	0.233
CNNs	0.260	0.270
Hybrid CNNs		
Text + Subject	0.263	0.235
Text + Speaker	**0.277**	0.248
Text + Job	0.270	0.258
Text + State	0.246	0.256
Text + Party	0.259	0.248
Text + Context	0.251	0.243
Text + History	0.246	0.241
Text + All	0.247	**0.274**

Table 2. Experiments on LIAR dataset

4.1. Hypotheses on Google Queries

As humans, the first thing we often do when we want to verify information is to query Google, and usually, we find the answer we were looking for in the first links it provides us. Following this intuition, we wondered: could one use similar resources in order to improve the accuracy of fake news detection?

Google's search engine was originally based on PageRank [22], which assess the number and quality of links to a web page to determine a rough estimation of its importance. However, this algorithm mainly measures the popularity of the web page rather than the accuracy of the information it contains. Google improves its searching algorithms continuously [4] in order to increase the relevance for different domains and thus the quality of the results provided. With the rise of fake news, Google engineers started to focus on how to improve the existing algorithms in order to provide more trustworthy results to queries. In [23] a team of Google researchers describe a new algorithm called Knowledge-Based Trust (KBT). KBT pulls facts from web pages to estimate the correctness and accuracy of those facts; it scores the relevance of web pages, and use this scores to determine how trustworthy and reputable a given website is.

Despite its increasing effort in fighting fake news, when querying a statement either fake or true, Google provides as result web pages connected to the query entered by the user. Even if some fake news is queried, the search engine will return a number of facts from further sources that it evaluates as relevant for the query. Furthermore, since users tend to trust Google results, that rank, position and relevance [24] of the results they receive matter when they try to find further

[4]https://moz.com/google-algorithm-change

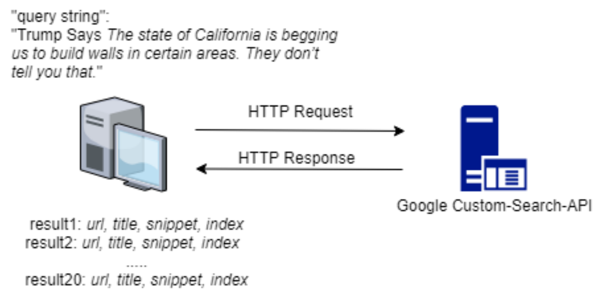

"query string":
"Trump Says The state of California is begging us to build walls in certain areas. They don't tell you that."
HTTP Request
HTTP Response
Google Custom-Search-API
result1: url, title, snippet, index
result2: url, title, snippet, index
.....
result20: url, title, snippet, index

Figure 1. Querying Google's Custom-Search API

evidence related to their queries. Taking these features of Google's results into account, our hypothesis is the following: *for legitimate news, the links that appear at the top of the results are the most trustworthy, and likewise, for fake news, the links that appear at the top of the ranking are the less trustworthy*. With this hypothesis in mind, we started to work our way to obtain meaningful features from the metadata provided by Google's Custom-Search API.

4.2. Metadata provided by Google's Custom Search API

Google's Custom-Search API lets websites and applications retrieve search results from Google programmatically. We use a Python library [5] to issue declarative queries in the same way as searching Google's search engine using a browser. The LIAR dataset provides for each entry, among other data, the full statement as well as its source (the name of the *speaker*). For each of these entries, we generate a compound query that contains both the *statement* itself and its *speaker*. An example of the querying process for one entry of the dataset is shown in Figure 1.

The result set, retrieved in JSON format, contains the first *n* links provided by Google. Each link is associated with the following metadata, which we will use to create our features:

- Title: the title, as found in the HTML header, of the link
- Snippet: the text snippet associated with the news for the given link
- URL: the full URL of the link
- Rank: the rank (position) of the link in the answer provided by Google. E.g. rank = 1 means top result on the first page, while rank = 14 means forth result on the second page.

Before focusing on feature creation, we analyzed a number of results provided by Google. We verified that the results were related to the title of the statement given, thus are overall coherent with the query. Once this correlation was confirmed, we performed more experiments on the result set, which led to the decision that collecting the first 20 results for each statement (the first two pages) was the best strategy. In case 20 results were not available, we collected all the available results. To summarize, for each statement we run a Google search and then collect the metadata associated to the first 20 results (or less if 20 results are not available). In the next subsection, we explain which features we created using this metadata and how we created them.

