Political Promise Evaluation (PPE)

Bhanu Jain, Rohit Begani

December 11, 2017

Abstract

PPE provides a unified view of promises made by Political Candidates while campaigning and the promises fulfilled during their tenure. PPE scours and summarizes Political Promises which are then matched with relevant articles to predict the current status of the the promise. The newspaper articles were extracted from the New York Times corpus and matched with the appropriate political promise to allow correct analysis of the promise in focus.

1 Introduction

Elections play a huge role in deciding the growth path of a country for a long time to come. Selecting the right candidate can help a country take exponential leaps in their growth while inversely it can cause the country to lose out on years of progress.

We aim to provide unbiased natural language processing based evaluation for Political Candidates' Campaign Promises. This evaluation can be invaluable in helping people decide who to vote for specially during re-election. Moreover this evaluation tool provides an easy way for people to keep track of Candidate's promises and keep him accountable for the same.

As a general newspaper reader it's usually very difficult to dig-in enough to read up on the less scandalous promises. Rather than showcasing the more click-bait articles we aim to provide a more objective evaluation about all the promises made by a candidate in a concise and unambiguous form.

In this age of Internet and due to the presence of multiple digital news sources, we can access real-time information about the world. Due to the vast influx of information it's usually very easy to miss out on important information. We extract all the promises made by a candidate during their campaign from online sources. The articles are extract from the New York Times corpus. These articles are then matched with the extracted promises and matched with their relevant promise. The articles matched with distinct promises are then analyzed to understand the current status of the promise.

2 Problem Description

Our complete objective can be stated in a succinct form as:

- Extract all the promises made by a candidate.
- Extract articles and texts related to promises after the candidate has taken office.
- Classify articles with respect to each promise.
- Evaluate articles to measure promise performance.

As an example, a promise made by Donald Trump is, *"Build a wall on the USA and Mexico border."* A matching article related to this promise can be *"Article on the wall between USA and Mexican border in Reuters[12]"*. Now this article will be analyzed with the methods explained further to evaluate the performance of the mentioned promise.

3 Background and Related Work

A lot of work is currently being done to understand the context of a textual information, recently a couple of facebook engineers have been working to understand the text of users post[17]. There are a few online softwares also which attempt to find meaning from textual data like TextRazor[7] which attempt to extract meaning from textual data but are mostly focused on context matching in a generic form.

There are multiple manually managed websites online which are dedicated towards listing all the promises made by prominent Political Candidates and tracking the progress of the candidates in regards to the promise. A few of them are TrumpTracker[9] which is focused on the work of President Donald Trump, TrudeauTracker[8] which focuses on Justin Trudeau and Politifact[4] which isn't as specific as the previous two, rather it tracks the work of multiple current and former candidates.

There has been a lot of research done to understand the effect of newspaper articles on the users and how it affects their perception, sentiment analysis has been used in past to understand it. But none of it tries to parse the newspaper content to actually understand the context in which political promises are being talked about[14][16]. There can be multiple situations where a politician fulfills his promise but if it's not a populist decision then it might have a negative sentiment even if the candidate did end up fulfilling his promise to the public.

Our implementation aims to improve upon the existing systems by reading the newspaper articles and understanding the context of a specific term in the article. So for example if our promise is "Building a wall between USA and Mexico" then our application would attempt to point out every point in the article where that topic is being talked about and then understand the context of the article. By understanding the context we can understand if the newspaper is talking about a promise being fulfilled, broken or still in the progress.

4 Methodology

The system is divided into parts as explained in Figure 1

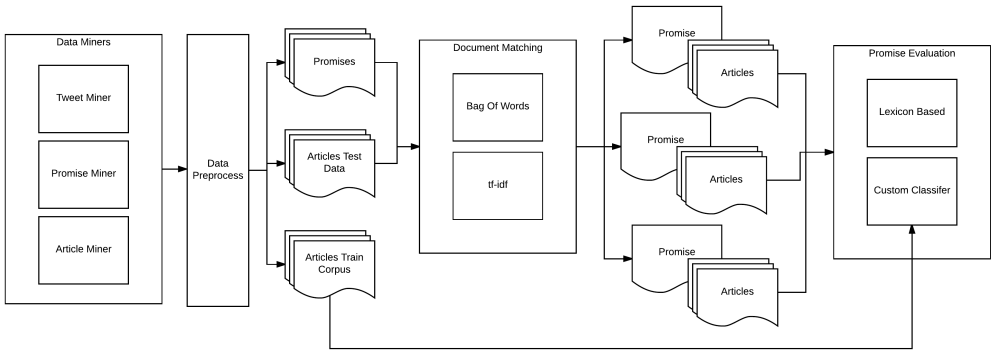

Figure 1: System Design

The main data aspects of our project are Promises and Articles/Text around those promises. We collect our data from multiple sources and feed it to the *Document Matching* module to get articles related to individual promises. Finally we have a *Promise Evaluation* phase where we evaluate the performance of each promise. Further we explain each component in detail.

