Determining Political Slant Through Commentary

Benjamin Hillard
University of Massachusetts Amherst

Samuel Cox
University of Massachusetts Amherst

1 ABSTRACT

There is an increasing divide in not only partisanship politics by communities as a whole in the United States. With pressing issues and crisis at the forefront of the ballot this year, now more than ever the educated voter is needed. Rise of internet media outlets and social media means now positively anyone who wants to speak has a voice, and consequently a rampant spread of fake news and bias options have flooded the internet. It is easy to lose sight of fact in a spell of emotion. That's why the need for political slant analysis needs now more than ever. The everyday reader has a right to know the intentions of whatever they read, especially when the writing might be more subtle. Political slant analysis is a known challenge and many attempts have been made to conquer it. Most with little success. We proposed an out of the box solution, in which we ignore the texts itself, but instead focus on the response to text. We have noticed a trend in which subtle political writing usually results in less subtle reactions from the reader, if given a place to do so. We plan to exploit this trend with the goal of solving a sub challenge of political slant analysis

2 INTRODUCTION

The problem we are dealing with is the historically very difficult problem of predicting the political slant of a piece of text. It is common for people to want to classify the slant of a piece of news or the general slant of an entire news corporation. The issue of political slant prediction is important because political language and leanings can be conveyed very subtly, and can be difficult for the average reader to recognize. If a news article is obscuring reality with the goal of pushing an agenda it is very important to be able to recognize this. If people are armed with the knowledge that certain news outlets are lying to them to push a particular agenda, or twisting reality to fit a certain interest, they can seek other news or information that will be as truthful as possible.

Our approach to this problem is to tackle a smaller, more realistically achievable one for people at our experience level. Classifying the political slant of an entire news article is a very difficult task, so we chose to predict the political slant of the article a comment was made in reply to. This approach allows us to look at shorter, often more direct pieces of text and classify them. A comment left under a conservative article posted in a conservative area of the internet is likely, we have learned, to be very easily recognized as conservative, often more clearly than the article.

The research question that our project seeks to answer is whether or not the types of conversations happening in a conservative or liberal community on Reddit are a good indicator of the types of conversations happening in other supposedly like minded communities. We both expected a high level of homogeneity across these communities, and were relatively surprised by the amount of differences between them.

We found that attempting to predict the comments cross-domain was extremely difficult, while predicting the comments made using training data from the same community was surprisingly easy. We discovered that the easiest community to make predictions about was Reddit. After all, the commenters would have chosen to view articles on a subreddit pertaining to their specific political belief, so these comments sections were quite homogeneous. In contrast, ABC news comments sections, while mostly liberal, had at least a few conservative comments in all of them. Fox News comment sections were nearly as homogenous, from what we observed, as Reddit. We were able to achieve high accuracy scores training and testing on the same domain. Going cross domain caused our accuracy to drop significantly.

3 RELATED WORK

Using natural language processing to determine political slant is nothing new. We found more than a few papers written about attempting this. If anything was made blatant about these papers it was the importance of a solution and the challenge it represents. These papers acted as a great roadmap for what to do and not to do. We looked at two papers in particular, first being "The Perils of Classifying Political Orientation From Text" by Hao Yan, Allen Lavoie and Sanmay Das. the second being "Sentiment Political Compass: A Data-driven Analysis of Online Newspapers regarding Political Orientation" by Fabian Falck, Julian Marstaller, Niklas Stoehr, Soren Maucher, Jeana Ren, Andreas Thalhammer, Achim Rettinger and Rudi Studer.

"The Perils of Classifying Political Orientation From Text " was in a lot of ways the big brother paper to ours. They were using baseline political text they had received from congressional records, and testing cross domain on articles from political websites. Since we also had baseline data and were planning to test cross domain, we modeled our experiment after thiers. Their conclusion was that the cross domain problem had benchmarked their ability successfully classify. Because of this we were not shocked when we found ourselves unable to reach past 62 percent accuracy. Which is 11 percent below the cross domain benchmark they were stuck on. However we had hoped for more success due to the fact that commentary on articles tend to be less desicreet than in their bias than the articles themselves. Another problem they talked about in their paper was differing political jargon across the political compass as well from site to site. We ran into our own version of this problem when realizing how much the terminology in the political sphere changed in just the four years between our test data and our training data. We set out to replicate this study and improve the results by classifying comments instead of articles, but end up proving that commentary is subject to the same perils. We did our best to stay uniform to their study. We used large mostly neutral data to train on, while using more slanted data to test on. We both used logistic regression with term count as a feature. We also used data that had been lower cased and cleaned of punctuation like they did. With the exception of BERT we did most things similar as

we could to them. Thus showing that comments are no exceptions to there same perils