4.3. Features Creation

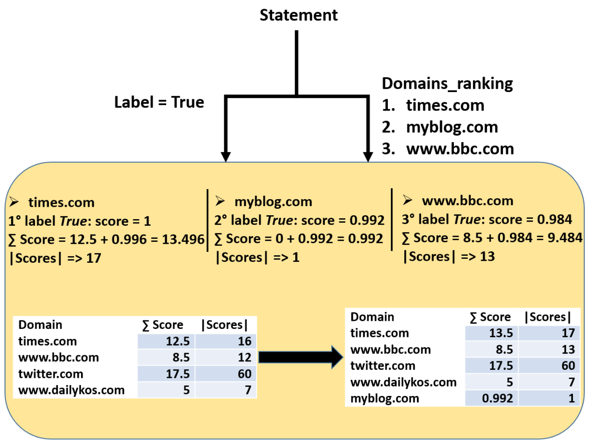
After obtaining metadata on each statement, we started to create the features that we thought were most meaningful for our fake news detection task. These features are: *Statement Domain Score* and *Similarities for Titles and Snippets*. In this subsection, we describe how we created these features.

4.3.1. Statement Domain Score As previously stated, we think that the order in which the results appear on Google is meaningful, hence we propose to use the ranking in order to create a feature we call "statement domains score". First of all we normalized all URLs. While URLs capture the full path to the contents, we wanted to get information about the information provider. For example, we shortened an URL such as *https://www.nytimes.com/interactive/somenews* into its base domain *www.nytimes.com*. Then, we applied a simple process to get a score for each domain from the training set. As we tackled a 6 classes problem, we divided the score we assigned from 0 to 1 into 6 intervals. We then further divided each interval into 20 values (the maximum numbers of domains provided as results from Google) that can be assigned to each domain depending on its rank in the results. For the 3 positive classes, our algorithm assigned values from the highest to the lowest depending on the position in the ranking while, for the 3 negative classes, the values were assigned in reversed order (from the lowest to the highest). Table 3 shows how the intervals were assigned for each label and the list of values the domains get depending on the labels of the statements and their ranking in the Google result set. For each class we took the interval assigned to it (e.g. between 1 and 0.84 for the True class) and we created, with a linspace function, a linearly spaced vector of 20 items where the first item has as value the variable *a*, while the last item has as value the variable *b*.

[5]https://developers.google.com/api-client-library/python/apis/customsearch/v1

Labels	Interval		Values
	a	b	
True	1	0.84	linspace(a,b,20)
MostlyTrue	0.83	0.67	linspace(a,b,20)
HalfTrue	0.66	0.50	linspace(a,b,20)
MostlyFalse	0.34	0.49	linspace(a,b,20)
False	0.17	0.33	linspace(a,b,20)
PantsOnFire	0.01	0.16	linspace(a,b,20)

Table 3. Six classes values split


Statement
Label = True
Domains_ranking
1. times.com
2. myblog.com
3. www.bbc.com
➢ times.com
1° label *True*: score = 1
∑ Score = 12.5 + 0.996 = 13.496
|Scores| => 17
➢ myblog.com
2° label *True*: score = 0.992
∑ Score = 0 + 0.992 = 0.992
|Scores| => 1
➢ www.bbc.com
3° label *True*: score = 0.984
∑ Score = 8.5 + 0.984 = 9.484
|Scores| => 13

Domain	∑ Score	|Scores|
times.com	12.5	16
www.bbc.com	8.5	12
twitter.com	17.5	60
www.dailykos.com	5	7

Domain	∑ Score	|Scores|
times.com	13.5	17
www.bbc.com	8.5	13
twitter.com	17.5	60
www.dailykos.com	5	7
myblog.com	0.992	1

Figure 2. Updating the domain scores

Having defined a range of values for each statement, it is now possible to calculate a score for each domain. Figure 2 shows how, for each statement, scores are assigned for each domain. An algorithm iterates through the result sets of all statements. For each result set, the algorithm extracts the list of domains and assigns to each domain the score corresponding to its ranking in the result set depending on the class assigned to that statement in the dataset (see table 3). Every time the algorithm detects a new domain it creates two variables, the $\sum Score$ and the $|Scores|$. $\sum Score$ is initialized with the score of the first occurrence of this domain while $|Scores|$ acts as a counter and is initialized to 1. Every time a same domain is returned, the $\sum Score$ is updated by adding to its current score the new score as given by our algorithm while $|Scores|$ is incremented by 1. Once the algorithm has iterated through all results, it returns, for each domain, the final *domain_score* by taking the average as shown in equation 1.

