Analyzing the sentiment of the US Elections

As displayed in tweets using Google Cloud Services

Group 11

R.C.E.A.M. van der Heijden	2048313
P.J. Jongenelen	2046980
E.M.J. Mans	2047839
J. Sanderink	2048198

04-11-2020

1. Overview of the data pipeline

Pipeline function

As it has not escaped from the attention of many, the US presidential elections are taking place at the moment, with election day itself on November third. The Republicans with current president Donald Trump are up against the Democrats with their representative Joe Biden in a race for the White House. The campaign teams of both candidates are very interested in the public opinion and how to influence the electorate in such a way that they will vote for a specific candidate. One way of monitoring the public opinion is reading tweets, since Twitter is a free source of psychological wisdom (Reips & Garaizar, 2011). However the quantity of tweets has exploded and it is practically impossible to manually read all the tweets about the election of the United States. The need for an automated method able to handle large volumes of data rapidly is born; at this point the pipeline making use of Google's Google Cloud Services comes in.

The purpose of the pipeline is to read the tweets which stream in from the Pub/Sub topic *election_data*, then sequentially: cleaning the tweets, assessing the sentiment score of the tweet using the pre-trained Natural Language ToolKit (NLTK) VADER lexicon, and writing these results into a table on the BigQuery service. The earlier mentioned campaign teams are able to do (almost) real time analysis of the public opinion through the medium of twitter. So broadly, the current pipeline reads the published data from the Pub/Sub topic *election_data* and notices the BigQuery Dataset *election_data* and within this dataset the table *tweet_info*.

Goals and Requirements

We defined two goals for our pipeline: the first is to successfully process and score tweets on the sentiment of their language, and the second goal is being able to classify the tweet as containing the names of Joe Biden or Donald Trump. As our use case was to handle streaming data, we needed our pipeline to be able to read data from a Pub/Sub (though we do not provide streaming data, see Discussion for explanation). By transforming raw tweet input data to the BigQuery format with sentiment labels and scores, the pipeline ensures that the output can be stored in a format that allows SQL queries for end analyses or (in this case) visualisation tasks. This implementation makes it easier for campaign teams to analyze the general sentiment in the world and in their electorate, with data analysts being shielded away from any data engineering part, allowing them to only be concerned about their end analyses.

The required input of data should be in the form of the official twitter API. This form consists of 18 features (which can be found in Table 1, page 5). Of this input data format, our pipeline will select the categories it needs for further processing; these column indices are hardcoded. These are the **content** of the tweet, the **date** of the tweet, and the **author**'s twitter handle (username).

2. Design and implementation of data pipeline

Data architecture

As the first assignment led to tremendous difficulties in our group and unfortunately was not successfully finished, we had double the motivation to deliver a working data pipeline this time. To limit the amount of potential sources of errors, we decided on a relatively easy pipeline, such that we can show that we are indeed able to create a working one. A conceptual overview of the architecture can be seen in Figure 1; it's intricacies are explained beneath the image.