4.1 Data Miners

We collected our data around *Donald Trump*. The promises were collected before *Donald Trump* held office,*(Jan 20 th, 2017)* while articles/text were collected post that date. Custom data miners were built to extract data from twitter and web for promises and articles. The following sources were used for gathering data.

1. Promises: TrumpTracker API[9]
2. Articles: Twitter[10], New York Times[2], Automated Google Search and Parsing[1], All News Dataset[15]

As there is a limitation with Twitter API, where a non paid user can fetch only past seven days of tweets, we decided to drop the idea of using twitter and focused only on news articles and Google search. All News Dataset was used as a training corpus to train a custom classifier in *Promise Evaluation* phase.

4.1.1 Promise Miner

We analyzed multiple resources to extract and collect promises. The best resource turned out to be the TrumpTracker API[9] which provides JSON based promises for a multitude of political candidates. Finally we were able to collect **174 promises** for *Donal Trump*. Further, we picked top three promises related to *Mexico Wall*, *Obamacare* and *Taxes* as a baseline. Any evaluation and results are based on these three promises.

4.1.2 Article Miner

The data was collected from multiple sources to get a more holistic view of the current status of promises made by candidates.

1. **New York Times**
   New York Times has newspaper articles available for the last century so it made sense to have it as one of our datasets. We restricted the articles based on certain parameters to only extract the relevant data.

   (a) Search Terms:

      i. Donald Trump
      ii. Mexico Wall
      iii. Obamacare

   (b) Date Range: 01/20/2017 to 11/30/2017

   We were able to collect a total of **624** articles for above mentioned parameters.

2. **Google Search**
   Top three promises collected in Section 4.1.1 were used as a query to extract the top 10 results from Google. These results were HTML stripped and were reduced to 10 text summaries related to the work done on each promise.

3. **All News Dataset**
   We used *All News Dataset*[15] to get a total of **142,473** articles from 15 different Media outlets. This allowed us to get an encompassing dataset which would include news from even polarized media sources.

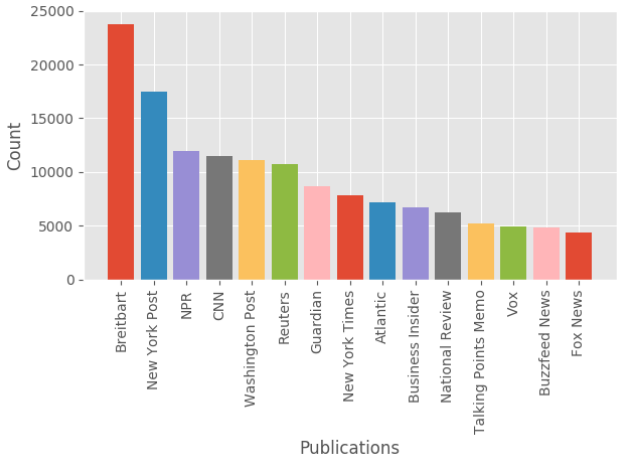

Figure 2: Training Data Publication Distribution

4.2 Data Preprocessing

4.2.1 Data Labeling

The data pulled from All News Dataset was labeled as 0 or 1 where 0 translates to articles related to broken promises and 1 translates to articles related to completed promises.

The distribution in Figure 3 clearly shows a huge difference between the articles with label 1 and those with label 0. This can be attributed the fact that this is a political dataset and it's way more likely to talk in a tone of achievement rather than non-achievement. The intuition behind the same is that if there is a promise which hasn't been fulfilled for the past 6 months then it's way less likely to generate a newspaper story as compared to a promise on which continuous work has been going on for the past 6 months [11].

We can observe from the distribution that there is no real linear separation between the data and thus no clusters can be observed. The data was evenly spread out and thus covered information consistently.

4.2.2 Data Sanity

- We restricted the provided dataset to include only the Promise title and the Promise description.