$$domain\_score = \frac{\sum Score}{|Scores|} \tag{1}$$

Once the domain scores for all domains have been computed, we compute scores for each statement in the training set by summing the scores of all their associated domains as shown in equation 2.

$$statement\_domain\_score = \frac{\sum_{i=1}^{n} domain\_score(i)}{n} \tag{2}$$

We apply the same principle to compute domain scores for the validation and test set. However, since we obviously do not want to bias our model by looking into validation and test sets when creating the features, when a domain present in these two sets does not have a score assigned from the training set, we simply assign to it the mean score of the other domains for that statement. This is a well-known strategy adopted to address missing values in cases like ours, where the dataset is small and we do not want to lose any example [25].

4.3.2. Similarities Another approach for fake news detection that is getting more and more attention is the so-called Stance Detection. "Stance Detection involves estimating the relative perspective (or stance) of two pieces of text relative to a topic, claim or issue." Evaluating the stance of news that report a statement is an important task in checking its veracity [26]. There is also a worldwide initiative entitled *Fake News Challenge* [6], which allows people to challenge others by providing the best methodology for fake news detection. This initiative sees Stance Detection as the first step towards a solution for fake news detection.

In [6] Horne and Adali showed that the text of the news, its title and their correlations play a role in determining if a news is fake or not. By following this study, for we used each statement the titles and snippets to calculate the three following features based on stance similarities:

- Similarities among all titles for each item in the result set of the query about a statement
- Similarities among all snippets for each item in the result set of the query about a statement
- Similarities between the title and snippet for each item in the result set of the query about a statement.

These features are created using TF-IDF [27] to compute the weighted term frequency vectors, and then to obtain a similarity score using the Cosine Similarity in the Vector Space Model [28].

[6]http://www.fakenewschallenge.org/

5. Additional Features: Text Analysis and Speakers Credit History

In this section, we describe how we created additional features that we used to obtain the final results. They are not novel contributions brought by this work, but they allow us to start from the same point as the experiments made by the previous contributions. Moreover, since text analysis is the main methodology applied in the context of fake news detection there is no reason to skip it.

5.1. Word Frequency using TF-IDF

To create features based on text analysis, we used TF-IDF (see above) as it is a well-established approach that has proved to be efficient for text classification [29] [30]. We computed term-frequency and inverse document-frequency, with the following parameters (see Section 6 for our full experimental setup):

- *max_features=2000*
- *stop_words='english'*
- *tokenizer=word_tokenize*
- *n-grams=(1,5)*
- *analyzer='word'*.

We obtained vectors of dictionary-based features of the same dimension for all statements. We also tried a similar approach by using Doc2Vec, an extension of Word2Vec [31] for working with documents rather than words, but it proved to perform poorly compared to TF-IDF.

5.2. Sentiment Analysis using LIWC

To create groups of features, we used the Linguistic Inquiry and Word Count (LIWC) framework [32], a well-known tool for automatic text analysis. It elaborates on the way in which the statements are written from an emotional and cognitive perspective. LIWC counts, for a given text, the percentage of words that reflect different emotions, thinking styles, social concerns, and parts of speech, using a validated dictionary. It provides a vector of 93 features that include different categories of language such as articles, prepositions, past-tense verbs, numbers, affect, occupation etc., as well as a few other attributes such as the total number of words used.

5.3. Speaker Credit History

Work done by [21] uses the speakers history (credit history), which does not correspond to the dataset itself however, as the data was derived externally from the Politifact API. We also make use of this feature, but, to not bias the results, we do not use the metadata attached to this information for any other feature.

In our approach, we analyzed the credit history individually for each speaker, only considering data from the training set. Credit history vectors contain the counts of accurate and inaccurate statements over the history for each class. For instance, the speaker "Hillary-Clinton" has a credit history vector *ch = [6, 27, 30, 51, 65, 60]* meaning 6 times she made statements that were labeled as *pants-fire*, 27 as *false*, 30 as *mostly-false*, 51 as *half-true*, 30 as *mostly-true*, and 60 as *true*.

