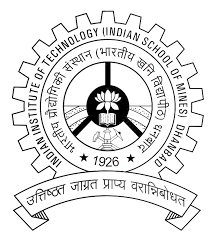
**FAKE ARTICLE DETECTION USING STANCE IDENTIFICATION**

**Semester: VIII**

**Session: 2017-2018**

**PROJECT GUIDE**

**PROF. G.P.BISWAS**

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY (INDIAN SCHOOL OF MINES)**

**DHANBAD-826004**

**DATE: 09/05/2018**

**SUBMITTED BY**

**SHARATHCHANDRA NANDIKONDA 14JE000443**

**PRAVEEN KURAPATI 14JE000486**

**VIKASH RAJPUT 14JE000374**

**PARVIN MOR 14JE000087**

**CERTIFICATE**

This is to certify that the dissertation titled “FAKE ARTICLE DETECTION USING STANCE IDENTIFICATION” is being submitted to Indian Institute of Technology (Indian School of Mines), Dhanbad, by Mr. Sharathchandra Nandikonda(Adm no.14JE000443), Mr. Praveen Kurapati(Adm no. 14JE000486), Mr.Vikash Rajput(Adm No.14JE000374) and Mr.Parvin Mor(Adm No.14JE000087) in partial fulfilment of their Bachelor of Technology degree in Computer Science & Engineering of the same institution incorporates the results of their own work, carried out under my supervision and guidance. This dissertation has not been submitted for any other degree elsewhere to the best of my knowledge.

**Prof. G.P. Biswas**

…………………………

Professor

Department of Computer Science & Engineering

IIT(ISM) Dhanbad

Jharkhand- 826004

**DECLARATION**

**Date: 09/05/2018**

We, hereby declare that the thesis submitted is our own work and that we have exercised reasonable care to ensure that the work is original and to the best of our knowledge it does not breach any law of copyright and has not been accepted for the award of any degree or diploma of any other universities or other institutes of higher learning. To the extent, if any work from other papers or thesis is related, such work has been cited and acknowledged within the text of our work.

SHARATHCHANDRA NANDIKONDA 14JE000443

PRAVEEN KURAPATI 14JE000486

VIKASH RAJPUT 14JE000374

PARVIN MOR 14JE000087

**ACKNOWLEDGEMENT**

We express our deep sense of gratitude and respect towards our project mentor, Prof. G.P. Biswas, Professor, Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad. We are very grateful for the generosity, expertise and guidance we have received from him while working on the project. Expressing our sincere thanks to our mentor, we would like to convey our sincere regards and gratitude for encouraging us to undertake this project and to work on it at time and again, it would not have been possible without him. We have all but enjoyed working on this project under his wings and we shall keep so for the times to come.

We would also like to express deep gratitude to Professor P.K. Jana, Head of the Department of Computer Science & Engineering for his cooperation during the course work. We were beneficiary of essential advice and assistance of the faculty and staff and Indian Institute of Technology (Indian School of Mines), Dhanbad. We also owe thanks to all persons who directly and indirectly helped me during the work till now.