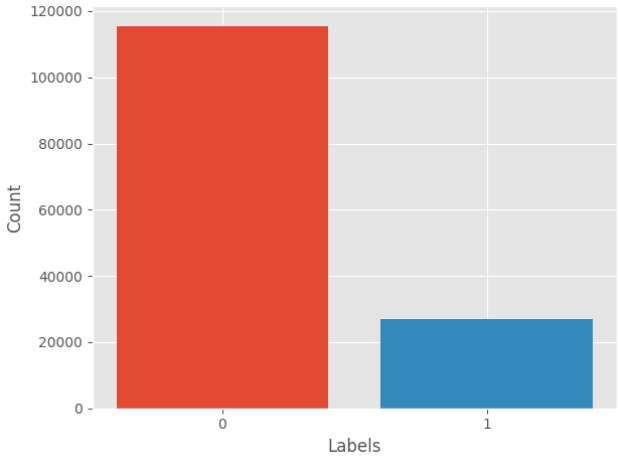

Figure 3: Training Data Label Distribution

Label	Article Count
0 (Broken)	115576
1 (Completed)	26897

Table 1: Distribution of Broken vs Completed Promise based Articles.

Table 2

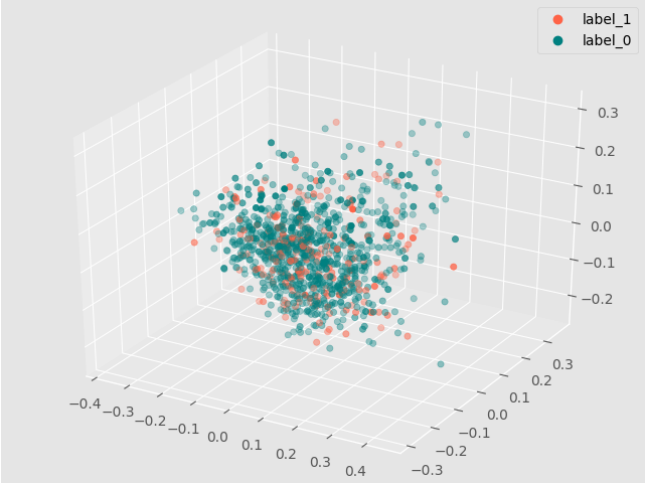

Figure 4: Training Data Vector Distribution

- A lot of Promises didn't have a description attached to it in that case we included the Promise Tags to help us understand the context of the promise.
- The information pulled from these varied sources were all converted to a JSON format to create a uniform dataset.
- The stop words, punctuations, special and ASCII characters were removed.

4.3 Data Representation

The data extracted was converted and stored as a JSON of promise arrays to allow fast retrieval and processing.

4.4 Methods/Models for Article Mapping

Since, our project was a progressive multi-step project rather than a linear project with a single aim so we experimented with multiple methods to solve different steps of our project.

After collecting the whole corpus of articles and promises our next step was to assign all articles to their relevant Promises so they could be used for sentiment analysis later on.

To do the above we tried to follow multiple ways

4.4.1 Bag of Words

The Naive Bayes Model was a good starting point for us as it helped us understand what features we really needed to get a good matching. Since we were matching keywords in promises with keywords in Articles at this stage so we missed out on a lot of potential matches. The promises being one-line sentences whereas the Articles being huge blob of texts weren't easy to match as a lot of the promises just didn't have enough content to be sufficiently matched with a corpus of articles.

1. We attempted to create a corpus of keywords for every article according to the frequency of every word in that article.
2. Then we tried to create a similar corpus of keywords for the collected promises based on the title and description of the promises. A lack of description for many promises made it extremely difficult to get a proper bag of words for them.
3. Many of the promises had zero associated articles even though we knew that there should have been more. We realized that this was because we were trying to match exact keywords in the promises which wasn't a good idea because we didn't have a big enough bag of words to get a proper result.

4.4.2 Tf-idf

The Tf-idf method treated both articles and promises as documents and then created vectors for them. The vectors were matched to generate a matrix and the top n closest articles for every promise were assigned to the promise. This gave a much better result then the Bag of Words approach. The improvement in the results were dependent on the way we processed and cleaned data too. The data was lemmanized this time, so that a direct word to word matching wouldn't be required to assign proper articles to the given promises.

