Taylor Imhof

Bellevue University | DSC 680

Project Two: Milestone #2 (Draft White Paper)

11/12/2022

BUSINESS PROBLEM

An organization has approached me to design a sentiment analysis application that they can use to gain better insights into how their customers are reacting to different products and services that the organization provides.

BACKGROUND/HISTORY

Sentiment analysis is a relatively new and exciting application of the branch of machine learning called natural language processing (NLP). With NLP, computers are trained to be able to understand human language and use it in different ways. With sentiment analysis, the NLP models are trained to pick up on whether a statement is positive, negative, or neutral. With regard to the business problem, understanding whether the organization's customers are happy or upset is very important to know how to curate its offerings to better serve and retain its customer base.

DATA EXPLANATION (DATA PREP/DATA DICTIONARY)

In order to gather the data for this project, I found a useful Python package called the Python Reddit Application Programming Interface (API) Wrapper (PRAW). This package allows me to easily obtain data from Reddit's website via an API that the Reddit team has exposed for third-party developers. Using the PRAW package, I obtained the headlines from the popular sub-reddit r/politics where the community shares and discusses current events under the domain of global politics. It is very common for people to

get emotionally charged, so I thought that this would be an interesting sub-reddit to investigate. The API simply provides the headlines in textual format, so there was some data munging that needed to be performed to get the data into a format that I could more easily work with.

METHODS

After obtaining the textual headline data and storing it into a simple Python list, the next step was to run a sentiment analysis tool over all of the natural language data. To do so, I utilized functions provided by the Natural Language Toolkit (NLTK) package. I downloaded both the lexicon that would be used with the specific analyzer (detailed shortly) as well as a pre-made set of stop-words. Stop-words are essentially meaningless and otherwise un-informative words that do not expose whether a statement is positive or negative. Downloading this pre-made set would make processing the data more streamlined.

The next step was obtaining the values measuring whether each headline was positive, negative, or neutral. For this step, I made use of the <SentimentIntensityAnalyzer> (SIA) function provided from NLTK's VADER class. VADER, which stands for Valence Aware Dictionary and sEntiment Reasoner), is a sentiment analysis tool that was designed specifically with the use case of social media in mind. The SIA function made it very easy to iterate through each of the headlines and calculate polarity scores that measured whether each headline was positive, negative, or neutral. I stored the results of the polarity scores as well as the headlines themselves in a Python dictionary. In order to convert the dictionary into a data structure that is easier to work with, I made use of Pandas' <pd.from_records> function to convert it to a Pandas dataframe.

Here is an example of what that dataframe looked like:

	neg	neu	pos	compound	headlines
0	0.264	0.736	0.000	-0.7269	Karine Jean-Pierre blasted for warning ...
1	0.000	1.000	0.000	0.0000	Lauren Boebert in Precarious Position a...
2	0.000	0.709	0.291	0.5719	Kelly wins in Arizona, pushing Democrat...
3	0.000	0.821	0.179	0.3400	Herschel Walker is a pawn in a game he'...
4	0.103	0.561	0.336	0.5423	911 dispatchers overlooked in $1,000 bo...

As you can see, the first four columns are metrics that show how strongly the headline is rated in terms of negativity (neg), neutrality (neu), and positivity (pos). The compound column provides an aggregate of all three values and was used to generate the final label column that would be used when analyzing the performance of the VADER model. The headline column simply contained the textual headline that was obtained in the previous step from the Reddit posts themselves.

Using the values in the newly calculated compound column, I encoded a new label column that would be -1 for negative, 0 for neutral, and +1 for positive. I assumed that if the compound was [-0.2 < x < +0.2] that it was neutral (0). Otherwise, if it was > +0.2 then I labeled it as positive (+1) or if it was < -0.2, I labeled it as negative (-1).