An analysis on the credit history of speakers showed that there are on average 4 statements per speaker with a high standard deviation. There are few speakers with a "rich" credit history whereas many of the statements come from speakers with "poor" history (less than 4 statements). There are 1,850 speakers with only 1 statement, while there is a speaker with a maximum of 616 statements. Considering this imbalanced distribution of statements per speaker, we analyzed the speakers' profile in the following way: for each speaker we individually calculated the credit history vector and adjusted it with respect to the global mean credit history vector, which represents the average score (total statements / number of speakers) for each of the 6 classes. Finally, from the vector, we get a single score between 0 and 1, where a score towards 0 indicates the speaker is not trustworthy whereas a score towards 1 indicates he/she is trustworthy.

6. Experimental Setup

The task at hand is a multi-classes classification problem. In this section, we describe the environment used to perform our experiments and validate our model.

6.1. Environment

Our approach is based on the Python programming language, and in order to implement and evaluate it, we used the IPython framework [33], which is a tool for interactive scientific computing. Within IPython we used the Scikit-learn [7] toolkit, which provides the necessary implementations for the machine learning algorithms and for TF-IDF. We used the Gemsim [34] open-source vector space modeling and topic modeling toolkit to implement Doc2Vec. Moreover, we additionally made use of the Pandas library [8] for data processing, and of Numpy [9] for numerical analysis.

[7]http://scikit-learn.org/

[8]http://pandas.pydata.org/

[9]http://www.numpy.org/

6.2. Methodology

We introduced the features we use in section 4 and 5. Table 4 summarizes them and introduces the acronyms that will be used in the rest of the document:

Feature	Acronym
Statements Domain Score	SDS
Similarity among Titles	ST
Similarity among Snippets	SS
Similarity between Titles and Snippets	STS
Sentiment Analysis using LIWC	SA
Text Analysis using TF-IDF	TA
Speakers Credit History	SCH

Table 4. Features List

Before starting the experiments we pre-processed the data. To begin with, since we had to train the models with a high number of features, we performed features scaling and mean-normalization in order to rescale all data into a range between 0 and 1. Subsequently, we chose appropriate machine learning algorithms for our problem. Because of the type of data (textual and numeric), the number of samples and the number of features, we chose the following machine learning classification algorithms: Multinomial Naive-Bayes (MNB) with basics *sklearn* settings, Support Vector Machine (SVM) with polynomial kernel, and Neural Networks (NN) with 2 hidden layer of 25 neurons each and *adam* as weight optimization solver [35], [36].

We tuned each feature by experimenting with different parameters. For each new experiment, we trained all algorithms and then tested them on the validation and finally on the test (holdout) set. Based on the tests, we fine-tuned our features until obtaining optimal F1-Score values. In the following section, we introduce the most important experiments and we discuss their results.

7. Experimental Results

In this section, we describe the experiments that led to our final model. Starting with the features introduced in sections 4 and 5, we performed different experiments by modifying the parameters, fine-tuning the features and obtaining missing information using crowdsourcing.

7.1. Results with all Original Features

Figure 3 shows the results of the first experiment using all our features. The results (around 27% F1) are not satisfying and even lower than the state of the art. To understand the reasons of behind this, we performed

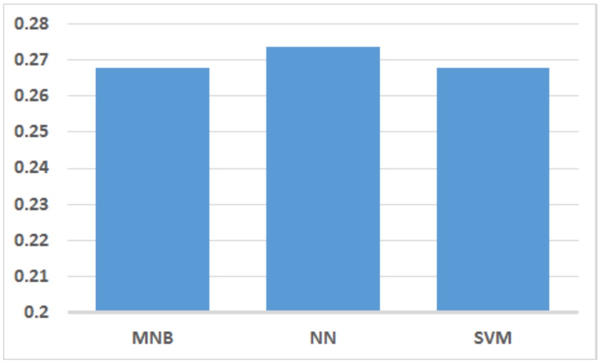


Figure 3. F1 scores using all features and Multinomial Naive-Bayes (MNB), Neural Networks (NN), and Support Vector Machine (SVM)

a round of validation using 10 cross-fold validation on the *training* set and unexpectedly got F1 scores higher than 55%. Since the only features of the training set that had been created by considering the labels were the statement domain score and the speakers' credit history, we decided to check more carefully these two features.