The step by step procedure was:

1. We treated both the articles and the promises as documents.
2. We removed all the punctuation, the stop words and converted all the strings to lowercase.
3. We Tokenized all the words using the Tokenize method of the NLTK library[3].
4. After this we used the Tf-idf algorithm to get the Tf-idf numbers.
5. We decided to Lemmaentize the words to get all the inflections for the words.
6. Finally we used Cosine similarity to match all the articles to their relevant Promises.

4.5 Methods/Models for Promise Labeling

4.5.1 Naive Bayes

1. We used the Naive Bayes Classifier in our first attempt to label the Promises.
2. The Naive Bayes Classifier was pre-trained on a Movie Review Database.
3. The Naive Bayes Classifier implement was implementing using the formula:

$$\frac{P(label)P(\frac{f1}{label}) * ... * P(\frac{fn}{label})}{SUM[1](P(1) * P(\frac{f1}{1} * ... * P(\frac{fn}{1}))}$$

4. The Naive Bayes Classifier was used to understand the polarity of the articles and the range was divided as:
   - -1.0 <= Negative <= 0
   - 0 < Positive <= 1
5. Since Naive Bayes Classifier was predefined using the Movie Review Database therefore, the results weren't particularly favorable for our objective.

Methodology	Repeal and Replace Obamacare	Build a Wall
Article Text Pattern	Positive	Positive
Google Pattern	Positive	Negative
Article Text NB	Positive	Positive
Article Summary NB	Positive	Positive
Google NB	Positive	Positive
Article Summary Pattern	Positive	Negative
Actual	Negative	Positive

Table 3: Labeling of promise for Naive Bayes and Lexical Pattern

6. This highlighted the need to create our own Classifier using a Corpus of Newspaper Articles based on Politics and Candidate Promises.

As can be observed from the table there weren't any significant differences between running the models on either the text or the summary. This could refer to the fact that the models were too vague to even depend on the content and give meaningful information.

The sentiment polarity calculated using Naive Bayes on the Full Text vs Summary was almost similar. The difference could probably be attributed to the Bag of Words methodology used which didn't perform sufficiently well on the summary because the information had been cut down.

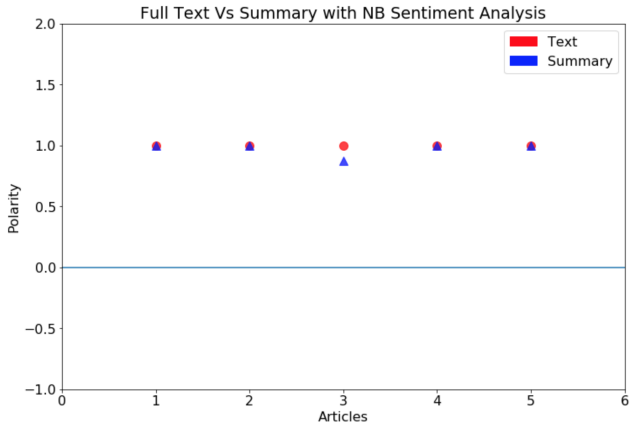

Figure 5: Naive Bayes on Full Text vs Text Summary

The difference of polarity using Pattern Sentiment Analysis were similarly close but there was less overlapping as compared to the Naive Bayes methodology. In one of the articles it actually switched from Positive to Negative polarity for the summary.

The Lexical Pattern and Naive Bayes were the first method utilized by us to get a general idea of how the project should progress. Expectedly it wasn't the best way to reach our objective but it was nevertheless a good baseline to build upon.

The difference in values between the text and summary highlighted the need to have more information datasets rather than small text datasets and we decided to increase our data collection methodologies to collect extensive data to allow proper evaluation.

4.5.2 Custom Trained Classifier

1. Using Naive Bayes highlighted the need to create our own classifier and train it on relevant data.
2. To train our classifier we used the Vader[13] (*Valence Aware Dictionary and sEntiment Reasoner*) which is a lexicon and rule-based sentiment analysis which is specially attuned to understanding sentiment on social media content.
3. We decided to use the above with a Bag of Words Methodology to make it work on Newspaper Articles.
4. The Bag of Words was based on Progressive Words to identify Newspaper articles which were focused on the successful completion of a Promise.
5. The Training was labeled as:

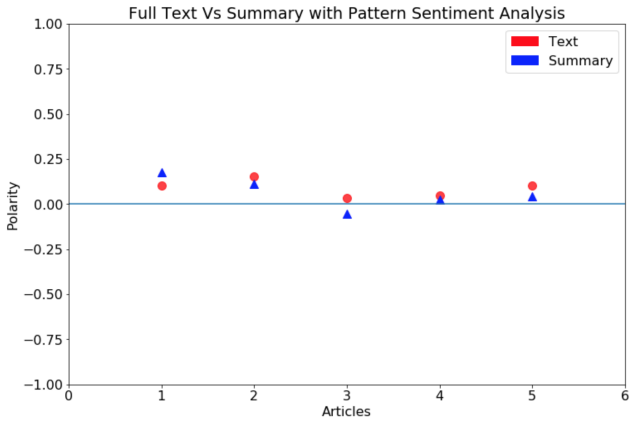

Figure 6: Lexical Pattern Matching on Full Text vs Text Summary

- Label 0: Broken or In Progress
- Label 1: Completed

For the progress co-efficient we devised the following equations:

$$progressSynonyms = \frac{synCount}{totalCount}$$

The progress synonyms included the words which were assigned as progressive words in the Bag of Words.

$$progressAntonyms = \frac{antCount}{totalCount}$$

The progress antonyms included the words which were the inverse of progressive words in the Bag of Words

$$progressCoefficient = tanh(pp) - tanh(pn)$$

The progress co-efficient was calculated by subtracting the inverse tan of progress antonyms from the inverse tan of progress synonyms.

The Progress Coefficient calculated above was used to calculate the label using the equation given below:

$$Label = Binary(Binary(progressCoefficient) * polarity)$$

After labeling all the articles the trained classifier was implemented using the following methodologies:

- Naive Bayes
- Random Forest
- State Vector Machine

Since the data had a skewed relationship between the 0 and 1 label therefore it was a bit more difficult to create an effective model for it. The value for 0 and 1 were divided as:

Label	Vector n-gram range	Mean	Standard Deviation
0	(1,1)	0.82	0.00
1	(1,2)	0.81	0.00

Table 4: Mean and Standard Deviation

After running them on the unbalanced training data we calculated the Precision, Recall and F-1 score to verify the results of the model. The values observed were as follows:

We ran the Naive Bayes on the actual test data which was extracted from the New York Times. We had a corpus of 624 articles in total with 24 of them predicted as label 1 and the rest predicted as label 2.

Label	Precision	Recall	F1 Score	Support
No Progress	0.83	0.99	0.90	28942
Progress	0.65	0.10	0.18	6677
Avg/Total	0.79	0.82	0.76	35619

Table 5: Precision, Recall and F1 for unbalanced Naive Bayes

The top keywords detected for Label 0 were:

$$[('trump', 317), ('mr', 227), ('york', 147), ('sign', 133), ('president', 132), ('health', 129),$$
$$('trumps', 110), ('care', 94), ('times', 79), ('republicans', 77)]$$

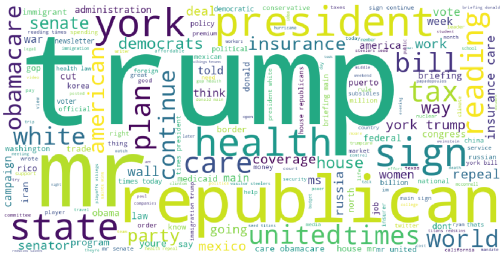

Figure 7: Word Cloud for Naive Bayes with actual text data

This shows that most of articles which talked about broken promises were more focused on the president and his biggest failings according to our dataset included the health related promises. As expected Republicans is one of the top keywords for label 0 with a much higher frequency than that for Label 1 even though the total number of articles are much lower. This clearly aligns with a generic intuition that broken promises have greater repercussion to the candidates party.

Whereas the top keywords for Label 1 were:

$$[('tax', 14), ('trump', 9), ('plan', 6), ('mr', 6), ('taxes', 6), ('federal', 5), ('trumps', 5),$$
$$('states', 4), ('united', 4), ('republican', 4)]$$

Figure 8: Word Cloud for Naive Bayes with actual text data

This clearly indicates the the presidents tax related plans were more likely to have been completed and this is true with actual factual data[9]. Naive Bayes was implemented with the Google Search Results to attempt to find better sources of information. We parsed the Google Search result pages to extract the search result and it's summary. But because of the concise content available with every search result, all summaries were identified to 0 label.

So there didn't seem to be enough content to return a 1 label even for promises which had already been achieved by President Donal Trump.

jan 26 2017 mexico president ha hit back donald trump decision build border wall repeating country would pay

This is an example of one of the Google Search results and it can be observed that the results are two short to extract meaningful content.