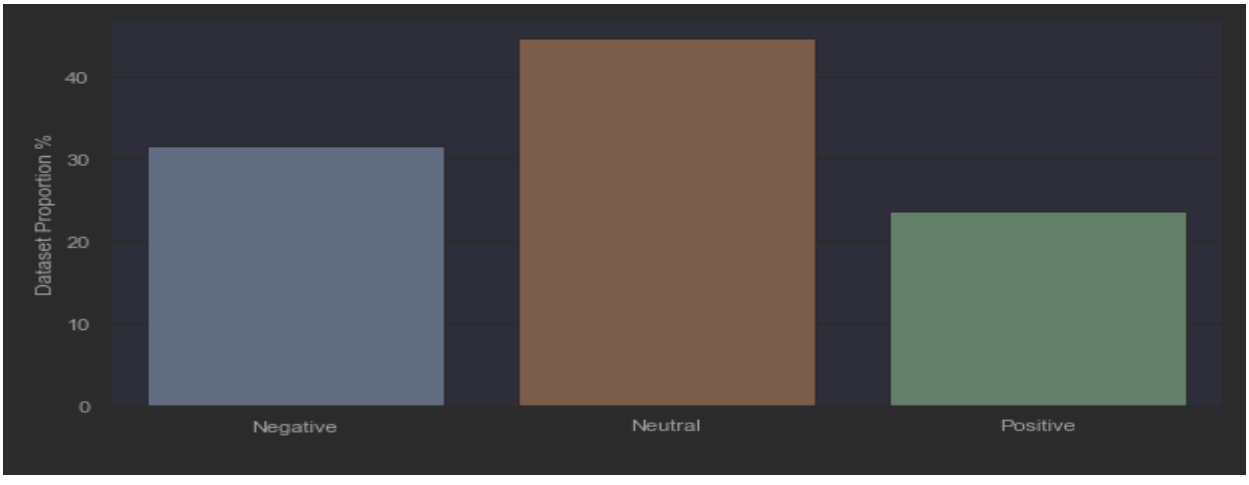
ANALYSIS

After creating the polarity scores and encoding them to cleaner, labeled values, it was time to visualize the data and analyze how well the model had picked up on the nuances of the English language. For example, I took a peek at some of the headlines that had been labeled negative to see if I would also categorize them in that manner.

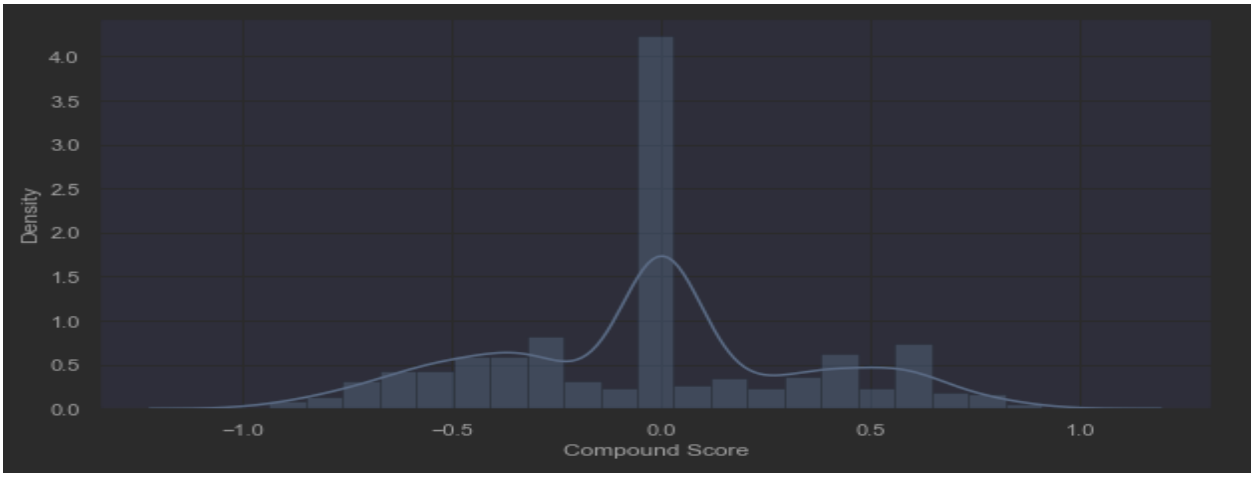
```
Below are some samples of negative headlines:

['Karine Jean-Pierre blasted for warning it may take ‘a few days’ to '
 'count votes: ‘This insanity has to stop’',
 'Gen Z helped to stop the ‘red wave’ in the midterms. The Republicans’ '
 'response? Try to raise the voting age',
 'Opinion | The Emperor of Chaos Has No Clothes',
 'Democrat Marie Gluesenkamp Perez scores upset victory in Washington '
 'House race',
 "GOP bomb goes off: Turning on 'toxic,' 'loser' Trump after humiliating "
 'losses']
```