The second paper we looked at "Sentiment Political Compass: A Data-driven Analysis of Online Newspapers regarding Political Orientation" was much more successful. Their method was outside of the scope of what we were comfortable attempting, but definitely an inspiration. This paper initially peeked our interest because of the emphasis they placed on finding a solution to classifying slant. They phrased it as "If voters are uninformed about the political attitude of media reporting, they may be manipulated in their democratic opinion forming". Essentially we read this as while no political slant recognition system exists, all democracies are at risk for being manipulated out of their right to form a genuine option.

While this is not the route we chose to take in our approach, it should be noted that it yielded much more impressive results then anything else we could find. Their model was a data driven analysis tool. Basically they used a web scraper to be constantly grabbing relevant news articles. Then using a series of predetermined parameters, there algorithm sifts out reliant articles to train on. Then using sentiment analysis techniques, the sentiment of the article topic is determined. Using the writer's sentiment to a particular political topic then derives the slant.

This technique solves a lot of the issues we ran into. By using sentiment as the focus, lots of cross domain tools become available. And since they have a steady stream of current media to train on coming from the web scraper, the train becomes diverse enough to negate the effects of cross domain. We should also point out that, using a constant stream of new training data makes it so the model is keeping up with ever changing trends in political jargon. Ultimately this is the direction we should have gone and some aspects of their model were discussed in our problem solving. In the end we concluded that the techniques to be implemented were outside our skill and comfortability level.

4 DATA

We are using multiple datasets for this project. The main one, and the largest one, that we are using is a public dataset we found of comments harvested from a variety of subreddits, which happened to include several large sets of comments from political news subreddits, such as a libertarian news, conservative news, and liberal news (the main news subreddit on Reddit). This dataset was posted to Github in 2016, and the comments reflect this–the absence of rampant discussion about Donald Trump, for one example, sets them apart in a major way from news comments posted in 2020. Each comment document includes the comment, the name of the commenter, the number of upvotes the comment received and the number of downvotes it received.

At first, we focused on differentiating libertarian and conservative news comments, but switched to conservative versus liberal news comments because the difference in political beliefs would be larger (libertarians align with conservatives on a lot more things than liberals do), and because the only libertarian news website with a comments section we were able to find had comments sections plagued with bot messages and very little real user comments.

The other dataset we used was one gathered by us. We collected 300 comments from ABC news comments sections and 300 comments from Fox News comments sections. The articles we collected these comments from were all posted to their websites in November 2020.

When harvesting these comments, we had several restrictions on the types of comments we would use for our dataset. The comment had to be less than 500 characters long. It also had to be at least five words long–we wanted to avoid comments like "I disagree" or "Ya, ok whatever" messing with our results. We noticed that many comments on news websites tend to be very short disagreement/agreement comments like the ones above with little other context, or surprisingly long, often detailed responses to the article. The latter was the type of comment we were interested in classifying. We also only wanted comments that were replies to the article itself, not replies to other comments. Replies to other comments, we noticed, were often personal attacks or other types of conversation unrelated to the article in question, so we chose to not harvest them. Our final constraint was that we only harvested the top 30 comments on any given article that was specifically in the political news section of the website. This is both because some articles had thousands of comments, especially on Fox News, and because we assumed that the most liked comments on a particular article would best represent the attitudes of the community of that particular website.

These comments were labeled with the political slant of the corporation behind the article in mind, with Fox News being conservative and ABC being liberal. The actual slant of a particular article was not as important to us as the mindset of the community behind the comments. Fox News has a quite conservative following, while ABC news has a more liberal following.