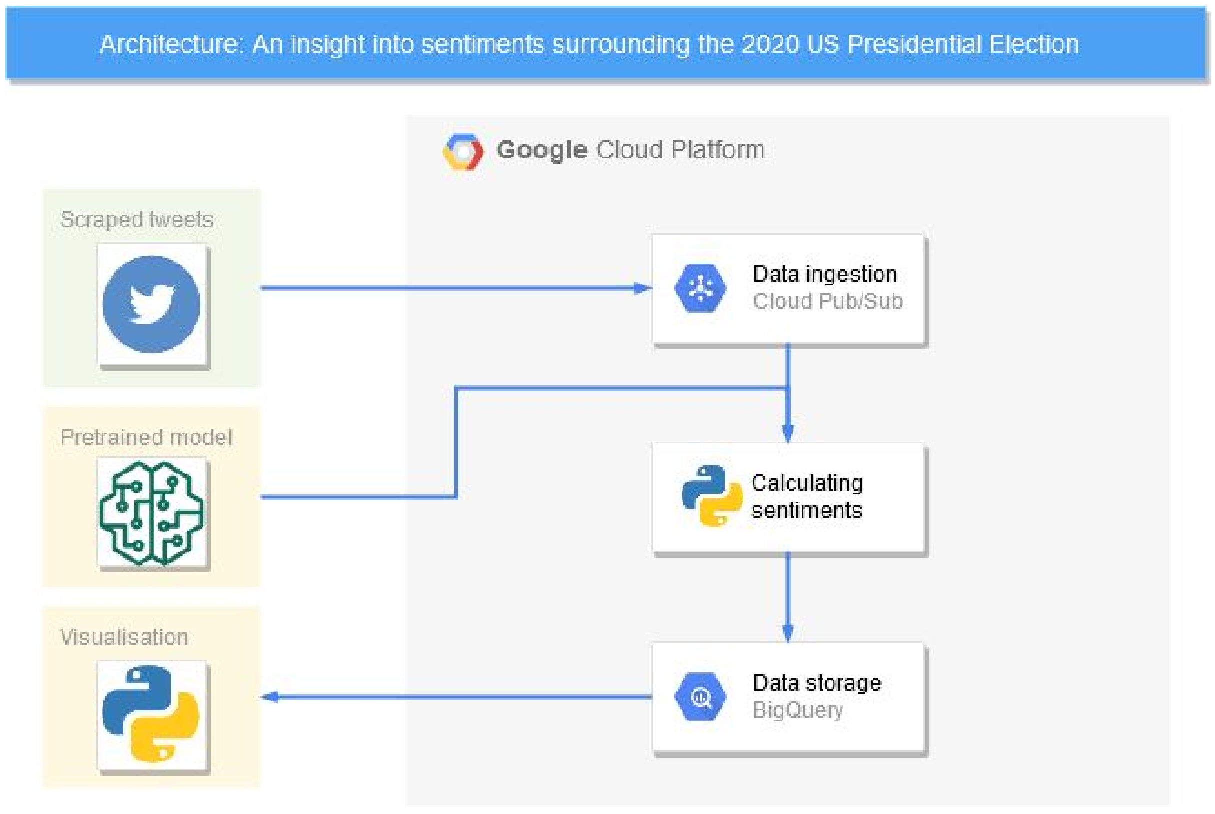


Figure 1: A global, conceptual overview of the architecture.

We have scraped tweets from Twitter locally, using Orange which is an application in the Anaconda environment. By using the Twitter add-on we scraped the tweets and stored them in a csv-formatted file. We used a Pub/Sub topic to be able to communicate the data from the csv file to the topic using the file *pub.py*. And from the Pub/Sub topic to the BigQuery data warehouse using the subscriber *reader.py*. Then we apply the NLTK SentimentIntensityAnalyzer using the Valence Aware Dictionary and sEntiment Reasoner (VADER) lexicon, which calculates the sentiment of a single tweet. These results are stored in the BigQuery database called *election_data*, along with the original row of data. To be more precise, they are stored in the *tweet_info* table. Using SQL queries we can access that data. A more in depth overview of the architecture, with the name of every part included, is visible in Figure 2.

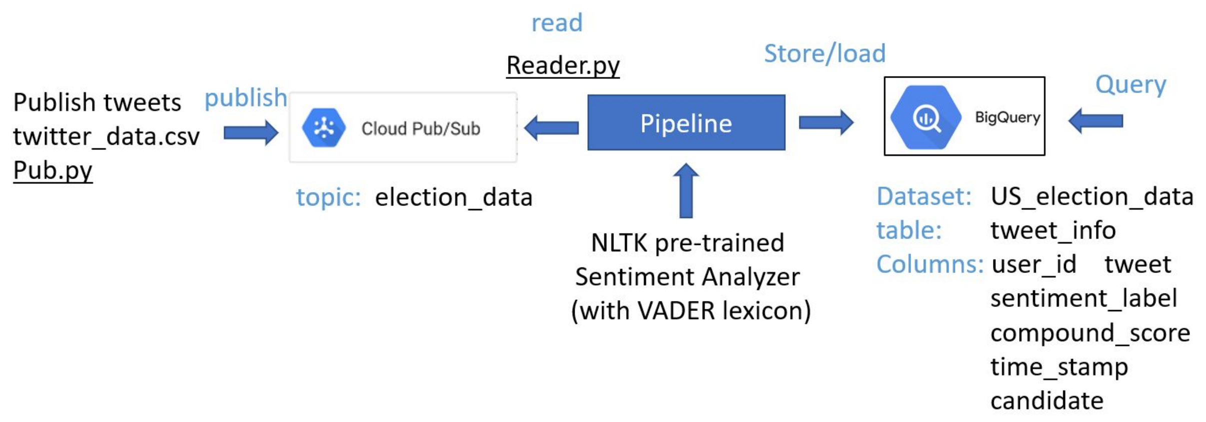


Figure 2: An in-depth overview of the architecture, naming all relevant components.

Processing Workflow

The dataset was scraped from Twitter using the Twitter API. We selected the tweets to: a. be in the English language and b. contain the strings "Biden" or "Trump" (tweets with both strings are also allowed). In total 5564 tweets were scraped and saved in a csv file. The 18 data columns present in the file can be seen in Table 1.

API columns

Content	In reply to	Author followers count
Date	Author name	Author listed count
Language	Autho description	Author verified
Location	Author statuses count	Longitude
Number of likes	Author favourites count	Latitude
Number of retweets	Author friends count	**Author**

Table 1: Data accessible via the Twitter API. The columns of interest are bolded.

The pipeline is designed in such a way that it expects the tweets with all the mentioned 18 features as input, using the earlier described format of a csv exactly. The data in the above described format is injected from the csv file into the subscriber of the Pub/Sub topic *election_data*. The pipeline includes a subscription to the same Pub/Sub topic and thus reads in the raw tweets with the associated metadata.

As our use case applies to streaming data, we have to set a windowing parameter. Given that we only provide data in a single batch - that is to say that we provide the data once and update the publisher with new (live) data - we set the windowing parameter to 1 minute. This is purely a practical decision, helping to mitigate the reader running needlessly without seeing any new data (the reader should have streamed all currently available data roughly in about 5 to 10 seconds). So, as a real life implementation of this pipeline would use the streaming implementation, we've done exactly that in our project, but a separate script scraping and providing constant live data would have to be added for that functionality. As we do not have that functionality, the windowing parameter is just 1 minute, though real life usage would benefit from a larger window as the application is not very time dependent.

