Sentiment Classification and Entity Linking for Determining Political Sentiments in Online Discussion Forums

John Klingelhofer
Rensselaer Polytechnic Institute
jkklingelhofer@gmail.com

Nathan Potolsky
Rensselaer Polytechnic Institute
nathanpotolsky@gmail.com

Abstract

Recent years have seen an increased level of interaction between online communities and real-world politics due to the rise of social networks, and with it enthusiasm for the prospect of political speech analysis though Natural Language Processing. This paper proposes a model for political sentiment classification utilizing a combination of the following strategies: a Long Short-Term Memory (LSTM) model for sentiment classification, trained on existing models for tonally similar (though non-political) online speech, entity linking using the RPI Joint Information Extraction System, and political affiliation extraction for a parsed entity using Wikipedia data. We will demonstrate how the results generated from this model can be used to highlight the range of political speech within different online communities, as well as how the output of this model can be used in the generation of a corpus of political speech surpassing current options in this emerging sector of computational linguistics.

1 Introduction

The inexorable relationship between technology and democratic politics cannot be overstated, with the later meant to act as the embodiment of public voices and ideas and the former the mode through which those ideas are distributed. From the printing press to television, each leap forward in technology has an immediate impact on the political landscape, but no technology has altered the landscape quite as radically as the internet and the social networks that exist on it. As the amount of data on a platform increases, users of this platform must curate an increasingly small selection of this data for personal consumption. Combine this axiom with the machine-learning backing of most social networks and search engines to show you content you're more likely to approve of, and we see online communities occupying a much more narrow ideological spectrum than would have been encountered within their real-life counterparts in a pre-internet age.

The social media platform Reddit offers an ideal space in which to observe the behaviours of and ideological discourse of online communities, as here communities are organized into distinct subsections called "subreddits", far more public than Facebook communities and cohesive that Twitter communities. In addition, most estimates suggest that in online communities only ten percent of the population actively comments or contributes content. Reddit's voting feature, in which even the passive participants can cast votes to make content more or less visible, enables this typically silent majority to still have an active role in the visible content, thus making analyses on the visible content more representative of the overall community than on comparable social networks.

In order to analyze the content of these communities, we propose a union of Named Entity Recognition (NER) and sentiment analysis in order to determine the political sentiment of a given comment based upon the negative or favorable language surrounding a named entity, as well as the political affiliation of that entity which is parsed through that entity's Wikipedia page.

2 Methods

2.1 Overview

The disparate elements of this project join together in a web application connecting several different technologies through which users can explore the

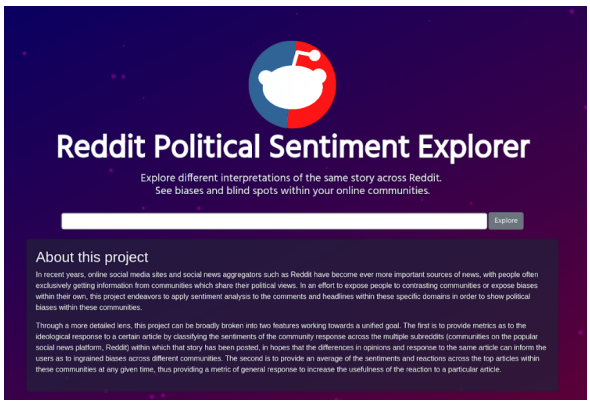

Figure 1: Title page for the web application

differing interpretations of Reddit articles across different communities. Here, users submit the link to an news article on the page seen in figure 1.

The functions in our Reddit Toolkit will then create a list of dictionaries with keys linking to the comments, score, and other relevant details for each subreddit, as well as all of the text and scores of comments in the thread. These comments are then passed to the Entity toolkit, which returns the political parties of all of the entities mentioned in the comment as -1 for Democrat-leaning, 1 for Republican-leaning, or 0 for no affiliation detected. The sentiment toolkit then attributes a positive, negative, or rarely neutral sentiment to the comment as -1, 1 or 0. The political value is then multiplied by the sentiment, and this is multiplied by the comment's score for an overall value. This means that a negative sentiment towards a Republican would have the same impact on the overall score as praise for a Democrat. These values are then totaled for a single value for the submission, with examples provided showing how this score was generated. In figure 2 we see a brief, simplified visual representation of connections between the various elements.