Date: 09/05/2018

**SHARATHCHANDRA NANDIKONDA 14JE000443**

**PRAVEEN KURAPATI 14JE000486**

**VIKASH RAJPUT 14JE000374**

**PARVIN MOR 14JE000087**

**CONTENTS**

**ABSTRACT 6**

**1.INTRODUCTION 7-10**

**1.1 Fake news and stance detection 7**

**1.2 Stance detection dataset for fake news classification 8**

**2. FEATURE GENERATION 10-13**

**2.1 Term Frequency-Inverse Document Frequency 10**

**2.2 The vector space model and tf-idf 11**

**2.3 Cosine Similarity 13**

**3. PROCEDURE 14-15**

**4. IMPLEMENTATION DETAILS 15-17**

**5. RESULT 18**

**6. ROLE OF STANCE DETECTION 19**

**8. REFERENCES 20**

**ABSTRACT**

We created an easy-to-use system to detect the credibility of a user’s claim or article, based on the concept of stance detection. The goal of this project is to identify whether given headline-article pairs: (1) agree, (2) disagree, (3) discuss the same topic, or (4) are not related at all. A machine learning model in Tensorflow that’s based off of several research papers in the field of stance detection is built. Our model uses a combination of Bag-of-Words, Google’s word-2-vec, TF, TF-IDF (Term Frequency, Inverse Document Frequency), and ‘stopwords’ inside Scikit-learn to vectorize our input. That is run through a neural network single hidden layer with ReLU activation, a fully connected layer and a softmax activation function to produce one of 4 outputs.

**1. Introduction**

**1.1 Fake news and stance detection**

With the advent of fake news being used to influence elections, the identification of false information has become an important task. Governments, newspapers and social media platforms are working hard on distinguishing credible news from fake news. The goal of the Fake News Challenge is to automate the process of identifying fake news by using machine learning and natural language processing. This process can be broken down into several stages. A first helpful step towards the identification of fake news is to understand what other news sources are saying about the same topic . That is why the fake news challenge initially focuses on stance detection. **Stance detection** comprises the estimation of the relative perspectives of two different text pieces on the same topic as described. Specifically, the task is to estimate the stance of a news headline, relative to the contents of a news article which can but does not have to address the same topic. Thus, the relative stance of each headline-article pair has to be classified as either **unrelated, discuss, agree or disagree**. The discovery of a disagreeing headline-article pair does not necessarily correspond to the discovery of a fake article but it is an automated first step which could make human reviewers aware of a discrepancy. Human reviewers or specialized algorithms can then ultimately decide which articles are fake.

**Input**

A headline and a body text - either from the same news article or from two different articles.

**Output**

Classify the stance of the body text relative to the claim made in the headline into one of four categories:

* **Agrees**: The body text agrees with the headline.
* **Disagrees**: The body text disagrees with the headline.
* **Discusses**: The body text discuss the same topic as the headline,but does not take a position
* **Unrelated**: The body text discusses a different topic than the headline

**1.2 Stance detection dataset for fake news classification**

In order to understand and effectively solve the problem of stance detection between headline-article pairs, it is crucial to get an in-depth understanding of the dataset.

**Dataset Requirements:**

1. Availability of both truthful and deceptive instances.

2. Verifiability of ‘ground truth’.

3. Homogeneity in lengths.

4. Homogeneity in writing matter.

5. Predefined timeframe.

6. The manner of delivery (e.g. humor, sensational, newsworthy).

To deal with some of these challenges, we outsource some of corpus definitions to the website fakenewschallenge.org which compiles an ongoing list of fake and trusted news sources.

The dataset consists of about 50,000 headline-article pairs each labeled with either unrelated, discuss, agree or disagree.

|  |  |
| --- | --- |
| **Headline** | **Robert Plant Ripped up $800M Led Zeppelin Reunion Contract** |
| **Unrelated** | … Richard Branson’s Virgin Galactic is set to launch SpaceShipTwo today. ... |
| **Agree** | … Led Zeppelin’s Robert Plant turned down £500 MILLION to reform supergroup. ... |
| **Disagree** | ... No, Robert Plant did not rip up an $800 million deal to get Led Zeppelin back together. ... |
| **Discuss** | ... Robert Plant reportedly tore up an $800 million Led Zeppelin reunion deal. ... |

Two observations must be made here to address challenges down the line.

First, the dataset is highly unbalanced with most of the pairs being unrelated and only 1.68% of all pairs labeled disagree. As shown in the result section, this can influence results and training time, but can be partially corrected.

Second, the pairs have been generated by repeatedly and randomly pairing headlines and articles. This means that one headline may be paired with multiple articles and vice versa. It turned out to be impossible to split the dataset without either articles or headlines appearing in both training and test sets. Otherwise, a significant number of training examples could not have been used.