5 METHOD

When attacking a natural language processing problem, we first want to gain a better grasp on the method we as humans use to process the problem. Anyone with the faintest idea of the goings ons of american politics can spot where the political offications lie in a comment. In gathering the data for the test set, we were given the opportunity to read tons of comments from both the majoritively conservative Fox news and liberal/central by contrast ABC. The first thing we picked up on during the annotation was grammar. Conservative leaning comments tend to have more misspelled words. In addition comments from conservatives tend to include more capital letters, more explanation points, and oddly enough more emojis. Unfortunately most of these potential features didn't make the cut do to the fact that our training data was so immaculately cleaned. The reddit data that we started the project with had been modified to all lower case letters and removed punctuation. It also contained sadly no emojis. At this point we considered pivoting and manually annotating a new data set, but there was no time to gather comments near the magnitude of what we already had from reddit. At this point we made the bet that large data would win out over less data with more potential features.

Previous to annotating our own test data. We ran several experiments on just the reddit data to gain more comfortability with NPL processes, and fish out possible interesting features of

the data. As mentioned before, we gained the majority of our data from a github repository containing several very large csv files pertaining to comments sections of various subreddits. We used news-news a file containing all comments tagged news from the main new subreddit, as a control. Being by far the largest file (5x the size of other news files) and without question the most diverse, it seemed like a good starting off point. We set out to test the control news-news against the more specific data from the other news file. This required first coming up with a feature to run logistic regression on.

As stated before the reddit data was cleaned down to basically just lowercase tokens. With little room for feature development, we decided to go with a simple bag of words implementation. To do this we created a function that would take a csv row of text and create a dictionary for each cell. We then appended each cell to an array and used scikit learn dictionary to vector function for formatting. We could then use this function to pass in some training data and it would return something numerical object that we could use as input for logistic regression.

Once we had a basic logistic regression up and running, we divided the data into all the possible permutations of class. First we tested each data set against the control new-news set, which we started calling vanilla news. In testing we received accuracy scores in the low 90s with r/conspiracy scoring the highest. Then r/libertarian followed by r/conservative. These results didn't shock us, one because the vast amount of data we were expecting high accuracy scores. And two coservative comments find their way into the mixing pool that is r/news much more often than conspiracy comments and or liberterian. This made those sets more distinguishable especially at scale. What was more shocking is that we had slightly lower accuracy scores when we transitioned to testing things like conspiracy vs conservative or conservative vs libertian or coservative vs libertaian. But nevertheless we felt confident our system was working so it was time to move on to the real deal.

At this time we had collected several manually annotated data sets to match our reddit data sets. We had hundreds of comments from lebertian websites such as reason.com, conservative comments from townhall.com and even conspiracy comments from websites like infowars.com. But when we trained on data like r/conservative and r/conspiracy, then test on annotated data from their counterpart websites our accuracy plummeted. We went from getting low 90s and high 80s to just under 54 percent accuracy. This began the struggle of cross domain. To discover a solution to this problem a lot of thought was involved. We started to therozie what the accuracy plummeted the way it did. The first thing we looked into was the age gap between the data sets. The reddit data was from 2016, nearly 4 years old. We did some basic word searching in the data. Some key differences: the token "Trump" appeared considerably less in reddit data compared to our annotated data. Another popular token in the annotated data "biden" was relatively non-existent in the reddit data. This turned our eye to another problem. Politics is a fast paced field, which means our bag of words implementation is only as good as the proximity the training set has in age to the test set. Even if it wasn't cross domain, placing the weight on word frequency would become an issue as popular vocabulary changes. As politicians and causes fall from the limelight a logistic regression trained on those weight words becomes less accurate.

This is where we turned our attention to neural networks. We considered trying several options. First we looked to GLOVE as possibly using the pre-trained word embeddings as some sort of norilazer for our bag of words model. The more we read into that the more outhere of an idea it started to sound. We also did some research into VADER but concluded that it was for sentiment analysis and wouldn't help us with our classification problem. Coincidentally this was around the time we learned of BERT. We decided that using the hidden layers of BERT trained on our reddit data, was the best option to approach cross domain. This is because BERT is trained on a wide range of internet text. We figured BERT not only had the best chance of performing cross domain but also keeping up with fast change political vocabulary.