2.2 Sentiment Classification

For sentiment classification, we used an LSTM within the TFLearn (a module which serves wrapper for TensorFlow which decreases the verbosity of creating networks) trained on a robust corpus of IMDB movie reviews for sentiment. There are other sentiment corpora available, such as the very popular Cornell movie review database and some (relatively small) databases with tagged sentiment from Twitter posts. The IMDB dataset proved to be superior to these two options due to the demo-

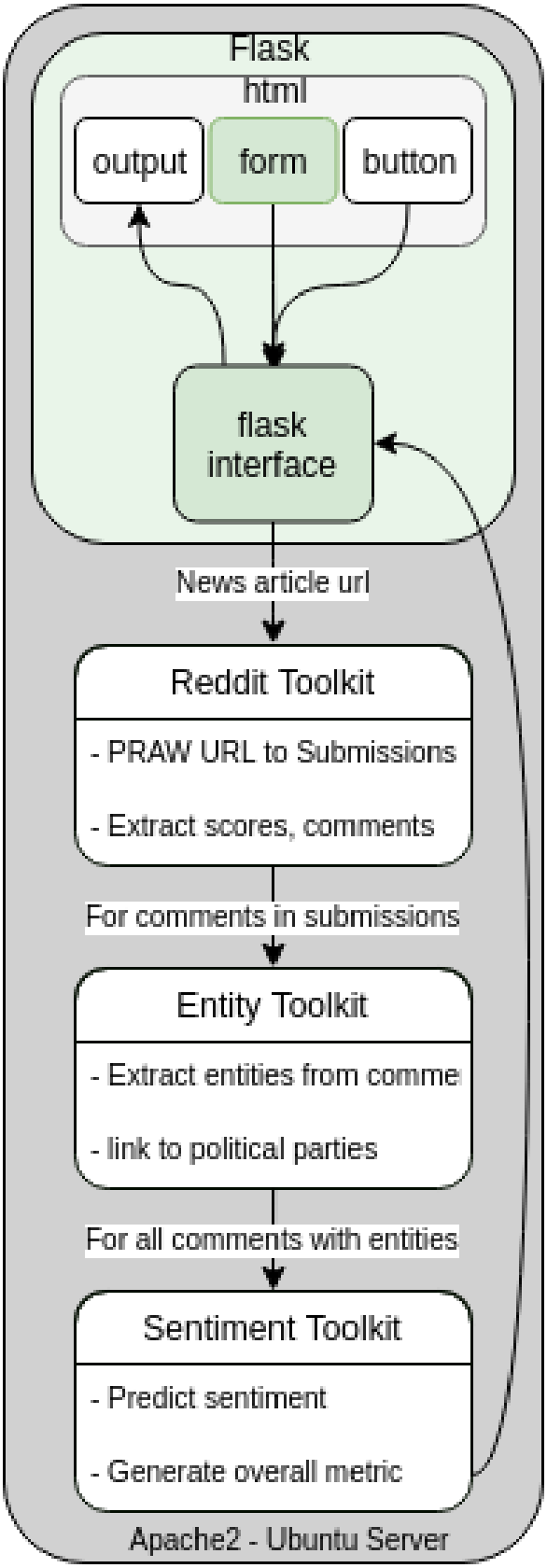

Figure 2: Brief overview of the relationships between the the technologies here applied.

graphic overlap and shared slang between IMDB and Reddit, and the Twitter corpus's brevity lead to decreased accuracy in some of the longer-form Reddit comments which, like IMDB, can have migrating sentiment over time.

All data submitted was translated into a vector, based off of the word-to-id dictionary which came with the IMDB dataset. For our purposes, this dictionary was limited to the top 10,000 most common words. Unfortunately, this IMDB dataset is labelled only as positive or negative. As a result, our model does not have any intrinsic ability to recognize ambiguity. As our LSTM returns the probability that a given string of text is positive or negative, we approximate an ambiguous classification by returning 0 if the absolute value of the probability of a positive classification minus the probability of a negative classification is less than 0.2, however this is far from an ideal method. In testing properly spelled words (excluding names or non-English phrases) fell outside of this dictionary. Once properly vectorized and formatted, the data was fed into the fully-connected LSTM, trained with a learning rate of .0001, categorical cross-entropy with the Adam optimization algorithm for regression. From other papers on this subject, this seems to be the dominant organization for LSTMs.

2.3 Entity Linking and Political Affiliation

For linking entities, we utilized the RPI Joint Information Extraction System, which was developed by Qi Li and Heng Ji. We compiled our comment data into one large string, with string delimiters at the beginning and end of each comment respectively so the entity tags may be easily associated with the comment they had originated from. The comments themselves were translated from Unicode to ASCII with Unidecode, to be compliant with the requirements of the API as some Unicode characters resulted in process errors. We then sent that as the data payload in a POST request to the API with Python's HTTP Requests library. Using Beautiful Soup, we parsed out the div HTML tags, which contain the entity tag data and used regular expressions to parse out the entity name and type from the div tag. For our project, we were only interested in named, people entities, not location, geo-political, or organization entities. Had we employed co-reference resolution, we could have made use of pronouns such as he, she, they, and so on to improve our accuracy.

