Can you trust your politician? A NLP approach to politics.

Edgar Montano and Jeremie Roche

[1]Natural Language Processing Spring 2016

In this paper we will use natural language processing in order to extract information of political candidates and see if the main topics they push during there campaign is actually acted on. We will be using information extraction techniques to extract key phrases from their political speeches as well as legislative documents.

Introduction

Our main approach was to select specific candidates in which we would analyze the political speech. Among some of the candidates we choose, we decided to use recent candidates due to relevance. However we were met with several obstacles when approaching this.

The primary obstacle presented by analyzing both legislative documents and political speech was the vast difference between the two. In fact, political speech has been the topic of many NLP research, partly due to the fact that politics have their own unique rhetoric. In addition to this, campaign speeches actually fall under the category of persuasive speech, which we actually used to our advantage. Similarly, legislative documents are offered in XML format, which means items are actual DOM elements, therefore we can traverse a XML file in a tree like structure, this provided us with an advantage on identifying phrases.

1 Research

In order to analyze political speeches we needed additional research on how to approach this. Since political speeches and transcripts of these speeches are slightly different then the average text, primarily because spoken language varies different then test data, we needed an approach to analyze political speeches and identify key phrases

One approach derived from previous research on tagged political speeches. The research paper *CORPS: A Corpus of Tagged Political Speeches for Persuasive Communication Processing by Marco Guerini, Carlo Strapparava, and Oliveiero Stock, provided us with an approach to analyzing campaign speeches. This research paper provided us with a key approach on how to analyze political speeches. Since campaign speeches have indicators when the audience gives an applause, we can use applause as a breakpoint on a key note. Usually a key phrase or point they are trying to convey is stated right before an applause. This means we can narrow our search in order to find key phrases. However this technique is not entirely accurate. For instance, an applause may occur during the beginning of the speech. This usually involves the candidate going up on stage, and saying thanks. This means a majority of the speeches we analyze will have "thank you" as a key phrase. However this obstacle was easy to handle, we just simply ignore these phrases.*

In addition we needed an approach to analyze legislative documents. Unfortunately natural language and written language, especially legislative text, have vastly different approaches to them. The premise of our approach on analyzing legislative relies on previous research conducted by Ron Van Gog and Tom M. Van Engers *Modeling Legislation Using Natural Language Processing. This particular research paper describes the possibilities of translating legislative documents which is written into formal language, into an easier format to parse and identify key elements. This paper helped us identify a key aspect of exploiting the XML file format, and*

clever DOM tree manipulation, to increase the performance of our task. We utilized the XML section headers to give a clear and concise estimate on what might be a key phrase from a legislative document.

2 Campaign Speech Analysis

This portion of the project was probably the more difficult one. It was the least automated aspect of the project. We had to deploy various methods in order to properly analyze speech. In addition, we encountered issues with the data. It turns out that most campaign speech text made available are done automatically using various tools such as Youtube's auto-caption functionality. While this is a nice feature to supplement the actual voice, simply analyzing auto-generated documents proved to be a painstaking task since it provided too many inaccuracies. This means we had to limit our actual test data, to documents we can verify were authentic transcriptions of the speeches, rather then auto-generated text. We did this by simply checking if the grammar of the speech were noticeably ambiguous at points. For example, auto generated transcriptions often give incomplete sentences with multiple instances of a word. "The the" occurring twice when a person stutters. Manual inspect of the documents sifted through these instances. However, there is one instance of a document I would like to discuss, since even manual auto notation would have proved wrong.

The particular campaign speech that was most difficult to analyze and add to our test group was none other then Donald Trumps's campaign speech. This was Trump's Iowa Freedom Summit speech (included in documents). Our phrase extraction program gave the following phrases a frequent phrases occurring in the document: "Ba, pa, bing". After careful analysis, (watching the entire video corresponding to this speech) we confirmed that this was in fact a phrase he stated, and he also stated multiple times. This informal language use provided us with yet another obstacle in campaign speech analysis.

