X Data as a Predictor of Political and Ideological Alignment

Eloragh Espie
eae2273
eloraghespie@utexas.edu

Rylan Vachon
rmv764
rylanvachon@utexas.edu

1 Introduction

In an age where digital footprints become increasingly defining aspects of our identities, understanding online communication's nuances is a skill many have begun to cultivate. In this research, we dive into the realm of social media data analysis to explore the predictive power of Twitter content in determining an individual's political stance as measured by the Political Compass. By combining data collected from the Political Compass' existing election data with a corpus of scraped tweets, this project aimed to understand the potential of text data in signaling political ideologies.

This project is important because it offers a unique perspective on text data's insights concerning political leanings. As social media platforms continue to serve as metaphorical arenas for political discourse, understanding the explicit and implicit cues within users' language becomes increasingly relevant for political media literacy. Our research underscores the importance of exercising caution in online communication, as text-based forms of expression can be easily misinterpreted, potentially leading to incorrect assumptions about individuals' political or ideological affiliations.

Through this research, this paper aims to contribute to a deeper understanding of the connection between online discourse and political identities, providing insight into the capabilities and limitations of using text data for predicting political orientations. This research's outcomes are valuable for academic inquiry and have practical implications for online communication strategies and political campaigning.

2 Data

For this project, we collected data using several methods. To collect the coordinates needed to train each model on the X data, we manually tracked and formatted data available to us from The Political Compass. To collect a set of tweets for each politician, we used a Twitter API Wrapper called Twikit.

2.1 Manual Data Curation

The manual data curation involved roughly twelve hours of work in total. Given that it was more prone to human error than automated data collection, the process took multiple steps to ensure the highest data quality possible.

We did this process several times, ensuring the names, X handles, political parties, and coordinates were accurate before the data was inserted into a SQL table[1]

We used the pre-existing election data available on the Political Compass. In total, we collected data from 38 different elections across the globe, spanning 2005 to 2023. We collected 221 data points, which accounted for 153 unique politicians. Several of these politicians ran in more than one election and observed slight deviations in their x and y coordinates from election to election, so a new record was created for each election to account for these changes. Out of the 221 data points, 11 of the politicians had no X profile to pull data from and were excluded from the scraping process. Of the 210 remaining, 193 data points had at least one tweet during the relevant election year. This was the final count of scrapeable election data points.

2.2 Automated Data Curation

Once the coordinate data was curated, we began work on developing the automated tweet scraper using Twikit. This process was long and likely inefficient due to the many workarounds required to

[1]This data can be accessed via the source code under data. It is the coordinates csv.

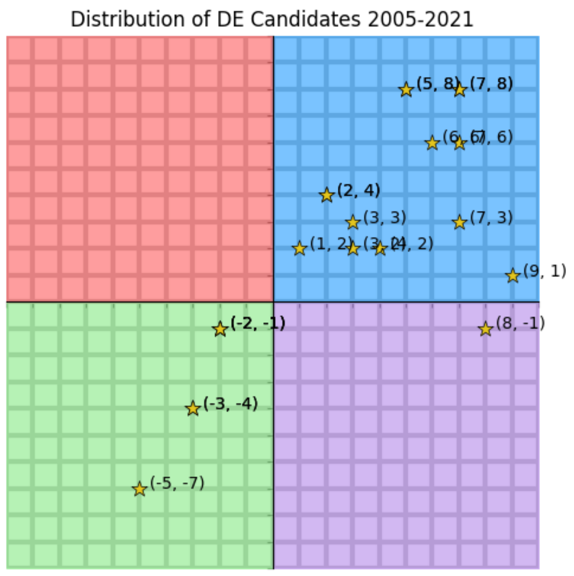

Figure 1: Manually curated data points for German elections from 2005 - 2021.

pull such a mass of data.