The dataset is cleanly split between both sets, but not headlines. Splitting by headlines or randomly would increase data bleeding significantly. Thus, the same headlines appear in both the train and test splits and until the final test set.

In addition to these official statistics, we also analyze the lengths of the headlines and articles in the dataset. This helped us choose a reasonable truncation length for articles and headlines in order to avoid vanishing gradients and to speed up training without losing too much information.

Most headlines are about 10 words long and most articles are about 250 words long. The headline lengths are maximally 40 words long. The article lengths are mostly below 700 words and only a few outlier articles exceed that threshold. Intuitively, one might think of clipping the articles at 700 words and to not clip the headlines at all, which would leave over 95% of all articles unclipped. However, as shown in the results section, it is better to clip the articles at 200 words in order to optimize speed and performance.

We further analyze which and how many words in this dataset do not match with any of the word-2-vec vector representations that will be used for our model. Most of these instances were words that were connected without a space, such as fidelcastro or relatedhalloween, and random sequences such as kzhsbgw87a or mhcztvzdfd, which likely correspond to unfiltered hyperlinks. A significant number of unknown words also contained named entities such as safira or 9to5mac, which is typical for a news dataset as news article often report on new named entities which might not have been included at the time of the word-2-vec vector training.

In order to get a deeper understanding of the performance of our models, we report multiple metrics as well as confusion matrices.

**2.Feature Generation**

Our approach evaluates the performance of models trained on three feature sets:

* Bigram Term Frequency-Inverse Document Frequency

For feature generation, we rely on the relatively new Spacy Python package to conduct tokenization, part-of-speech tagging, syntactal parsing, and named entity recognition. Spacy is implemented in Cython (a superset of the Python language that allows C code for be generated from Python using the Python/C API), allowing for very fast performance compared to other NLP packages such as NLTK.

Several evaluations from peer-reviewed journals find that Spacy achieves performance on parsing and entity recognition tasks that is comparable to other widely-used tools, while having significant advantage on speed. This is why we chose to use Spacy over more established options such as the Java implementation of Stanford's Probabilistic Context Free Grammar.

From the raw article text, we use Spacy and SciKit Learn to generate the relevant features. We utilize Spacy’s support for multi threading to parallelize the feature generation process and SciKit Learn’s pipeline feature to create fit-transform methods that can be used on the training data and then applied to the test set.

**2.1 Term Frequency-Inverse Document Frequency**

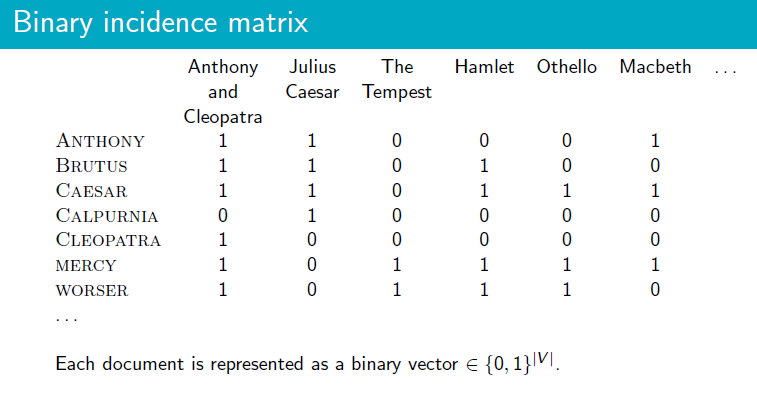
The first feature set is vectorized bigram Term Frequency-Inverse Document Frequency. This is weighted measure of how often a particular bigram phrase occurs in a document relative to how often the bigram phrase occurs across all documents in a corpus.

Because of the political nature of our corpus, we want to limit the model's knowledge of the people and institutions mentioned in the article text. Otherwise, we risk the model simply learning patterns such as "Clinton corrupt" which describe the topic and viewpoint of the text, rather than the outcome of interest (is this source reliable or not). Additionally, these patterns will be highly sensitive to the particular news cycle. To address this concern, we introduce step during tokenization to use Spacy's named entity recognition to replace all mentions of named entities with a placeholder, e.g. <-NAME-> or <-ORG->.