Once these tweets and their corresponding data arrive at the beginning of the pipeline, the data of interest is selected. Selecting these three columns is hardcoded and occurs in the second phase of data selection (see Figure 3). The *text* (content of the tweet), *user_id* (username) and *timestamp* are extracted from the raw data and flow further into the pipeline, while the other data is ignored. The tweet text then is parsed during phase 3, after which it gets cleaned by our preprocessor.

The tweet body is then cleaned in several steps; all the words in the tweet are converted to lower text, the hyperlinks are dropped (phase 4), numbers are removed (phase 5), symbols are removed (phase 6) and finally the interpunction is dropped in phase 7. This way the text is clean and readable for the sentiment analyzer.

Next is the actual analysis of the tweets. We use the NLTK package for Python, and specifically the Valence Aware Dictionary and sEntiment Reasoner (VADER) lexicon (Hutto & Gilbert, 2014). This is a pretrained model. In phase 8, every tweet gets a sentiment score, and a label dependent on the score is added (positive, neutral or negative).

Every tweet is labeled with a 'candidate' label. These can take the following values: ['Trump', 'Biden', 'Both']. This label ensures that it is easy to categorize tweets between the parties and do separate analyses on them.

Finally the tweets, their relevant metadata, the sentiment and the labels are written to the BigQuery table *US_election_data.tweet_info* (phase 9).

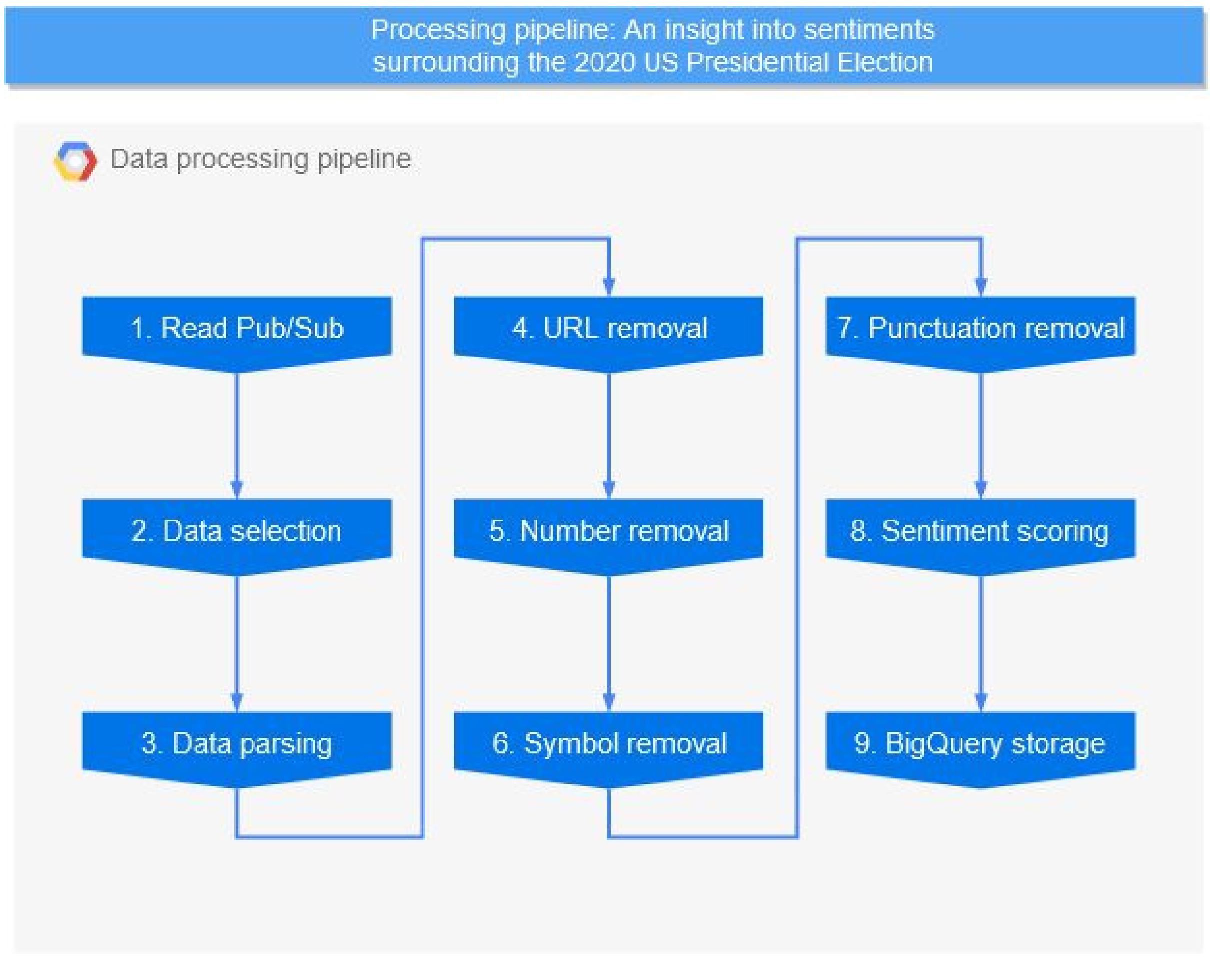


Figure 3: A schematic overview of the processing pipeline. All steps are done sequentially, with steps 4 to 7 falling under data cleaning.