The actual approach to phrase extraction for campaign speeches was achieved using two-phase-phrase-extraction. Essentially we ran our text analyzer which found the most frequent noun phrases in the speech. The most common phrase across documents was usually persuasive words or words employed to develop certain emotions from the crowd. In addition, phrases such as ”thank you” were highly common after applause. Our naive phrase extraction first extracting all of these phrases, and hashed them into a dictionary, and counted the frequencies. We then cached these results for later reference. The second phrase of extraction was used which involved similar techniques deployed in the *CORPS research paper.*

The second phrase extraction technique involved exploiting the fact that we were analyzing a transcribed document, meaning: ”applause”, ”cheers”, ”laughter”, and occasional ”boos” were embedded into the document. These served as signal points on which the speaker was conveying a key point in there speech, usually persuasive natural language techniques were employed to develop crowd sentiment towards a certain subject. Our second extraction phrase script located these points in the document, and created a buffer of the previous sentences that were stated. It's at this point we can choose to manually extract the phrases that invoked these crowd reactions OR optionally we could use our first phrase extraction tool to identify and automate this task. We choose manual extraction since it would actually give us a more accurate result then the possibility of using the first phrase extraction.

It is at this point we identify main topics in the campaign speeches. The point of doing this phrase extraction is to identify the key points of a speech. At this point, we can cross reference multiple speeches, and find the most common phrases across all speeches, to determine the main sentiment of a specific candidate. Our task is to identify this, and search through legislative documents to see if they either backed or sponsored legislative bills, laws, or documents in which reflected on these key points they spoke about in their speeches. In short, can we trust our politician's speech and see if they hold true to the promises they make. For example,

will Cruz have pursued immigration laws or anti-terrorist laws. (Note: our data is somewhat outdated, since now the current candidates have changed, however we still used the data as it is still relevant).

Legislative Document Analysis

In order to understand the position of particular politician, we examined the legislative bills that were introduced to either the House of Representative or the Senate, depending of course on which chamber they resided in at Congress. Congress.gov has a fantastic database of files, all sorted in various formats, such as plain text files (.txt), as images (.pdf), and as XML files (.xml). Our script was written in python, thus we though parsing the XML files would be the easiest and most efficient file type to extract give that there are many python libraries (in our case pickle and xmltodict) that can handle parsing xml files in a relatively neat and organized manner. We decided to include both libraries. The xmltodict was best used for reading off the raw instance data from a direct url request (urllib2.urlopen). Pickle is used as a method to cache information, and we plan on using this for the future to avoid the time to extract the data from a url request. In either parsing method, extracting the information from the bill can be a challenge, because sorting through all the trees from the xmltodict library requires navigating through miscellaneous fields such as #text, metadata, text, legis-body, and action-desc in order to extract important information from the text, such sponsors/cosponsors, dates, and section titles. While difficult to navigate, there are conventions to how bills are formatted, thus there is a scriptable way to consistently extract the information from the tree resulting from the xmltodict library that extracted the .xml file. This makes it a lot easier to parse the .txt file since while the bill conventions are still in tact, in order to consistently get the relevant information about a bill, on must keep track of relative locations and the words that were listed rather than navigating through a tree. However, right now we are only able to consistently extract the Section title

of the bill given that there is a host of formatting issues to deal with on the actual content of the bill. For the time being, we are OK with that as there is lots of interesting information and phrases that we can extract from the section title. Down the road we plan on extraction the more detailed contents of the bill, potentially through the .txt file, since a problem with the xmltodict tree is that it embeds all the formatting content on the collective writing, while the .txt only includes the rawest words. So in order to optimize our results, we had to implement two-phase-extraction for this portion as well. The first phase was to extract XML contents and section, and generate phrases from this, second phase was to download alternative text file and use phrase extraction on this portion.