After reviewing the first few records, it does seem that the model did pick up on some of the words that would be categorized as negative. For instance, “insanity has to stop”, “toxic”, “upset”, and “loser” are usually words associated with negative statements. For this reason, I would say that the model has been trained relatively well, though I would have to reserve that judgement for when I provided the trained model to the organization that requested my assistance to see how it performed on “live” data.

I felt it was also important to see how much of the data points were distributed in the different polarity categorizations (i.e., positive, negative, neutral). To visualize this, I made use of the popular Python visualization libraries Matplotlib and Seaborn.



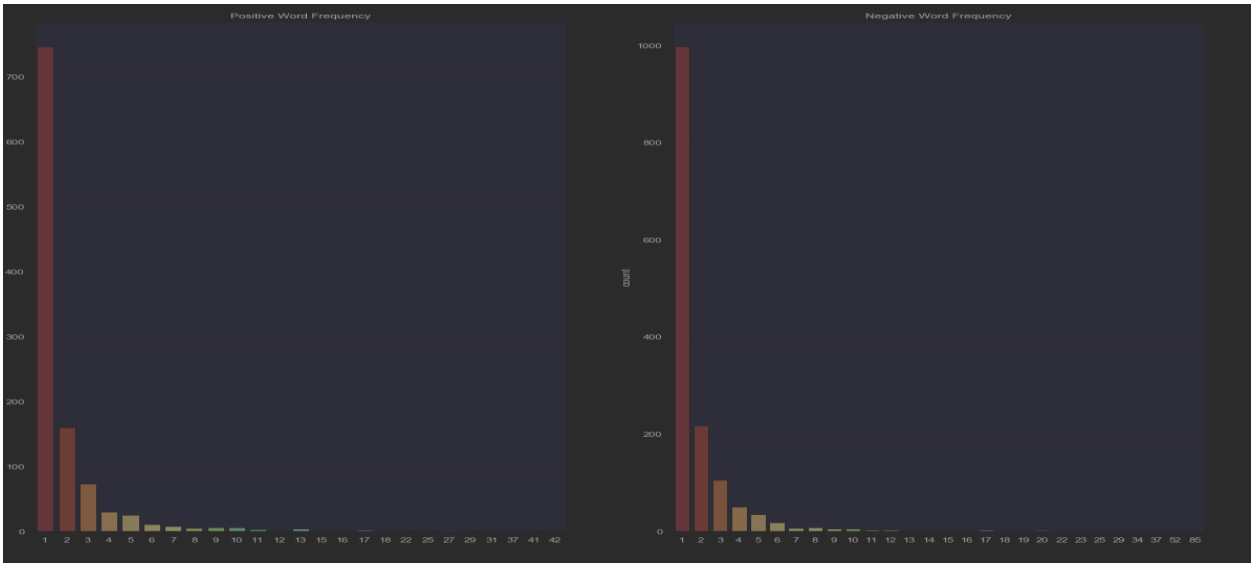
After reviewing the graphic detailed above, it was quite clear that the model did contain a significant amount of bias towards neutral headlines. While this does sort of make sense in the context of news headlines (people wanting to stay bipartisan), it does present a potential issue that it hasn't been trained enough on the positive and negative sentiment which would be more useful to the organization on whose behalf I am conducting this work.