3. Evaluation of data pipeline

We have split the evaluation of our project into two separate segments; first we will look into the focal aspect of the assignment: the performance from an engineering perspective. Afterwards, we will look at the results of our pipeline from a machine learning perspective, including some of the visualisations possible. Do note, we are aware of the fact that we didn't optimize our model nor the data preprocessing.

Pipeline performance

As desired, we ran our pipeline on three different machine types. The machine types we used were: n1-standard-4, n2-standard-4 and e2-standard-4, with the 4 in the name of each indicating that there are 4 vCPUs. All of them are available on Google Cloud Services. Table 2 shows an overview of the differences in their specifications (Google, 2019). They share the max number of persistent disks (128) and the max total persistent disk size (in TB) of 257.

	Memory (GB)	Local SSD	Maximum egress bandwidth (Gbps)
n1-standard-4	15	Yes	10
n2-standard-4	16	Yes	10
e2-standard-4	16	No	8

Table 2: Machine types specifications. Only the specs which differ among the three machines are shown.

Compared to n1 types, n2 machine types are the second generation. They can achieve higher per-thread performance, while benefiting from the same flexibility. The e2 types offer the lowest on-demand pricing, but they do not offer sustained-use discounts (as they have no local SSD).

n1-standard-4

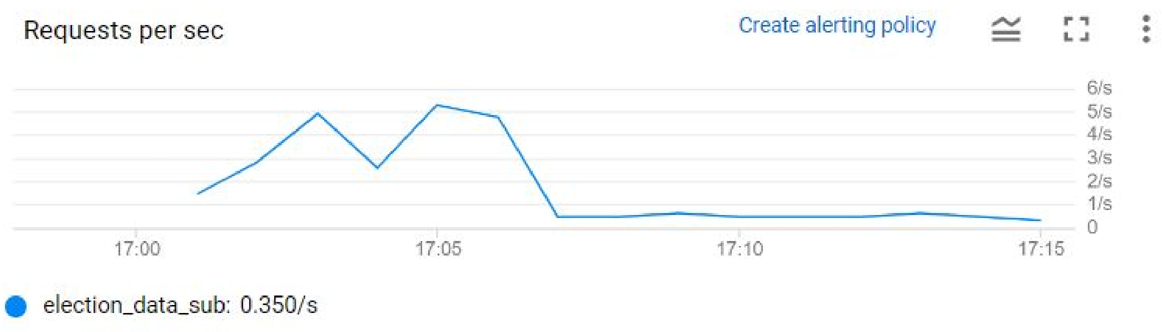


Figure 4: Request rate of our pipeline running on the n1-standard-4 machine type.

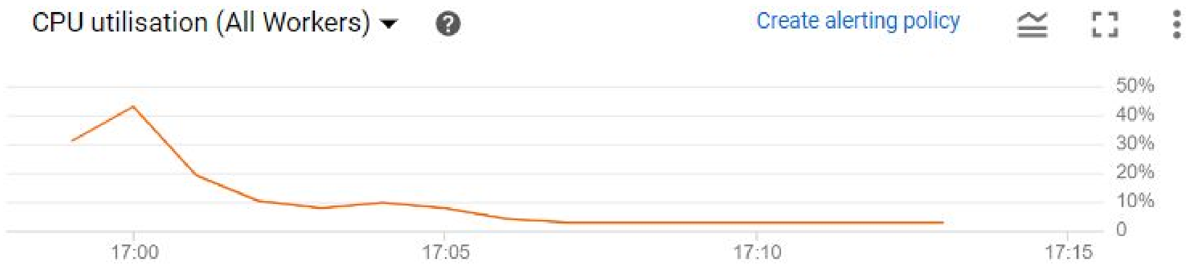


Figure 5: CPU utilisation of our pipeline running on the n1-standard-4 machine type.