Once we had extracted the tags for the named person entities (NPEs), we used the Wikipedia Python module to get back a list of results in order of salience. This list includes near matches, meaning that frequently incomplete use of NPE's such as politicians last names still return the politician in the list of results. From this list of page titles which may be correct, we use the Wikipedia API to extract the Wikidata ID. Then, through a Wikidata Python module we extract the political party of this entity. Unfortunately, due to limitations in both the keys returned by the direct access of the Wikipedia API as well as limitations within the relatively scant Wikipedia-wrapping Python module, this three-step approach is required and takes about a second or two to return the political affiliation, or lack thereof, from a person's name gathered in the previous step. In order to limit the computational and time demands of these repeated API calls, all entities, as well as the page title and party they correspond to, are recorded in a dictionary so that any delay only occurs on the initial look-up. This dictionary is saved each time a new entity is discovered, so as to preserve the progress made.

2.4 Previous Approaches

One of the earlier approaches we had tried for sentiment classification was to utilize a sentiment lexicon by Stanford's NLP group. It contained the top four to five thousand most commonly used words in the top two hundred and fifty communities on Reddit, discarding filler words like pronouns and conjunctions, and their computed sentiment values. We amalgamated the lexicons for communities we were more interested into one large lexicon, averaging the sentiment for words across different communities. When looking at a comment, we'd check the contents to see if any of the words matched our sentiment dictionary and added the corresponding sentiment for that word to the overall sentiment of that sentence or comment. We were going to have it coordinate with a Part Of Speech (POS) tagger to see what the sentiment of the executor in the sentence was, which would inform us of what political leaning was being expressed.

The RPI Joint Information Extraction System was helpful but also came with some limitations. Sometimes it would miss entities that it had picked up in other similar contexts or mis-tag them as the

wrong type of entity. This was more apparent for people whose names had other meanings. Additionally, calls to the API took a bit of time. To get the entity tags for one comment took roughly five seconds, which, to do individually, would take the order of minutes to process a hundred comments. We opted to roll up the comments together as making one large request proved to be quicker than many small ones. Making use of that strategy, one hundred comments takes about twenty seconds to process. A significant improvement, no doubt, but paled in comparison to the locally run named-entity recognition schemes which employed a naive Bayesian network for isolating named entities from chunked data. While the chunking itself worked well, it's results were diminished by the inaccurate perceptron POS-tagger naive to NLTK which was used in generating the features utilized by this network for categorization.

3 Testing

Due to the relative uniqueness of the subject being pursued, there was limited data available for testing. As a result, testing was broken into three stages for the disparate elements of the project. First and most importantly, we tested the generated model against a test set of 5000 IMDB movie reviews in order to ensure that the general sentiments were being parsed correctly, as without this keystone element future results would be destined to fail. On this set, the LSTM for sentiment classification achieved accuracy of 90% according to the accuracy output through the training-visualization tool TensorBoard, seen in 3. (Though the end of this graph is slightly higher than this, additional tests on the trained method revealed 90% to be a better reflection of its actual performance)

Ensuring correct assessments of pure Reddit data however was more difficult, due to no training data available in this sector. The training of this data was broken into two segments, each consisting of roughly 100 hand-labeled elements. The first segment consisted of labeled data which explicitly mentioned a political figure. On this data set, our classifier correctly labeled the political affiliation of the comment's sentiment in 82% of cases. For the most part, the misses in this instance were very short comments or comments featuring dialog and word choice not reflective of the comment's sentiment. We also tested categorization of

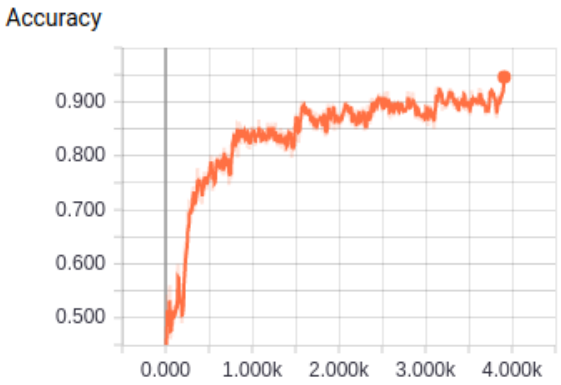

Figure 3: Accuracy of the LSTM for classification as it is being trained.