We trained BERT on a combination of vanilla news and conservative news comments that totaled in around 26000 unique comments. This took a long time to do. It was estimated to take about 7 hours but ended up taking just under 3 and a half hours. When finished we were left with a tensor array for every comment in the data set. This changed the size of the data set from 20 megabytes to just under a gigabyte. This presented a new problem, we would have to find a quick way to save and load this data or endure several hours of training everytime we wanted to test on some comments. We saved the data to a csv, but unfortunately that converts the tensor arrays to strings which is difficult to convert back. Fortunately this is where we learned about pickles. Not the bar snack, but a file time that PANDAS conveniently offers for saving data sets. Essentially it will take a PANDAS data set, save it locally and compress it in under a minute. They call this pickling and unpickling. By doing this we were able to save the hours of training into a 280 megabyte file that we could quickly save and load from local storage. This saved us a ton of time. Once we knew the data was safe, we felt comfortable to continue with experiments. First we tested on some reddit data from r/conservative and vanilla news that we carved out prior to training to test on. This was wildly successful with accuracy scores averaging 99 percent. From there we moved on to our annotated data that had evolved to just 300 comments from various fox news articles and 300 comments pulled from ABC news. We ran the annotated data through some cleaning code to make it as uniform to the reddit data as possible. This involved lower casing all characters and removing punctuation. Then we trained it on BERT to gain a tensor array similar to what we had done with the reddit data. We then trained a scikit learn logistic regression on the reddit tensor arrays and tested on the tensor array we gained from the annotated data. This gave us an accuracy score of 62 percent. Which isn't great but it's an 11 percent gain from what we had without BERT. Bert also gave us a 9 percent gain reddit to reddit testing. From here we got curious so see what our results would be not cross domain. So we did a train test split on the annotated data, ran both through BERT and trained our logistic regression on the tensors. This gave us an accuracy of 75 percent.

6 RESULTS

The first experiment we ran was just using the Reddit data to see how well we could perform in the same domain before we tried to go cross domain. We took the data from libertarian news and conservative news. We then created simple bag of word feature vectors for the two datasets, using simple whitespace tokenization to create the tokens. This was our baseline method. We wanted to make sure that the task was going to be possible, and so using this method that was extremely simple and seeing results was important to us before we continued. Once we had the BOWs, we used scikit learn's logistic regression to train and test the data. The data was shuffled and then split .8/.2 between documents to be used to train and to test. We didn't bother doing any fine tuning on hyperparameters, this was simply a way to establish a baseline. Our accuracy was surprisingly high–with this very simple method, we were getting about 90 percent accuracy on our test set. The success of this baseline method was almost surely due to the amount of data we had. We had more than 5,000 comments from each subreddit we were classifying between, and we were ot attempting to classify cross-domain yet.

Our baseline test for going cross domain used a similar method to the one above. We collected a dataset that consisted of Fox News comments and comments from Reason.com, a libertarian news site. Unlike our final dataset described above, we had very few restrictions for the types of comments we would collect. We collected as many as we wanted from any given article, collected replies, and collected very short comments. The level of success of this baseline test would give us clues as to what kind of data would be good data and what wouldn't be, as well as giving us insight into how much more data we would need for the final project. We cleaned and tokenized the comments and created bag of words vectors with them, being sure to limit the words used to the vocabulary of the Reddit data. In total, we had 300 pieces of data to use, half from Fox News and half from Reason.com. After using all of the Reddit data to train a scikit learn logistic regression classifier, we then tested this classifier on the news data we collected. The accuracy was pretty abysmal, at only 55 percent. This showed us how difficult going cross domain was going to be, especially because the news comments we were gathering were not only from a different community, they were from a different year where drastically different things are happening in the world.

	Reddit train / Reddit test	Reddit train / News Websites test	News Websites train / News Websites test
Percent accuracy using BOW features	*90%*	*53%*	*70%*
Percent accuracy using Sentence Vector features	*99%*	*62%*	*75%*

To beat this baseline method, especially cross domain, we wanted to use a method that we thought would work better cross domain and also in the same domain. We considered using a neural network as the model instead of logistic regression, but we thought that using a neural network cross domain would be ineffective, as it learned very specific features about the domain it was in. Therefore, we used a neural network to create the features and logistic regression as the model, in hopes of overfitting less to the data and maybe learning some patterns that would help us predict cross domain.