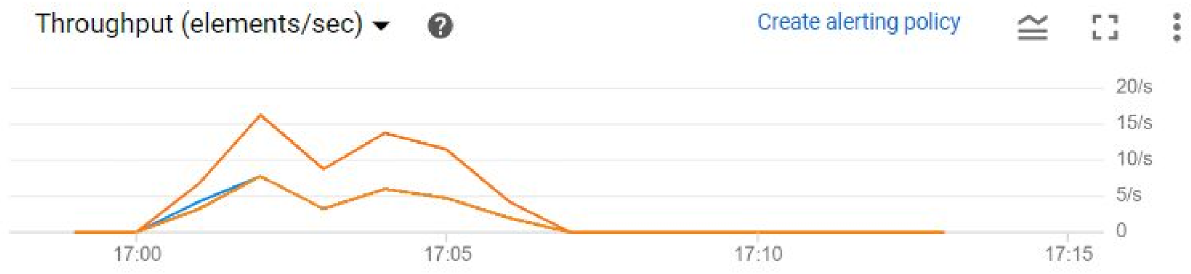
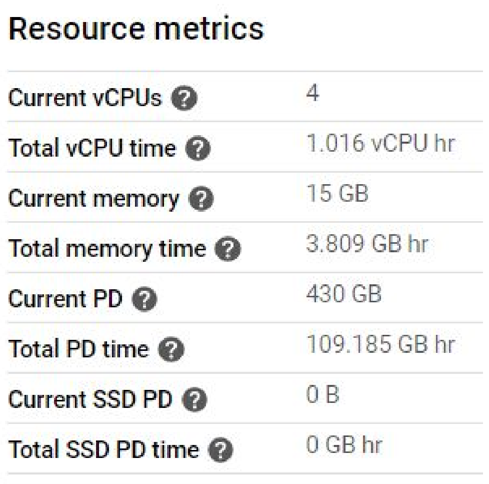


Figure 6: Throughput of our pipeline running on the n1-standard-4 machine type.



Resource metrics

Current vCPUs	4
Total vCPU time	1.016 vCPU hr
Current memory	15 GB
Total memory time	3.809 GB hr
Current PD	430 GB
Total PD time	109.185 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr

Figure 7: Resource metrics of our pipeline running on the n1-standard-4 machine type.

n2-standard-4

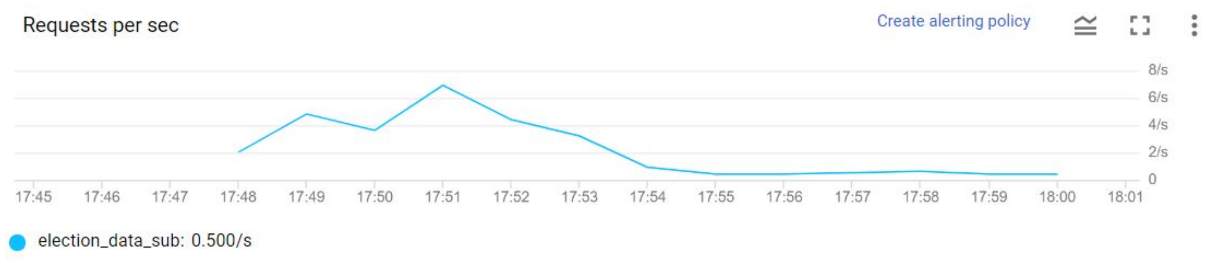


Figure 8: Request rate of our pipeline running on the n2-standard-4 machine type.

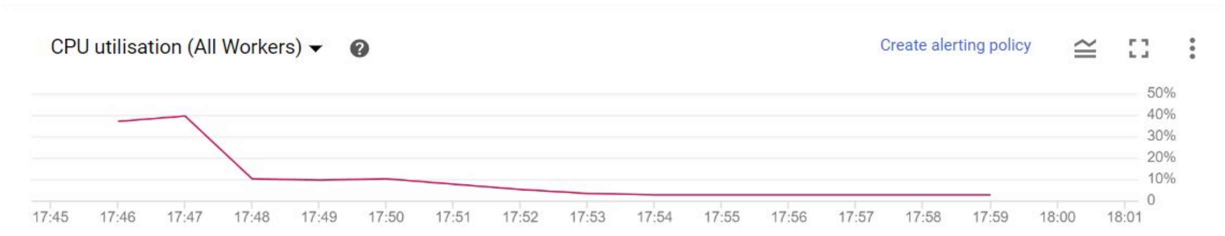


Figure 9: CPU utilisation of our pipeline running on the n2-standard-4 machine type.

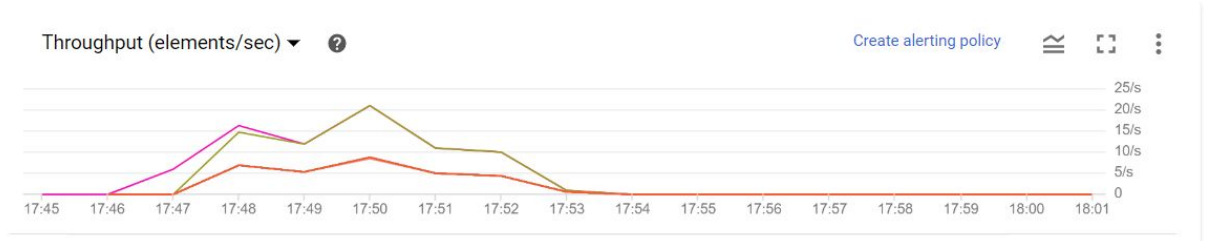
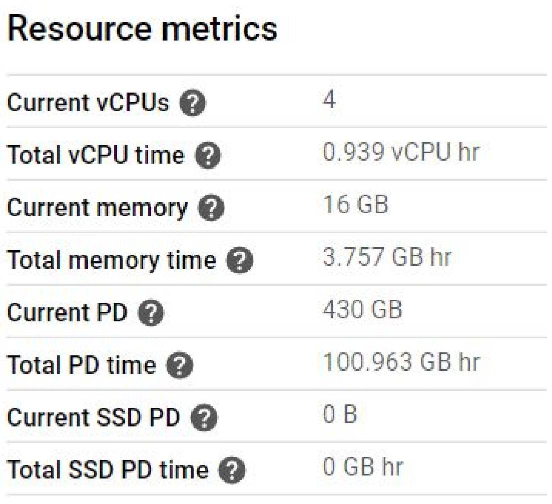


Figure 10: Throughput of our pipeline running on the n2-standard-4 machine type.