To further illustrate this bias, I also created a simple distribution plot. As you can see it is quite clear that most of the data points have been labeled as neutral (indicated by large peak at 0.0).

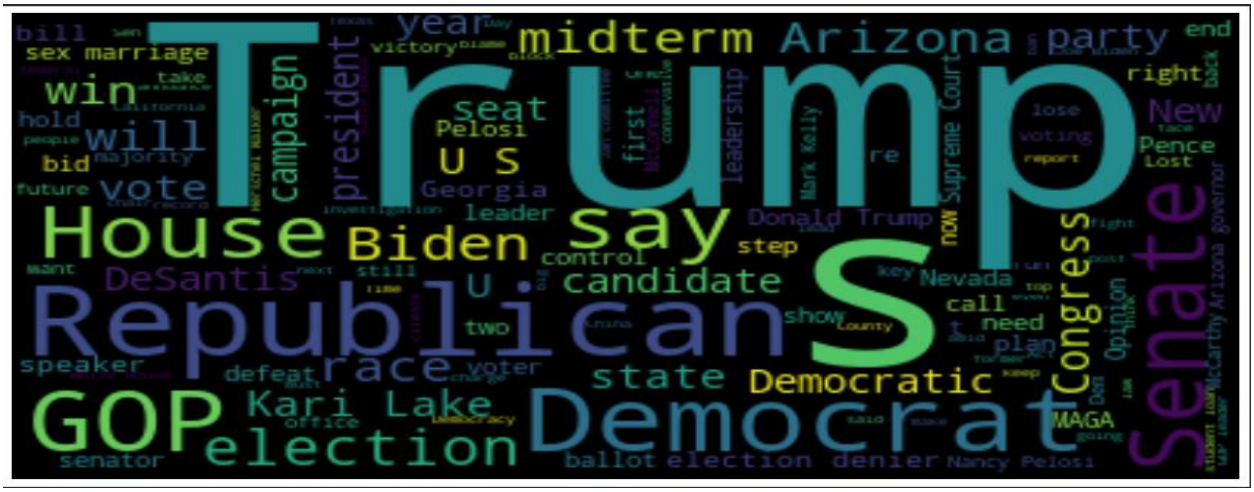
In order to analyze the impact of specific keywords, I ran the data through a tokenization function, and used the stop words that I downloaded in a previous step to remove uninformative words. By tokenizing the headline strings, I would have a list of the words along with their associated polarity metrics so I could have a more granular understanding of their impact in the sentiment analysis. After doing so, I found that the three most common positive tokens/words were: 'trump', 'democrats', and 'senate.' On the negative side, I found that the three most common tokens/words were: 'trump' (interesting), 'election', and 'gop'. Given that these data points were collected around the time of the mid-term elections, they were not all that surprising. Trump is still such a hot topic in the political realm, though I was still surprised that he was at the top of both the negative and positive lists.

I also wanted to see how these tokens were distributed among the entire collection of tokens. To do so, I created a count-plot using the Seaborn library for both the positive and negative token lists.



As can be seen above, some of the tokens are much more widely used than others. In order to understand the impact of this, it would be important to compare with the results of tokens from other domains, such as sports or weather.

The last thing that I created was a basic word cloud of the political headlines. I feel that anytime I have read through the findings of an NLP project, there is always a world cloud. To create this, I made use of the <wordcloud> package, and passed in the values of the headlines from the dataframe created early in the project.



The word cloud provides another way of looking at the similar findings that I used different approaches for as outlined in the previous sections. However, the word cloud format makes it much easier to pick up on some of the commonly used words.

CONCLUSIONS

After reviewing the findings of the trained VADER model, I feel that the sentiments of the headlines have been adequately calculated and labeled. It would take a thorough analysis of each of the headlines, as well as the review of a trained linguist to truly understand how well the model has fitted to the natural English language. However, I feel comfortable that this model could be used by the organization to begin understanding how their customer base is responding to their products and services.