The neural network we used was BERT, and we used it to create sentence vectors to be used as features for logistic regression. This would be better than bag of words for cross domain prediction for a number of reasons, but the most important was that we wouldn't have to throw out any of the data from any of our documents in the testing set from the news comments because they were not in the vocabulary of the training documents. There were a surprising amount of data that had to be ignored completely when using the bag of words model because of how many new topics have surfaced since the Reddit comments were made pre-2016 and how many topics that were urgent and important back then are rarely discusses now, but the sentence vector would take into account the whole document. Once we had fed BERT the tokenized sentence, we had to decide how to make our sentence vectors. We found that averaging all of the hidden layers was a very effective way of creating sentence vectors, rather than using a single hidden layer.

Streamlining the process of getting sentence vectors and then using them later was very important to us. Because of the size of our Reddit datasets, creating a sentence vector for each data set took about 7 hours. To make sure we never had to create the sentence vectors again, we saved them by saving them to compressed files with the pandas to pickle function.

Finally, once we made all of our sentence vectors and had an easy and time efficient way to save them, we began to run experiments with them. They gave us a good jump in accuracy when testing and training on the Reddit data (in an .80/.20 train/test split) up to 95 percent accuracy, which we felt was pretty astonishing. However, we were only able to achieve an accuracy of 65 percent in our Reddit train to news test experiment. Although this was a large leap from our last experiments, we didn't feel satisfied.

One thing this experiment had taught us, however, was that doing our prediction cross-domain was going to be harder than we were realistically going to be able to tackle. Our next experiment would give us a hint as to why it was even harder than we expected.

Even though we were achieving strikingly high results testing and training on Reddit data, when we both tested and trained on the news data (in a similar .80/.20 split as previous) we were only able to achieve 75 percent accuracy. This is probably in part due to the smaller amount of data–600 to split between training and testing compared with 30,000 to split between training and testing for the Reddit dataset. We also think that it was a result of the communities and the users in the communities we were trying to make predictions about. Generally, Reddit communities are surprisingly homogenous, we found through the process of this project. Also, Reddit communities evolve specific ways of speaking, specific shared cultural interests and moments, and specific likes and dislikes. However, most communities in the comments sections of news websites do not have this same level of shared community, especially outside of fringe news networks like InfoWars that have very loyal communities. A news network like ABC has a far reaching audience, but most likely a small loyal community compared to the people who post there. All of this is to say–it is unsurprising to

us that our classifier would do significantly better differentiating communities with many specific and often unique traits than it would with collections of mostly random commenters on a mainstream news website. To improve the experiment in the future, it would be best to collect data based on specific articles with a clear slant. If we were to improve the experiment, I think this could help a lot. We did our best to do this when collecting data, but collecting more data and making sure that the article they were collected from was indeed quite slanted liberal or conservative would help a lot in bringing out specific groups and kinds of commenters who would be more likely to read something with a specific slant that agreed with their views.

Still, it isn't a coin flip, there is clearly a trend for Fox News commenters to show similar ways of speaking and thinking as conservative Redditors. But the most successful attempts at classifying between conservative and liberal interested news websites/articles that we made relied on having data specific to that website to train with.

7 DISCUSSION AND FUTURE WORK

Our analysis shows that looking to alternative pieces of data such as comments to detect political slant has potential, and reveals the possibility that the main piece of data for slant prediction or political prediction might not need to be the article itself, and could include or be limited to (like in our project's case) the reactions to it from the community it was intended for. This means that we aren't looking at the reactions of random Twitter uses, for example, but instead looking at the reactions and thoughts of people who actually regularly use the website and are also rewarded by others for their contribution to the discussion (in the form of "upvotes" or "likes", depending on where on the internet data is being gathered from).

The next step for future work would be using a more complex/state of the art model and collecting a lot more data. It would probably also mean collecting entirely new data from the ground up in a more careful way, preferably from a larger number of websites over a longer period of publication times. Overall, our experiment failed to be as highly accurate as we would have liked, but shows some potential for further exploration as another strategy to classify political stance.