Resource metrics

Current vCPUs	4
Total vCPU time	0.939 vCPU hr
Current memory	16 GB
Total memory time	3.757 GB hr
Current PD	430 GB
Total PD time	100.963 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr

Figure 7: Resource metrics of our pipeline running on the n2-standard-4 machine type.

e2-standard-4

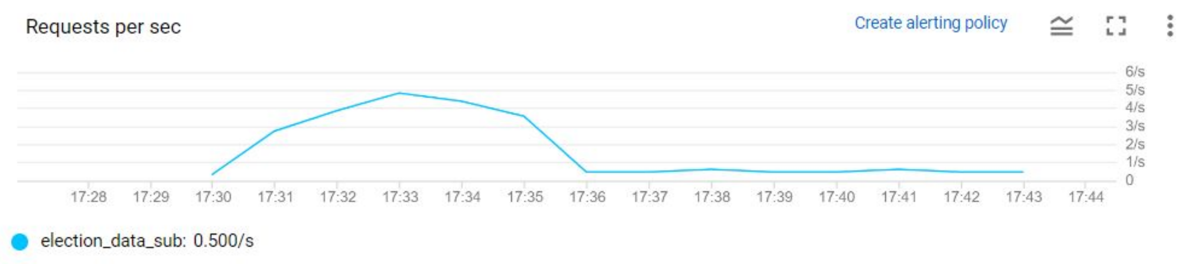


Figure 12: Request rate of our pipeline running on the e2-standard-4 machine type.

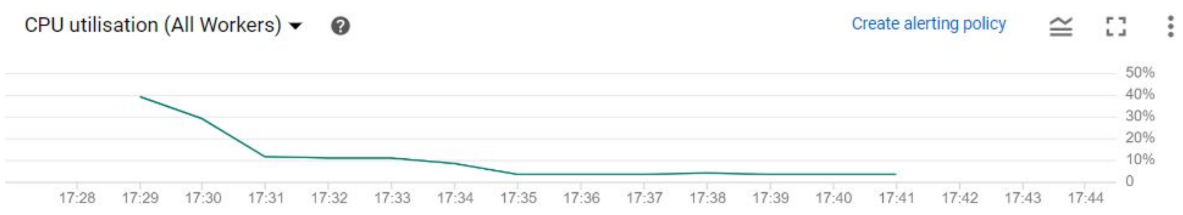


Figure 13: CPU utilisation of our pipeline running on the e2-standard-4 machine type.

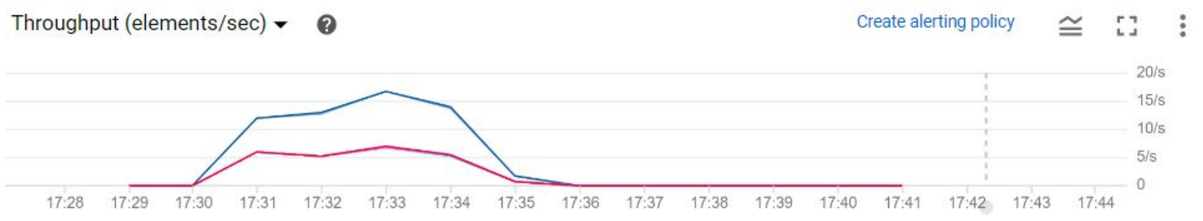
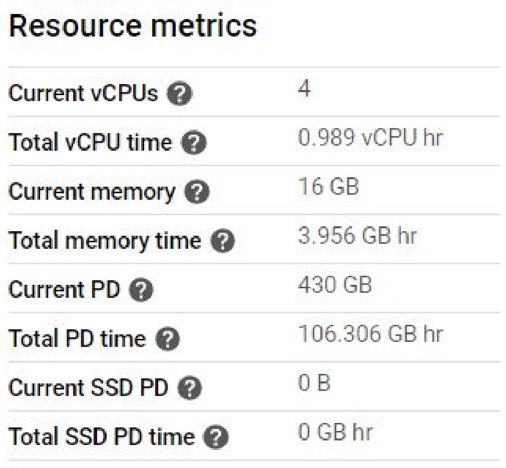


Figure 14: Throughput of our pipeline running on the e2-standard-4 machine type.



Resource metrics

Current vCPUs	4
Total vCPU time	0.989 vCPU hr
Current memory	16 GB
Total memory time	3.956 GB hr
Current PD	430 GB
Total PD time	106.306 GB hr
Current SSD PD	0 B
Total SSD PD time	0 GB hr

Figure 15: Resource metrics of our pipeline running on the e2-standard-4 machine type.