ASSUMPTIONS

The main assumption that I made during this project is that the polarity scores calculated by the VADER model on political headlines would be applicable to the domain of which the organization operates within. It is possible that their specific industry is not best suited for sentiments derived from political-slanted language. However, I feel that as a baseline, using this sentiment analysis model would provide

the organization with something to begin working off of that could be iteratively improved upon with future data.

LIMITATIONS/ CHALLENGES

One of the limitations that I encountered during this project was that the PRAW package only allowed me to collect just under a thousand headlines per call. While a thousand data points might seem like a decent amount, modern machine learning models are now capable of processing hundreds of thousands of data points without much trouble, especially with advances in big data processing solutions such as Apache Spark. One solution would be to collect the data using PRAW over the course of, say, a month and then concatenating all the data into a consolidated dataframe. That way, there would be much more than just a thousand rows of textual data to train the model on.

FUTURE USES/ADDITIONAL APPLICATIONS

I feel that the trained sentiment analysis model could be used on other topics of English discussion, such as sports and other unique forums. There could be an issue that since the model was trained on political topics specifically, it would not be able to pick up on some of the nuances of the language when used in different conversations. The model was also trained on headlines, which typically aren't written the same way that people communicate with each other. As such, the model also might not pick up on some conversational subtleties such as metaphors or jokes.

RECOMMENDATIONS

As mentioned above, the model was only trained on headlines. I would recommend that the model be extended to be trained on various types of written communication. For instance, once the organization uses the model on their customer's reactions, they could at the same time collect the text of the comments and utilize them to further train the sentiment analysis model. Also, I would recommend

continuing to collect more r/politics headlines to better train the model. It would also be interesting to capture additional datetime metrics so that perhaps a time-series analysis could be conducted to analyze trends of which political topics are discussed during different times.

ETHICAL ASSESSMENT

One of the major ethical considerations that would have to be made with this project is user consent. Since I used textual data that was written by real people online, it was not clear whether these individuals would have been alright with me using their submissions for this project. There still is not a whole lot of robust policies or regulations in place regarding the usage of data. However, it does seem that there are a lot of efforts in the pipeline to protect user data. Hopefully, soon there will be stricter guidelines as to how online data can be used and who can use it.

AUDIENCE QUESTIONS

1. How did you ensure that there were no duplicate headlines when you were collecting the data?
2. Why did you choose to collect data from r/politics instead of another sub-reddit?
3. What made you decide to train a VADER model as opposed to another sentiment analysis tool?
4. Why did you choose to encode neutral values between the range -0.2 and +0.2?
5. Why do you think there were so many headlines that were labeled as “neutral?”
6. Why did you choose to use Seaborn to visualize your data as opposed to a visualization application such as Power BI or Tableau?
7. Could the VADER model be used to pick up on the sentiment of another language besides English?
8. Why have you written your project using Python rather than another language such as R or SQL?
9. Why do you think that ‘Trump’ was the most common word for both negatively and positively charged headlines?
10. Do you think machines will be able to replace human language translators in the future?

REFERENCES

Apache Spark. (n.d.). Spark Overview. Retrieved from https://spark.apache.org/docs/latest/

NLTK. (n.d.). Natural Language Toolkit (Documentation). Retrieved from https://www.nltk.org/

PRAW. (n.d.). PRAW: The Python Reddit API Wrapper. Retrieved from https://praw.readthedocs.io/en/stable/

Python. (2022). Python 3.11.0 documentation. Retrieved from https://docs.python.org/3/.

Reddit. (n.d.). r/politics. [Sub-reddit]. Retrieved from https://www.reddit.com/r/politics/

vaderSentiment. (n.d.). VADER-Sentiment-Analysis [Github Repository]. Retrieved from https://github.com/cjhutto/vaderSentiment