We use SKLearn to calculate the TF-IDF for each bigram within each document and build sparse matrix of the resulting features. To keep the dimensionality of our data to a manageable size, we limit the vocabulary to only consider the top 3000 terms ordered by term frequency across the entire corpus. We did not experiment with different methods or thresholds for selecting the terms included in the vocabulary, or with different lengths of n-grams, but this may be an area to explore in future work.

**2.2 The vector space model and tf-idf**

A document vector captures the relative importance of the terms in a document. The representation of a set of documents as vectors in a common vector space is known vector space. Queries as vectors in the same vector space as the document collection.



Binary incidence matrix just tells us whether the term is present in the document or not and does not give any information about how many times each term/token occurs.

**The term frequency *tf*** of term t in document d is defined as the number of times that t occurs in d. But raw *tf* is not used directly, because it is not a relevant measure as the value of it is not directly related to relevance of term.

So, a logarithm of the tf is used as a weight measure it can be given as:

In addition to using tf we need to include a weight for rarity of term being searched, it can be implemented using *idf*.

*df* is the **document frequency**, the number of documents that *t* occurs in. *df* is an inverse measure of the informativeness of term *t.*

we define *idf*  weight given by

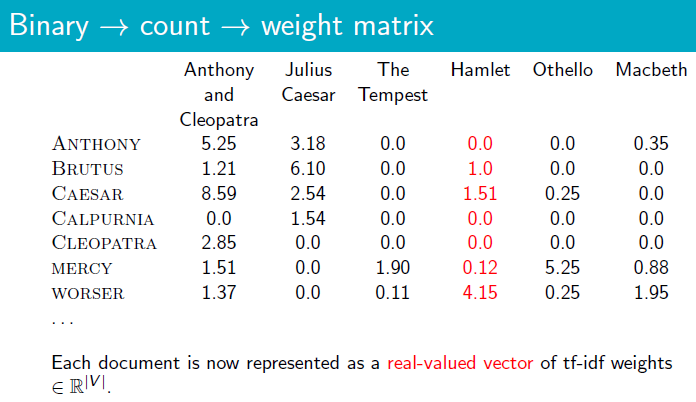
*idf =*

N is the number of documents in the collection. *idft* is a measure of the informativeness of the term.

The ***tf-idf*** weight of a term is the product of its tf weight and its *idf* weight.

**The tf-idf weight**

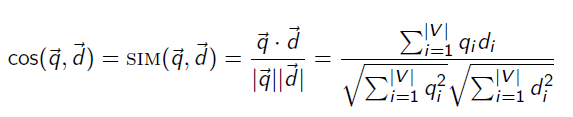
* increases with the number of occurrences within a document.(termfrequency)
* increases with the rarity of the term in the collection. (inverse document frequency).

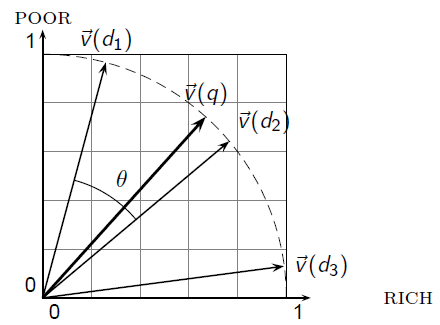


**2.3 Cosine Similarity**

Each document/ query can be given as vector of term with each value being *tf-idf* weight of term with document collection.

To find the similarity or relativity of query and document pair, we need a similarity measure. The cosine similarity is the best way to do it.



Normalization of each document is done to make it independent of size.