We studied the precision and recall of each individual class showed in Table 5 and noticed that the edge classes (1: true and 6: pants-on-fire) have high precision and low recall, which means that the samples are rarely labeled as belonging to these classes, but when it happens the classification is usually correct. The contrary happens with the central classes. By studying this outcome we assumed that since around 7.3% (1,872 out of 25,404) of the domains in the validation set and 7.1% (1,829 out of 25,404) of the domains in the test sets were missing, and since around 13.8% of speakers were missing as well, our approach for handling missing data (using mean scores from available data) produced an accumulation of values towards the central classes, which in turn led to poor results.

	precision	recall
true	0.35	0.20
mostly-true	0.24	0.25
half-true	0.27	0.32
mostly-false	0.23	0.17
false	0.28	0.41
pants-on-fire	0.38	0.22

Table 5. Best model with all features

7.2. Crowdsourcing Approach to Assign Missing Values

To address the the limitation just mentioned, we decided to use crowdsourcing to assign missing values.

We did it both for the speakers' credit history and for the domain score using the CrowdFlower[10] platform. In order to avoid low-quality answers from crowd workers, we relied on redundancy [37], asking more workers to perform the same task, and aggregating the answers.

For the speakers' credit history, we found that 803 speakers have "poor" credit history with less than the average number of 4 statements/speaker. For each of 803 speakers we generated a HIT task and we asked 5 workers to provide a score between 1 (totally untrustworthy) and 10 (totally trustworthy) for these speakers. The crowd workers were asked to provide their judgment about the speaker based on their findings from web searches. Then we assigned as value for the feature the average of the 5 scores obtained by the online workers, which is an appropriate tradeoff for the number number of tasks to be assigned [38].

Similarly for the domain score, we generated a list of 3,439 websites for which this score was missing (i.e., that do not appear in the training set). For each website in this list, we generated a HIT task and we asked 5 workers to evaluate the reliability of the websites. Workers had to click on the URL provided and to take a deep inspection on the main page of the website in order to judge the web page. Workers had to choose one of the following options:

- broken: if the website was not reachable
- unrelated: if the website was not related to politics
- untrustworthy: if the website was related to politics but seemed untrustworthy
- trustworthy: if the website was related to politics but seemed trustworthy

We found out that there was little variance in the users' judgments; they expressed almost always similar judgments on all domains. We then assigned the following values to the judgments under this reasoning: 0 to *untrustworthy* since websites that create fake news tend to offer statements on politics but misleading information can be found on them[11]; 1 to *broken* because we cannot trust volatile or unreachable web sources, 3 to *unrelated* since even if websites such as web radio stations are not related to politics sometimes they can share opinions and statements that can be trustworthy; and 10 to *trustworthy* because they are related to politics, look legitimate and crowd workers did not find any misleading contents on them.

Table 6 shows the precision and recall for the best F1 score of 0.284, which is obtained with *SVM* after recalculating the features for *SCH* and *SDS* using the crowdsourcing contribution. We notice that even if we managed to diminish the variance in precision and recall among the classes, the increase in F1-score brought by crowdsourcing is limited.

	precision	recall
true	0.29	0.27
mostly-true	0.27	0.26
half-true	0.27	0.31
mostly-false	0.24	0.17
false	0.29	0.38
pants-on-fire	0.42	0.38

Table 6. Best model with all features using also crowdsourcing contributions

To improve the contribution provided by the *SDS* feature, having seen that it performs well in cross-validation, we tried to rebalance the values assigned to the crowdsourcing judgments in order to make the distribution of the training set more similar to the distribution of the validation set. The best fit came when we assigned 1 to *untrustworthy*; 2 to *broken*; 5 to *unrelated*; 10 to *trustworthy*. Figure 4 shows the distribution comparison in all classes for the training set, the validation with the previously assigned values and the validation with the newly assigned values, while table 7 shows the values of F1-score with or without this distribution adaptation.

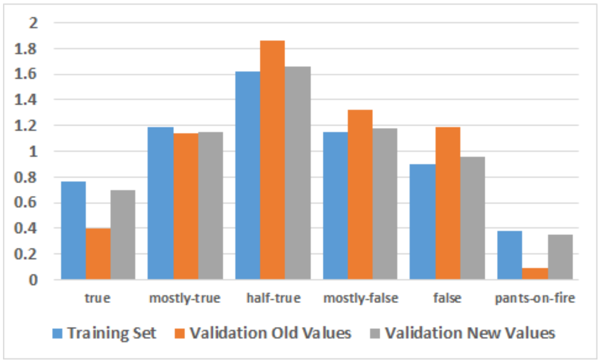


Figure 4. Distribution of domain scores for the Training and Validation sets

However, even if this approach to assign values to the statement domain score works well for the validation set, it obviously does not perform as well when applied to other datasets (e.g. when using the test set).