We applied this methodology to a set of House of Representatives and Senate Bills Introduced by a Congressmen. After the parsing, we started to parse the sections titles into noun phrases using NLTK, in order to extract phrases that we feel unravels the core principles of the bill. This practice excludes a lot of phrases along the way and down the road we down like to include more verbs into the mix, but we feel that in our preliminary findings we are able to extract the necessary information for our data. Once we extracted the phrases, we assigned grades to the bill relative to how a bill viewed a certain issue. We did that through having three categories of tokens (Base, Positive, and Negative). Positive words were words that signaled that they were in support of a certain issue. Negative words were words that signaled that they had a negative opinion of a certain issue. Base words were words that amplified the direction in which the bill had a positive or negative opinion of an issue. Base words would also be added whenever the issue name was in the text. For instance, the Expand Enforcement of Immigration will have Expand and Immigration both increase the base scores, while Enforcement added a negative point. Since there were more negative points than positive points, the base scores counted negatively to the bills grade of immigration. If it was instead Expand Support of Immigration, everything would stay the same except that Support was a positive word, which means

that the base scores adds to the positive score. In order to prevent a score on bills not relevant to an issue, we added a requirement that there should be at least one mention of the issue keyword in a section.

Once we assigned the grades to a bills, we aggregated the scores of all the bills that a congressmen sponsored/cosponsored and assigned that grade to the congressmen. After that we categorized them into one of three categories. They were either Positive (overall positive opinion of a specific issue), Negative, (overall negative opinion of a specific issue), or Neutral (Aggregate score was to 0 or was not part of any bill that mentioned a specified issue). We evaluated this set with a manual categorization, which is based on their total history on an issue, rather than a measurement on a specified collection of bills introduced before the congress floor, on 10 congressmen of interest on two issue, Immigration and Terrorism.

3 Evaluation

Some things that we noticed from our evaluation was that our program was very good at placing congressmen into the Positive Category. The ones that were placed in the Neutral category generally contained congressmen that were usually either positive or negative. As we investigated further we found out that this was due to the fact that the ones that we categorized as neutral generally expressed their opinion on an issue through commentary and votes on an issue rather than personally introduce a bill before congress. There was also not as many congressmen placed in the Negative category, since even if they expressed passionate negative responses to a certain issue, usually the most powerful way to express that is through votes rather than bring forth original legislation.

4 P as in Performance based caching through Pickles

To maximize our results we used pickles! And no, not the food. We used a caching library,

that allowed us to cache array,vector, and dictionary values. Instead of iterating through 2000+ documents each time we wanted to extract phrases and cross-reference them, we had our scripts do this task once, and then store the results in a .pickle file. This pickle file could later be loaded with the extracted phrases so we can cross reference for later use with a significant increase in speedup. The actual task of sifting through documents rely on our document downloader which makes an increasing amount of HTTP requests to congress.gov, from there, it downloads scripts, the issue at hand was obtaining the results from an extensive list of legislative documents, this is solved through caching with pickles, allowing us to develop our own library of common phrases. The use of pickles actually increased the performance of cross-referencing by more than double.

5 Our System and the future of politics

Our system was met with various challenges, but we were able to efficiently approach each task with a rather dynamic approach. So what does this mean for politics? Well if provided with enough data, assuming transcripts become more widely available, we can develop a machine learning algorithm to actually detect, annotate, and extract information from speeches. It could then use various ML techniques. Or maybe an even practical and less involved possibility is using our phrases extracted, to make a sentence generator. We developed our system so it could be modular and adapt to different scenarios. Included in our git repository is a module.config file, that allows you to run plugin files that you can use to develop in addition to our system. For example, you can have the config file first execute a phrase extraction operation, and use pickle to dump these files and create a Donald Trump-inspired sentence generator or language. Included in this project is an example of such functionality.

I think the highlight and pride of how versatile our system is, is through the demonstration of TrumpSpeak. A python script, Donald Trump inspired language. The language is limited with

the following restrictions: 1. Every sentence MUST begin with ”I Donald Trump,” Followed by a verb, and a noun phrase we extracted from his speeches.
yielding

References and Notes

1. Marco Guerini, Carlo Strapparava, Oliviero Stock, *CORPS: A Corpus of Tagged Political Speeches for Persuasive Communication Processing* (Journal of Information Technology and Politics, Vol.5(1) 2008).
2. Ron Van Gog, Tom Van Engers, *Modeling Legislation Using Natural Language Processing*,