All nine plots show a relatively similar distribution, so we decided to bring the resource metrics to a comparison in Table 3. Because we have to manually stop the pipeline, the time the pipeline ran unfortunately is not equal, though we let them run for comparable times. The results are very, very comparable in the other metrics as well. The n2-standard-4 has the shortest total vCPU time by a relatively small margin. It also has the smallest total persistent disk time, with close to 10% less than the previous generation n1 type. However, we think that our pipeline might not be calculation intensive enough to see large differences, which we regret.

	n1-standard-4	n2-standard-4	e2-standard-4
Time left running	16m37	15m28	16m37
Total vCPU time	1.016	0.939	0.989
Total memory time	3.809	3.757	3.956
Total PD time	109.185	100.963	106.306

Table 3: A comparison between metrics of the three selected machine types.

Output

In order to gain insights into the results generated by the pipeline, we created a Jupyter notebook (*visualize.ipynb* on GitHub) that retrieves the data from BigQuery and visualizes it. We performed three different visualizations. The first checks whether the sentiment analysis model classifies texts correctly. The second looks at what sentiment prevails in tweets about the candidates.The third was a 'fun' additional analysis that explored whether positive or negative sentiment correlates with the length of the tweet. This section will summarize our findings thereof.

In order to check if the sentiment analysis model makes sense on our dataset, we printed the three most positive, and three most negative tweets. The sentiment of the tweets classified conforms quite well to the sentiment we would attribute them to. Even though it is only a glimpse at the data, we are happy that the sentiment analysis correctly classified these tweets.

The second analysis covers the main use-case of our pipeline. We want to see if there is any difference in the sentiment present in tweets about one or the other candidate. In order to achieve this, we plot three histograms on top of eachother, one for each of the candidates and one for tweets mentioning both Trump and Biden. The x-axis shows the valence score of the sentiment analysis. We see a similar shape for all three series, namely a large portion of neutral texts, flanked by two similar distributions, one in the negative spectrum and one in the positive.

The third and final analysis checks the correlation between tweet length and sentiment score. The Pearson's correlation coefficient equals 0.05, and the accompanying scatterplot clearly indicates that there is no correlation whatsoever between these two variables in our dataset. The six tweets and two graphs mentioned in this section can be found in Appendix A.

4. Discussion

Introduction

We have taken the liberty to stretch the segment of "Individual contributions of the students" into a broader part of our report, namely discussion. It is split into three subparts, namely critical remarks, a reflection on the assignment and the group process and the desired description of the individual contributions.

As, despite our devotion and several helpful sessions with mr. Kuma, we unfortunately could not deliver a fully functional implementation for the first assignment, we were wary about this assignment as well. This led to several decisions to improve our chances of creating a pipeline with full functionality, but choices that conversely negatively impacted the scientific rigor. Given that the primary goal of the assignment is to create a data pipeline on Google Cloud Services, we felt it was justified, but we feel the need to mention, describe and support our decisions here.

Discussion

First of all, we started the assignment on time. This results in our data being scraped on the 22nd of September, rather than November 3rd (the actual Election day), which would be somewhat more interesting. However, we believe this still to be fine for this assignment, nonetheless a note on this felt appropriate.

Secondly, we used a streaming pipeline as Twitter data can constantly (or periodically, with a small periodicity) be scraped. In a real world environment, this would be the preferred implementation, allowing for constant monitoring of sentiment and seeing changes to it live. However, to be able to achieve this, a script that continually scrapes Twitter would need to be added. We have not done so, both in order to combat complexity and to try to preserve our Google Cloud Credits.

Lastly, our data cleaning at the moment is very rudimentary. Though hyperlinks, numbers, symbols and punctuation are removed, that's where the cleaning stops. If we were to remove Twitter handles (mentions) and stop words, the results of the model would in all likelihood be better. We advise against taking any analysis done here at face value and advise conductors of future research to exclude mentions and stop words.

Process reflection

Whereas in our previous assignment we mainly worked collectively with one person typing and 1 to 3 watching and giving input, we now decided to see whether a more individual approach would work better, achieve better results and be more efficient. The flipside of this approach is that ultimately, in our case one person has delivered virtually all code, as he/she found a successful implementation first, while the others followed other paths that eventually did not work (yet). We have to conclude that this one person was responsible for practically the entire codebase of our final project, while the rest lagged behind and has no concrete results to show for their efforts. What approach has our preference is still to be discussed internally, however we have seen both the positives and the negatives of our two approaches so far.

We haven't done any official collective reflection as of yet, but we noticed that we were better prepared for this assignment than the previous one. The examples provided by mr. Kuma were helpful, and we were able to figure out most of the functionality as a group. Just like the previous time, mr. Kuma was very forward with his assistance, for which we want to express our appreciation once more.

Individual contributions

Despite our efforts to assist and help her, Ellen Mans has done the vast majority of the programming for this assignment. Rodger van der Heijden, Tim Jongenelen, and Joost Sanderink have put in a lot of time and effort into getting a conceptual understanding of the topics and how to code them, but their efforts proved fruitless. Ellen Mans picked up the topics a bit faster and thus got to the final results. The three gentlemen also did help Ellen with setting up the BigQuery environment and the understanding and usage of git.