100 comments in which a political figure may or may not be explicitly mentioned. As stated earlier, when no political figure is mentioned explicitly the figure from the article's headline is used as the sentiment. Here, the classifier's accuracy dropped by a considerable margin to 68% when counting the comments to which our model perceived no strong sentiment, but 74% correct categorization when a sentiment was applied. It is worth mentioning that in both data sets only comments with a perceptible political sentiment by the manual tagger were included, and our results would surely be poorer if we were looking at a truly random selection of sample comments as, as detailed in the LSTM section, our model does not currently have much of a capacity for emotional ambiguity.

4 Conclusions

The results described above are far from ideal, but they do support the core concept which led us to this assignment, that political sentiment could be determined based upon the political affiliations of entities as well as the sentiment expressed in the verbal proximity of those entities. It also shows that even sentiment information from a radically disparate source (a movie discussion forum) can still have relevance when used to train to a model applied in a different domain.

5 Future Work

5.1 Improving Corpora for Sentiment Analysis

One of the primary limiting factors in the pursuit of true reflections of the political sentiment of a community is the lack of training data surrounding

such a community. While the IMDB data set provided an approximation of the sentiment of a given comment, it failed in some cases due to the alternate connotations of certain words within the contents of politics, and within the context of a back-and-forth discussion rather than the one-sided approach of IMDB reviews. In order to improve our model to the degree of comparable research, we'll need a robust corpus trained on Reddit data specifically. Fortunately, there are multiple alternatives available for future research within the realms of sentiment analysis.

5.1.1 Crowd-sourcing of Political Sentiment

Given much of the attention in recent media coverage paid to the influence of exclusively partisan "trolls" to influence online discussion, there is a good chance that many online political enthusiasts would be willing to participate in the tagging of online speech given the role of research utilizing this corpus in regards to stopping such behaviour. Given the ease at which extensions interact with Reddit, it would be relatively trivial to design one which would allow users to tag in-browser the political affiliation of comments and upload those tags to a database. A corpus such as this would enable us to tie certain language directly to political affiliations, even in scenarios when no entity is named.

5.2 Expansion of Current Data

The alternate method for the generation of a more robust corpus would be considerably easier as it requires no community or outside involvement, however it could easily lead to a feedback loop of over-fitting affiliation. Rather than looking at the comments of news articles as we are now, we could employ the same techniques in examining the post history of a particular user, easily parsable through the PRAW utility. If all of the comments in which a named entity is recognized connects to a certain political affiliation, then we could cast the assumption that all other comments within political subreddits in which no subject is detected are written with the same underlying affiliation - provided of course that a named entity is recognized in the title of the post the user us referring to. While this would enable us to drastically increase the size of our corpus, the fact that this robust corpus's data is predicated upon our current imprecise method may mean that we further ingrain current issues in our classification scheme. Perhaps the largely correct data achieved through this method can be cleaned through manual tagging by a site such as Mechanical Turk, though we're wary of using it in this situation due to both inaccurate tagging in past application of this service as well as the need for a knowledge of American politics and slang for correct parsing often absent from the generally non-American Mechanical Turk workforce.

5.3 Increased Entity Linking Accuracy

Presently, the affiliation of an entity is only returned if there exists a "Political Party" field on that entity's Wikipedia page. While this method works for politicians, many people such as Supreme Court Justices - who are meant to exist outside of the binary political system but still have liberal or conservative connotations - return no affiliation. As an alternative, it would be relatively easy to search through the page summary available through the already applied Wikipedia Python module for keywords denoting a political affiliation, but due to the sheer number of false options pursued, the computational/time cost of this option may be too steep.

5.4 Non-American Political Affiliations

One of the flaws in the current incarnation is that it only recognizes Republicans and Democrats. Given that the Wikipedia pages for most political parties notate their general location on the ideological spectrum from liberalism to conservatism and authoritarianism to anarchism, it should be possible to generate data for the affiliations of Reddit comments in a far more nuanced and flexible way rather than the current binary classification.

Acknowledgments

This project was supported by our professor, Heng Ji and our TA, Diya Lid whose insight greatly assisted our work.

We express our gratitude to RPI's BLENDER Cross-source Information Extraction Lab for their making publicly available the RPI Joint Information Extraction System.

We would also like to thank Reddit for their easy to use API, Wikimedia Foundation for their storing of freely accessible data, Natural Language ToolKit for their tools, examples, and documentation, as well as BeautifulSoup, Praw, Keras, TFLearn, and Flask.