7.3. Final combination of features

In order to obtain our best model, we decided to experiment with various configurations for the algorithms. The list below shows the combinations of features we found to be the best after doing extensive

[10]www.crowdflower.com
[11]https://www.theguardian.com/technology/2017/aug/18/experts-sound-alarm-over-news-websites-fake-news-twins

F1-score	
Original Distribution	Adapted Distribution
0.284	0.293

Table 7. F1-score comparison for different distributions

experiments, which provide an F1-score of 0.308 – about 3% higher than the state of the art.

- TF-IDF Word Frequency with gram range from 1 to 4 and document frequency higher than 0.02
- LIWC Sentiment Analysis
- Similarity Title-Snippet
- Speakers' Credit History

Table 8 summarizes the results with various features combinations and the comparison with the 2 previous works based on this dataset.

Algorithm	Features	F1-Score
NN	All	0.273
SVM	TA/SA/DS/SCH	0.284
SVM	TA/SA/STS/SS/ST/SCH	0.289
NN	TA/SA/ST/SS/SCH	0.287
SVM	TA/SA/STS/ST/SCH	0.296
SVM	TA/SA/STS/SS/SCH	0.292
SVM	TA/SA/STS/SCH	**0.308**
Hybrid CNN	Baseline LIAR	0.274
LSTM	Long [21]	0.288

Table 8. Test Set results and comparison with the State of the Art

8. Conclusions and Future Work

In this work, we applied a strategy for obtaining task-generic features in order to tackle the fake news detection task. These features are paired to features obtained from text analysis. The foundation of our strategy is to query Google with news statements in order to obtain information provided in form of metadata that can be transformed into features. This approach is domain independent, which means that there is no need to create domain-specific approaches for using the metadata.

A possible limitation of our approach is the dependence on Google's Custom-Search API. IF the API changes the metadata it provides, or the format and schema it uses to provide them, we would have to adapt our approach. In case the change is on the schema, we just need to slightly change our pipeline to adapt it to the new format. Instead if the change is on the metadata provided, two outcomes are possible. If more metadata are provided, this would allow us to improve our approach, potentially. If some metadata get removed, this would most probably negatively affect our method.

Our approach improves on the current state of the art by about 3%. Considering that the work is based on a six classes benchmark dataset, we could assume an even greater impact on a two-class dataset, which is a more common scenario when end-users want to know if they can trust a news story or not. Unfortunately, there is no such dataset readily available on which we could test our system. However, since some work on two-class fake news scenarios using text analysis showed results slightly higher than 80% in terms of F1-score, we believe that our approach could be leveraged to create systems that provide high-accuracy results, and thus could contribute in the battle against the spread of fake news. Moreover, we want to point out that we took a number of new statements from the Politifact website and tried to apply our approach to them, yielding similar results as the ones we got on the Liar benchmark dataset. Therefore, this shows that our approach works for both older and newer statements.

One idea to improve our approach, besides increasing the size of the dataset to train a better models, would be to create open repositories of trustworthiness scores for web domains. With this idea in mind, we plan as future work to start working on an initiative to allow trusted users to rate web domains in order to create a repository which could subsequently be used by fake news detection approaches.

Acknowledgement

This paper was supported by the Hasler Foundation in the context of the City-Stories project.

References

[1] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018.

[2] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–36, 2017.

[3] R. Mihalcea and C. Strapparava, "The lie detector: Explorations in the automatic recognition of deceptive language," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pp. 309–312, Association for Computational Linguistics, 2009.

[4] S. Feng, R. Banerjee, and Y. Choi, "Syntactic stylometry for deception detection," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 171–175, Association for Computational Linguistics, 2012.

[5] B. Pang, L. Lee, *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[6] B. D. Horne and S. Adali, "This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news," *arXiv preprint arXiv:1703.09398*, 2017.

[7] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[8] S. Gilda, "Evaluating machine learning algorithms for fake news detection," in *Research and Development (SCOReD), 2017 IEEE 15th Student Conference on*, pp. 110–115, IEEE, 2017.