**3. PROCEDURE**

Let z be the number of headline-body pairs in the training set. Then, for any headline-body pair j, let x1,x2,...,xnj be the sequence of words or tokens (punctuation and other special characters) in the headline. Similarly, let y1,y2,...,ymj be the sequence of words or tokens in the corresponding body. nj is the total number of words in the headline and mj is the number of words in the body. We represent each word or token with a 1×d dimensional pretrained dense vector or embedding. Let d1,...,dnj be the embeddings of words in the headline and let e1,...,emj be the embeddings of words in the body. di, ei ∈Rd.

aj =

Correspondingly, let bj be the encoding of body. Then:

bj =

Once these representations have been constructed for all pairs z, we concatenate [aj bj] into a 1×2d vector wj ∈R2d  by tf-idf computations and feed it as input to a feedforward neural net.

h = f(wjW1 + b1)

hdrop = dropout(h)

pˆj = g(hdropW2 + b2)

Here, W1 ∈ R2d×H, b1 ∈ RH, W2 ∈ RH×4, and b2 ∈ R4.

H is the hidden layer size

f(x) is the ReLU (Rectiﬁed Linear Unit) non-linearity

g(x) is the softmax function.

The dropout layer randomly drops neurons while training. It helps reduce overﬁtting.

Finally, pˆj is a 1 × 4 vector denoting the probability of pair j belonging to each class (agree, disagree, discuss, and unrelated).

We evaluate the loss using a simple cross entropy penalty. If pj is a one-hot vector denoting the true class of pair j, then:

Most of the following models build on this architectural framework. In subsequent subsections, we abstract away from the dimensions of individual matrices and vectors to focus more on the overall structure.

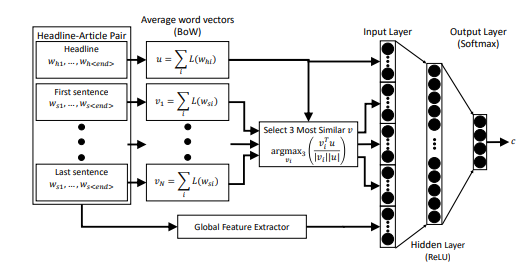


Fig. Bag of Words with global features model. L is an embedding

**4. IMPLEMENTATION DETAILS**

Our project was implemented in Python using Anaconda software. The dataset was collected from fakenews.org website. It has basically two csv files which contains pairs of headline and body text with the appropriate class label for each. The data provided is (headline, body, stance) instances, where stance is one of {unrelated, discuss, agree, disagree}. The dataset is provided as two CSVs:

**train\_bodies.csv**

This file contains the body text of articles (the articleBody column) with corresponding IDs (Body ID).

**train\_stances.csv**

This file contains the labeled stances (the Stance column) for pairs of article headlines (Headline) and article bodies (Body ID, referring to entries in train\_bodies.csv).

**Dataset class**

The dataset class reads the FNC-1 dataset and loads the stances and article bodies into two separate containers.

dataset = DataSet()

You can access these through the .stances and .articles variables

print("Total stances: " + str(len(dataset.stances)))

print("Total article bodies: " + str(len(dataset.articles)))

.articles is a dictionary of articles, indexed by the body id. For example, the text from the 144th article can be printed with the following command: print(dataset.articles[144])

**Hold-out set split**

Data is split using the generate\_hold\_out\_split() function. This function ensures that the article bodies between the training set are not present in the hold-out set. This accepts the following arguments. The body IDs are written to disk.

dataset - a dataset class that contains the articles and bodies

training=0.8 - the percentage of data used for the training set (1-training is used for the hold-out set)

base\_dir="splits/"- the directory in which the ids are to be written to disk

**k-fold split**

The training set is split into k folds using the kfold\_split function. This reads the holdout/training split from the disk and generates it if the split is not present.

dataset - dataset reader

training = 0.8 - passed to the hold-out split generation function

n\_folds = 10 - number of folds

base\_dir="splits" - directory to read dataset splits from or write to

This returns 2 items: a array of arrays that contain the ids for stances for each fold, an array that contains the holdout stance IDs.

**Getting headline/stance from IDs**

The get\_stances\_for\_folds function returns the stances from the original dataset. See fnc\_kfold.py for example usage.