This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM).[6]

The results returned by implementing the SVM were:

Label	Precision	Recall	F1 Score	Support
No Progress	0.84	0.99	0.91	28788
Progress	0.88	0.18	0.29	6831
Avg/Total	0.84	0.84	0.79	35619

Table 6: Precision, Recall and F1 for Support Vector Machine

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).[5]

Label	Precision	Recall	F1 Score	Support
No Progress	0.82	0.99	0.90	28947
Progress	0.64	0.08	0.15	6672
Avg/Total	0.79	0.82	0.76	35619

Table 7: Precision, Recall and F1 for unbalanced Random Forest

After doing all the experiments listed above we had a much better idea of what we needed to do. We decided to label all the articles as 0 or 1. For some of them we used the pre-trained classifier and then we attempt to do the same with the classifier created by us. The results given by our classifier for 3 selected promises are:

Method	**Mexico Wall**			**Obamacare**			**Taxes**		
	Label 0	**Label 1**	**Outcome**	**Label 0**	**Label 1**	**Outcome**	**Label 0**	**Label 1**	**Outcome**
Pattern Lexicon on Article Full Text	0	10	Delievered	0	10	Delievered	0	10	Delievered
Pattern Lexicon on Article Summary	3	7	Delievered	3	7	Delievered	2	8	Delievered
Pattern Lexicon on Google Search Summary	5	5	Unkown	2	8	Delievered	2	8	Delievered
Naive Bayes Lexicon on Article Full Text	0	10	Delievered	0	10	Delievered	0	10	Delievered
Naive Bayes Lexicon on Article Summary	0	10	Delievered	1	9	Delievered	1	9	Delievered
Naive Bayes Lexicon on Google Search Sum	2	8	Delievered	5	5	Unkown	3	7	Delievered
Custom Classifier with Naive Bayes	10	0	Not Delievered	9	1	Not Delievered	4	6	Delievered
Actual			**Not Delievered**			**Not Delievered**			**Delievered**

Figure 9: Label and Prediction Results for selected promises

4.6 Evaluation Metrics

There were multiple evaluation metrics which we used to check the validity of the models:

- In case of the first experiment it was possible to manually validate the progress and that turned out to be the best way to do it.
- For the experiments following that we calculated the precision, recall and f-1 score to understand the quality of the models developed by us.
- To evaluate the accuracy of the labeling methodology we manually labeled articles and then matched the labels generated by the models with the hand generated values.

5 Future Scope

This project has been hindered by the technologies currently available, but with the fast progression in world of textual context understanding soon it should be possible to extract proper context from articles. Once completed an application like this can be used to give the citizens direct access to the promises made by their candidates and keep track of the results they delivered on.

Over time this information can be collated to review the performance of different political parties and might even show a clear divide between the promises made by the parties candidates and those fulfilled.

Some other ways to improve this project might be:

1. Mine information from popular social networks like Reddit, Facebook, Twitter etc. to get more extensive data about candidates promises.
2. Extend the implementation of this project to other former and upcoming head of states/countries.

References

[1] Google search. https://developers.google.com/webmaster-tools/search-console-api-original/v3/how-tos/search_analytics.

[2] New york times api. https://developer.nytimes.com/.

[3] Nltk library. http://www.nltk.org/.

[4] Politifact. http://www.politifact.com/.

[5] Scikit random forest.

[6] Scikit svm. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html.

[7] Textrazor. https://www.textrazor.com/.

[8] Trudeau metre. https://trudeaumetre.polimeter.org/.

[9] Trump tracker. https://trumptracker.github.io.

[10] Twitter api. https://developer.twitter.com/en/docs.

[11] Us government tracker. https://www.govtrack.us/congress/bills/statistics.

[12] Usa-mexico border wall. https://web.archive.org/web/20170210172237/http://www.reuters.com/article/us-usa-trump-immigration-idUSKBN1591HP.

[13] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[14] Kevin Hsin-Yih Lin, Changhua Yang, and Hsin-Hsi Chen. What emotions do news articles trigger in their readers? In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 733–734, New York, NY, USA, 2007. ACM.

[15] Andrew Thompson. Newspaper article sources. https://www.kaggle.com/snapcrack/all-the-news.

[16] Lori Young and Stuart Soroka. Affective news: The automated coding of sentiment in political texts. *Political Communication*, 29(2):205–231, 2012.

[17] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.