The remainder of the tasks was distributed over the other three group members; Tim Jongenelen took care of the visualisation, while Joost Sanderink and Rodger van der Heijden were responsible for the report.

References

Google. (2019). *Machine types | Compute Engine Documentation | Google Cloud*. Retrieved from https://cloud.google.com/compute/docs/machine-types

Hutto, C. J., & Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014*, 216–225. Retrieved from http://sentic.net/

Reips, U. D., & Garaizar, P. (2011). Mining twitter: A source for psychological wisdom of the crowds. *Behavior Research Methods*, *43*(3), 635–642. https://doi.org/10.3758/s13428-011-0116-6

Appendix A: Output from visualize.ipynb

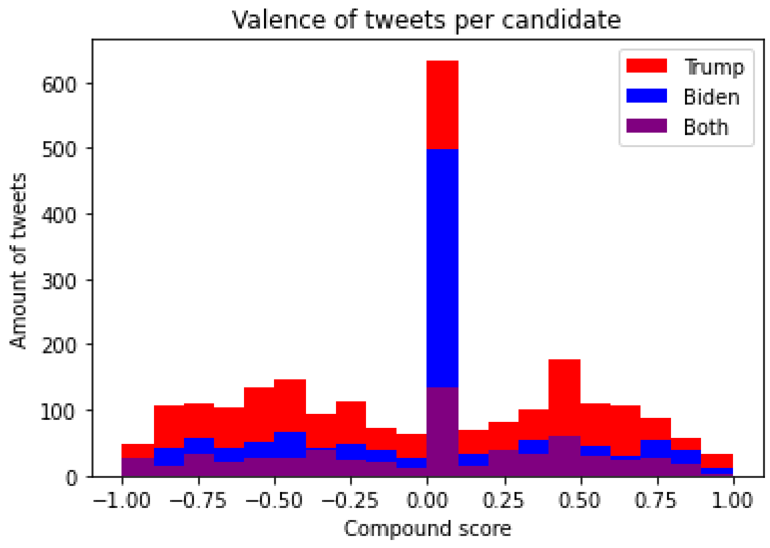
Three most negative tweets:

```
ezinger jaketapper trump raped a  year old and then threatened to kill her
family thats disgusting and there is an affidavit to prove it wheres the
proof giuliani shouldnt throw stones because they have pics with his pants
down literally;

mikepence realdonaldtrump disgusting trumpwhos a wife cheater tax avoider
child kidnapper traitor extortionist and lying fake covid can have him the
rest of the country will vote to put joebiden as president of the usa;

the only thing that we get out of this freak trump with the wig and paint
on his face is smears smearssmears lies lies lies and  deaths dear god
please get this evil creep out of our familys lives so fatigued and
depressing;
```

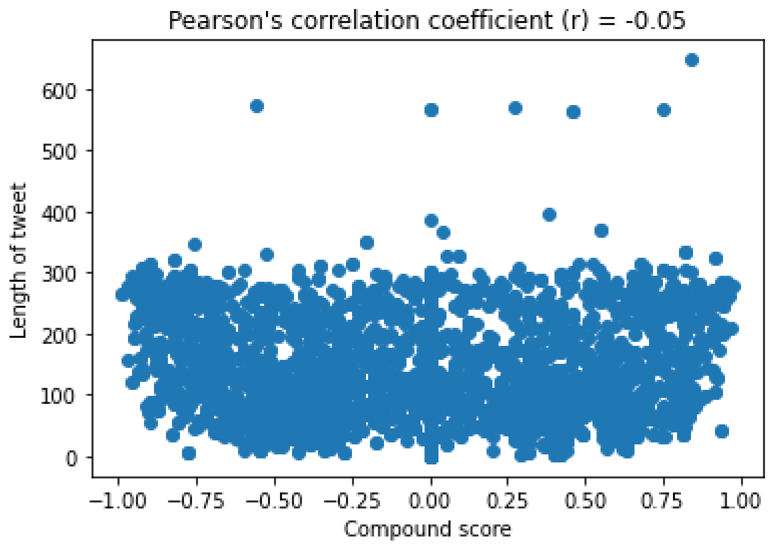
Three most positive tweets:

```
sadiececile why because president trump is the best candidate for our
economy for jobs for minorities and people support his causes we live in
america the land of freedom this is not china or cuba iam a proud american
and i support trumppence amp i only wish the best for everyone;

hannahalesiii  no cheesiness or cliches just sending love your way you are
an amazing mama to those sweet babies also trump calls himself stable you
dont want to be like him i hope your day gets better beautiful;

remarkable bipartisan consensus about us grand strategy great power
competition trump administration and free world biden lead to similar
conclusions us needs to focus on competition with china and russia and in
order to succeed it relies on the support of allies;
```

Sentiment of tweets per candidate, as well as for both candidates



Correlation between sentiment score and tweet length


Pearson's correlation coefficient (r) = -0.05
Length of tweet
Compound score