Political Promise Evaluation (PPE)

Bhanu Jain, Rohit Begani

December 12, 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 extremely helpful for people to 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 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 posts[17]. Additionally, there are a few online softwares, which attempt to find meaning from textual data like TextRazor[7]. Although such softwares extract the meaning from textual data, but are more focused on context matching in a generic form.
There are multiple manually managed websites, which are dedicated towards listing all the promises made by prominent Political Candidates and track the progress of these candidates with respect to their promises. 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, but none of it tries to parse the newspaper content to actually understand the context in which political promises are being talked about.[13][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](image)

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.

4.1 Data Miners

We collected our data around Donald Trump. The promises were collected before Donald Trump held office,(Jan 20th, 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]

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 TrumpTracker API[9] which provides JSON based promises for a multitude of political candidates. Finally we were able to collect 174 promises for Donald 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 political articles from 15 different media outlets. This allowed us to get an encompassing dataset which would include news from even polarized media sources. The data is unlabeled.

![Figure 2: Training Data Publication Distribution](image)

4.2 Data Preprocessing

4.2.1 Data Labeling

The data pulled from All News Dataset was labeled as 0 or 1 where,

- Label 0 refers to Articles reflecting limited or no progress.
- Label 1 refers to Articles reflecting completion or good amount of progress.

The label is defined as the following function,

\[
Label = Binary(Binary(ProgressCoefficient) \times SentimentPolarity)
\]

\[
ProgressCoefficient, PC = \tanh(PS) - \tanh(PA)
\]

\[
ProgressSynonyms, PS = \frac{ProgressiveSysnonymCount}{TotalTokenCount}
\]
We used a baseline of 21 lexicons related to progress. These base lexicons were expanded to synonym set and antonym set using WordNet. Finally, a total of 149 Progressive Synonyms and 26 Progressive antonyms were generated. These sets were used to compute $PS$ (eq 3) and $PA$ (eq 4). ProgressiveCoefficient (eq 2) is the difference between tangent of $PS$ and $PA$. \( \tanh \) is a monotonically hyperbolic function, bounded between -1 and 1. Finally, ProgressiveCoefficient was coupled with Sentiment Polarity obtained by Vader Sentiment Analysis to achieve the label. Vader\cite{13} (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis which is specially attuned to understanding sentiment on social media content.

![Figure 3: Training Data Label Distribution](image)

Post labeling, we got 115576 articles as Label-0 and 26897 article as Label-1, Figure 3 shows the distribution of train data after labeling. The huge difference in classes can be attributed to the fact that this is a political dataset and it’s way more likely to talk in a tone of non-achievement rather than 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 is more likely to generate more newspaper stories as compared to a promise which is completed.\cite{11}.

![Figure 4: Training Data Vector Distribution](image)

Figure 4 shows a 3-dimensional view of a subset of a training data. Each article was vectorized and fed to a 3 Component PCA to reduce the dimensions. We can observe that there is no real linear separation between the labels and no evident clusters. The uneven distribution of data might be due to custom labeling. Also, the class imbalance is very evident in the distribution. We chose not to normalize our classes as we had limited data and down-sampling might reduce the accuracy of our models. Also, with the same uneven distribution we are able to achieve good results.
4.2.2 Data Sanity

For each article and promise we performed a series of data sanitation and tokenization rules to obtain clean tokens.

- We removed stop words, punctuations, special and ASCII characters.
- We decided to Lemmatize the words to get all the inflections for the words.
- Finally, we tokenized text using NLTK Regex Tokenizer [3].

4.3 Document Matching

After collecting the whole corpus of articles and promises our next step was to assign Articles to their relevant Promises. To do so, we implemented the following procedures.

4.3.1 Bag of Words

We started with term frequency matching to assign articles w.r.t promises.

1. We created a vocabulary of keywords for every article and promise, sorted by the frequency.
2. Documents with max. number of similar keywords were paired together.

Matching with term frequency did not gave good results as there were a lot of words with high frequency count(such as Trump) which appeared in almost every promise and article. These very common and high frequent words resulted in overmatching. This also proves the fact that relevance does not increase proportionally with the term frequency.

On the other hand, few 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 promise vocabulary to get a proper match.

Bag of words which is frequency based document matching approach was a good starting point for us as it helped us understand what features were really needed to get a good match.

4.3.2 Tf-idf

Tf-idf is the most common document matching approach which tries to find the importance of a term in a document. This solved the problem of over high frequent terms in a corpus, by discounting the frequency of terms in the entire corpus.

To apply Tf-idf, we treated both articles and promises as single corpus and created Tf-idf vectors. Each vector was then ranked with Cosine Similarity, to generate top n similar vectors.

With Tf-idf we were able to identify articles for each promise. We picked top 10 most similar articles for 3 baseline promises (described in section 4.1.1).

4.4 Promise Evaluation

Promise Evaluation is the phase where we extract progress on a particular promise from all of it’s associated articles. We started with pure sentiment based approaches with an assumption that sentiment of an article is directly proportional to the progress in that article. We soon realized that sentiment is driven by opinion and subjectivity of the underlying text. From this baseline we trained a classifier to include objectivity along with the opinion.

4.4.1 Lexicon Based

We used pattern.en module from Computation Linguistic And Psycholinguistic Research Center and extracted polarity for each article. The pattern.en module bundles a lexicon of adjectives (e.g., good, bad, amazing, irritating, etc.) that occur frequently in product reviews, annotated with scores for sentiment polarity and subjectivity. It is based on the adjectives it contains, where polarity is a value between -1.0 and +1.0 and subjectivity between 0.0 and 1.0.