**Scoring Classifier**

The report\_score function is based off the original scorer provided in the FNC-1 dataset repository.

report\_score expects 2 parameters. A list of actual stances (i.e. from the dev dataset), and a list of predicted stances (i.e. what you classifier predicts on the dev dataset). In addition to computing the score, it will also print the score as a percentage of the max score given any set of gold-standard data (such as from a fold or from the hold-out set).

predicted = ['unrelated','discuss',...]

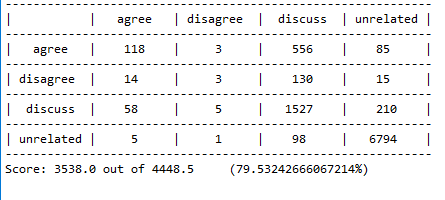
actual = [stance['Stance'] for stance in holdout\_stances]

report\_score(actual, predicted)

This will print a confusion matrix and a final score your classifier. We provide the scores for a classifier with a simple set of features which you should be able to match and eventually beat.

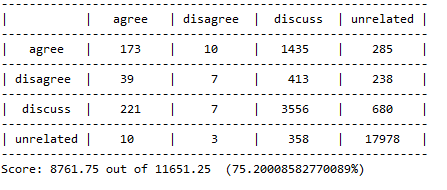
**5. RESULT**

Taking inputs from train.csv and training the model with article document pair we got the following results presented in the form of a confusion matrix shown below.



With the score 3538.0 we successfully categorised those articles with 79%.

For test dataset the model achieved the following results



With the score 8761.75 we successfully categorised those articles with 75%.

**6. ROLE OF STANCE DETECTION FOR IDENTIFICATION OF FAKE NEWS ARTICLES**

There are two important ways the Stance Detection task is relevant for fake news.

1. Gathering the relevant background information about a claim or news story, including all sides of the issue, is a critical initial step in a human fact checker’s job. One goal of the Fake News Challenge is to push the state-of-the-art in assisting human fact checkers, by helping them quickly gather the information they need to make their assessment.

In particular, a good Stance Detection solution would allow a human fact checker to enter a claim or headline and instantly retrieve the top articles that agree, disagree or discuss the claim/headline in question. They could then look at the arguments for and against the claim, and use their human judgment and reasoning skills to assess the validity of the claim in question. Such a tool would enable human fact checkers to be fast and effective.

1. It should be possible to build a prototype post-facto “truth labeling” system from a “stance detection” system. Such a system would tentatively label a claim or story as true/false based on the stances taken by various news organizations on the topic, weighted by their credibility. For example, if several high-credibility news outlets run stories that Disagree with a claim (e.g. “Denmark Stops Issuing Travel Visas to US Citizens”) the claim would be provisionally labeled as False. Alternatively, if a highly newsworthy claim (e.g. “British Prime Minister Resigns in Disgrace”) only appears in one very low-credibility news outlet, without any mention by high-credibility sources despite its newsworthiness, the claim would be provisionally labeled as False by such a truth labeling system. In this way, the various stances (or lack of a stance) news organizations take on a claim, as determined by an automatic stance detection system, could be combined to tentatively label the claim as True or False. While crude, this type of fully-automated approach to truth labeling could serve as a starting point for human fact checkers, e.g. to prioritize which claims are worth further investigation.

7. REFERENCES

[1] Stance Detection for Fake News Identification - Damian Mrowca, Elias Wang

<https://web.stanford.edu/class/cs224n/reports/2760496.pdf>

[2] Neel Rakholia and Shruti Bhargava, “Is it true? – Deep Learning for Stance Detection in News”.

[3] S. Tavernise, “As fake news spreads lies, more readers shrug at the truth,” Dec 2016.

<https://www.nytimes.com/2016/12/06/us/fake-news-partisan-republican-democrat.html>

[4] 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, p. 14, 2015.

[5] “Fake news challenge stage 1 (fnc-i): Stance detection.”

https: //www.fakenewschallenge.org/