API wrappers are incredibly helpful when it comes to subverting paywalls such as the X paywall, but they are also prone to identification and bans. The most frequent issue we ran into was rate time outs. Purchasing the official API is one hundred dollars per month for fifty thousand data pulls. If we had purchased this, we would have no problem pulling data until we hit that limit, which we likely would not have. However, very few university students have a hundred dollars to spare. By going the alternate route, we were able to hit the same endpoints as the official API, but they each had a specific number of hits one authentication token could receive every fifteen minutes. I am unsure how these rate limits are decided, as they seemed fairly random. One allowed for fifty hits per 900 seconds, another ninety-five, and another one hundred and fifty. Regardless of the origin of these seemingly arbitrary time-outs, we used the time module to stagger our API calls so we could continuously run the script.

We ran into several other issues, such as read timeouts and authentication errors after hitting an endpoint too many times, but none were as prolific or aggravating as the rate issues.

Due to the inconsistency of the script's run time, we could only amass four thousand and forty-four tweets. That amounts to an average of roughly twenty-one tweets per politician data point.

3 Methodology

Our methodology involved an iterative approach in regards to the preprocessing. We trained two linear regression models, one to provide the x (economic) coordinate and the other to train the y (social) coordinate.

3.1 Preprocessing

Our initial goal was to perform three iterations of this project using three different types of preprocessing: CountVectorizer, GloVe embeddings, and BERT. We still intend to attempt the BERT iteration, but in the interest of time, this paper will only discuss the first two iterations.

CountVectorizer

GloVe embeddings

3.2 Feature Extraction

Feature extraction for CountVectorizer included utilizing the CountVectorizer tool to tokenize, fit, and transform the text data into a vector array. GloVe required the use of a lemmatizer, tokenizer, and the Pandas DataFrame tool to create a vector array. Both underwent train test splitting to create a train size of 80 percent and a test size of 20, with random states set at 1 for the x-axis and 3 for the y-axis.

3.3 Model Training

Model training was the same for both the CountVectorizer and GloVe arrays. Both utilized the SciKit-Learn Linear Regression model with models created for each axis, x and y.

3.4 Evaluating the model

This project spanned a broad range of potential outcomes. Both linear regression models had a potential output scale of -10 to 10. Due to this wide range, we had to approach evaluation differently. We were not especially concerned with the model being completely accurate in the individual coordinates. Instead, we used mean absolute error to evaluate the model.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i) \tag{1}$$

MAE takes the sum of the output value (x_1) minus the expected value (y_1) for all values of n. It then divides that sum by the n number of datapoints to find the average error among the entire testing set.

4 Outcomes

4.1 Model Outcomes

Our model was only able to process our CountVectorizer array, as we did not have enough time to properly implement the GloVe array data. Our x-axis CountVectorizer model achieved a 3.31 MAE, 1.31 points over our minimum requirement of 2. Our y-axis CountVectorizer model did much better with a 2.97 MAE, though it is .97 points over our minimum requirement.

4.2 Project Outcomes

With more tweets and a proper implementation of the GloVe embeddings the MAE could dip under 2. This would help make the project accurate enough to be effective in predicting a users political alignment through their tweets. This would make finding a users political alignment easier than taking the official questionnaire.

5 Team Structure

5.1 Eloragh Espie - Data Collection

Eloragh Espie was responsible for writing several scripts that ingested, processed, cleaned, and formatted all of the data necessary to train each linear regression model.

The bulk of this work was on the script that allowed for the automated scraping of tweets using Twikit. Due to rate limits, this script took several iterations and much patience to gather a reasonable amount of data.

She was also responsible for storing and retrieving the data, as well as uploading it to sharable resources and the source code.

In the source code, there is a folder where all of her code is available for review.

5.2 Rylan Vachon - Model training

Rylan Vachon was responsible for preprocessing the data provided and using it to train both linear regression models. He wrote the code used to generate the political compasses based on our data.

Rylan's work also took several iterations as he attempted to preprocess the data several different ways in order to understand if context was helpful with such short form data.

He was also responsible for implementing the MAE evaluation to determine if our output fell within an acceptable range of error.

In the source code, there is a folder where all of his code is available for review.

5.3 Other responsibilities

Both Eloragh and Rylan contributed equally to all aspects of this project and agreed to the work required for their respective roles. All papers and presentations were done in a joint effort.