4.4.2 Pre Trained Naive Bayes

This was a pre-trained sentiment analysis classifier, trained by NTLK on movie reviews. This classifier gives a polarity for the given text between -1 and 1. Both Lexicon based and pre-trained classifier performed poorly, which led to building of our own classifier.
4.4.3 Custom Trained Classifier

With the idea of capturing objectivity along with subjective sentiment we trained our own classifier on data from the same domain. We started with labeling All News Dataset (section 4.2.1) using the Vader\[13\] (Valence Aware Dictionary and Sentiment Reasoner) which is a lexicon and rule-based sentiment analysis, specially attuned to understanding sentiment on social media content. Using Vader and custom progressive rules (section 4.2.1) we got articles labeled as 0 (Articles reflecting limited or no progress) and 1 (Articles reflecting completion or good amount of progress)

With labeled data, a classifier was trained using the following methodologies:

- Naive Bayes
- Random Forest
- Support Vector Machine

Naive Bayes was an obvious choice as it is significantly fast and it captures lexicon context. SVM works well with data represented as dense or sparse arrays of floating point values for the features.\[6\] 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.\[5\] With training on a 70-30 split, we achieved the following accuracy.

<table>
<thead>
<tr>
<th>Label</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Progress</td>
<td>0.83</td>
<td>0.99</td>
<td>0.90</td>
<td>28942</td>
</tr>
<tr>
<td>Progress</td>
<td>0.65</td>
<td>0.10</td>
<td>0.18</td>
<td>6677</td>
</tr>
<tr>
<td>Avg/Total</td>
<td>0.79</td>
<td>0.82</td>
<td>0.76</td>
<td>35619</td>
</tr>
</tbody>
</table>

Table 1: Precision, Recall and F1 for unbalanced Naive Bayes

Table 1 shows the performance on Naive Bayes. The classifier being a high precision and low recall can be attributed to the fact that classes in our training data are unbalanced.

<table>
<thead>
<tr>
<th>Label</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Progress</td>
<td>0.84</td>
<td>0.99</td>
<td>0.91</td>
<td>28788</td>
</tr>
<tr>
<td>Progress</td>
<td>0.88</td>
<td>0.18</td>
<td>0.29</td>
<td>6831</td>
</tr>
<tr>
<td>Avg/Total</td>
<td>0.84</td>
<td>0.84</td>
<td>0.79</td>
<td>35619</td>
</tr>
</tbody>
</table>

Table 2: Precision, Recall and F1 for Support Vector Machine

Table 2 and Table 3 shows the performance of SVM and Random forest. With Random Forest we were expecting better results for our unbalanced data, but it performed exactly the same as Naive Bayes. Best f-1 score of 0.79 was achieved by SVM but it was relatively slow. As there is no significant performance improvement we chose Nave Bayes for our further experiments.

4.5 Evaluation Metrics

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

- For Progress Evaluation of Promises, after labeling each article with 0 or 1, we assigned a final label Delivered, Not Delivered, Unknown to each promise. These were based on,

\[
\text{Promise Label }= \text{Delivered, if } \text{count(Article with Label-1)} > \text{count(Article with Label-0)}
\]

\[
\text{Promise Label }= \text{Not Delivered, if } \text{count(Article with Label-1)} < \text{count(Article with Label-0)}
\]

\[
\text{Promise Label }= \text{Unknown, if } \text{count(Article with Label-1)} = \text{count(Article with Label-0)}
\]
The Promise labels were then compared to human annotated labels to score the accuracy.

- For Classifiers, we calculated the precision, recall and f-1 score to understand the quality of the models developed by us.

5 Experiments and Results

We started experimenting matched articles with Lexicon (section 4.4.1) and Pre Trained Classifiers (section 4.4.2). Both the approaches performed poorly as they were very biased towards sentiment and sentiment was evaluated over subjectivity of the text. Figure 5 shows both lexicon and pre-trained approaches labeled majority of the articles as 1 and thus all the three selected promised were marked as delivered. This attributed to 66% accuracy over three baseline promises.

![Figure 5: Label and Prediction Results for selected promises](image)

We also experimented with using summary of an article rather than full text. As can be observed from the Figure 5 that 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 (Figure 6) 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 6: Naive Bayes on Full Text vs Text Summary](image)

The difference of polarity using Pattern Sentiment Analysis (Figure 7) 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 above comparisons with text and summary highlighted that, with summary as we loose information, the sentiment is also lost. Thus we considered full text in our further experiments.

In our last experiment we used Custom Trained Classifier (section 4.4.3) with the idea of capturing objectivity along with opinion. We ran the Naive Bayes on the actual test data which was extracted from the New York Times. This classifier gave us 100% accuracy (figure 5) over three baseline promises (section 4.2.1).

We had a corpus of 624 articles in total with 24 of them predicted as Label-0 and the rest predicted as Label-1. Figure 9 shows that most of the articles which talked about broken promises were more focused on
the president and his biggest failings, including 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 repercussions to the candidates party.

Top keywords for Label-1 were tax and trump. Figure 9 indicates that the president’s tax related plans were more likely to have been completed and this is true with actual factual data[9].

6 Conclusion and Future Scope

In conclusion our project provides the user an automated way to check the status of the promises made by their political candidates. It uses text parsing and context understanding to differentiate between political articles and this context is used to assign a status to the promises.

This project has a wide growth scope and has a number of future applications. Reading contextual data from newspaper articles and other online sources would allow a user to hold political candidates accountable. 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 etc. to get more extensive data about candidates promises and public sentiment regarding these promises.

2. Extend the implementation of this project to other former and upcoming head of states/countries.

3. Implementing this prototype for other candidates would help to understand the different requirements for different geographical locations and cultural differences.

References