[9] W. Y. Wang, "Liar, liar pants on fire": A new benchmark dataset for fake news detection," *arXiv preprint arXiv:1705.00648*, 2017.

[10] C. F. Bond Jr and B. M. DePaulo, "Accuracy of deception judgments," *Personality and social psychology Review*, vol. 10, no. 3, pp. 214–234, 2006.

[11] K. Shu *et al.*, "Fake news detection on social media: A data mining perspective," *SIGKDD Explor. Newsl.*, vol. 19, pp. 22–36, Sept. 2017.

[12] V. Rubin *et al.*, "Fake news or truth? using satirical cues to detect potentially misleading news," in *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pp. 7–17, 2016.

[13] D. M. Markowitz and J. T. Hancock, "Linguistic traces of a scientific fraud: The case of diederik stapel," *PloS one*, vol. 9, no. 8, p. e105937, 2014.

[14] J. T. Hancock, M. T. Woodworth, and S. Porter, "Hungry like the wolf: A word-pattern analysis of the language of psychopaths," *Legal and criminological psychology*, vol. 18, no. 1, pp. 102–114, 2013.

[15] D. F. Larcker and A. A. Zakolyukina, "Detecting deceptive discussions in conference calls," *Journal of Accounting Research*, vol. 50, no. 2, pp. 495–540, 2012.

[16] A. C. Olivieri *et al.*, "Assessing data veracity through domain specific knowledge base inspection," in *Advanced Computer Science and Information Systems (ICACSIS), 2017 International Conference on*, pp. 291–296, IEEE, 2017.

[17] V. W. Feng and G. Hirst, "Detecting deceptive opinions with profile compatibility," in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 338–346, 2013.

[18] D. M. Cook *et al.*, "Twitter deception and influence: Issues of identity, slacktivism, and puppetry," *Journal of Information Warfare*, vol. 13, no. 1, pp. 58–IV, 2014.

[19] Z. Papacharissi and M. de Fatima Oliveira, "Affective news and networked publics: The rhythms of news storytelling on egypt," *Journal of Communication*, vol. 62, no. 2, pp. 266–282, 2012.

[20] X. L. Dong *et al.*, "Integrating conflicting data: the role of source dependence," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 550–561, 2009.

[21] Y. Long *et al.*, "Fake news detection through multi-perspective speaker profiles," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, vol. 2, pp. 252–256, 2017.

[22] L. Page *et al.*, "The pagerank citation ranking: Bringing order to the web.," Technical Report 1999-66, Stanford InfoLab, November 1999.

[23] X. L. Dong *et al.*, "Knowledge-based trust: Estimating the trustworthiness of web sources," *CoRR*, vol. abs/1502.03519, 2015.

[24] B. Pan, H. Hembrooke, T. Joachims, L. Lorigo, G. Gay, and L. Granka, "In google we trust: Users' decisions on rank, position, and relevance," vol. 12, pp. 801 – 823, 06 2007.

[25] M. Saar-Tsechansky and F. Provost, "Handling missing values when applying classification models," *Journal of machine learning research*, vol. 8, no. Jul, pp. 1623–1657, 2007.

[26] B. Riedel *et al.*, "A simple but tough-to-beat baseline for the fake news challenge stance detection task," *arXiv preprint arXiv:1707.03264*, 2017.

[27] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 133–142, 2003.

[28] K. Erk and S. Padó, "A structured vector space model for word meaning in context," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 897–906, Association for Computational Linguistics, 2008.

[29] W. Zhang *et al.*, "A comparative study of tf* idf, lsi and multi-words for text classification," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2758–2765, 2011.

[30] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 133–142, 2003.

[31] Y. Goldberg and O. Levy, "word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method," 2014.

[32] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn, "The development and psychometric properties of liwc2015," tech. rep., 2015.

[33] F. Pérez and B. E. Granger, "Ipython: a system for interactive scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, 2007.

[34] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, ELRA, May 2010.

[35] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *Journal of advances in information technology*, vol. 1, no. 1, pp. 4–20, 2010.

[36] S. B. Kotsiantis *et al.*, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[37] P. G. Ipeirotis *et al.*, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, (New York, NY, USA), pp. 64–67, ACM, 2010.

[38] H. Li and Q. Liu, "Cheaper and better: Selecting good workers for crowdsourcing," in *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015.