

Sentimental Analysis of Reviews

C964 – Computer Science Capstone

Western Governors University

Contents

Prompt A.....	4
Letter of Transmittal.....	4
Project Recommendation.....	6
Problem Summary.....	6
Application Benefits	6
Application Description	6
Data Description.....	6
Objective and Hypothesis.....	6
Methodology	7
Funding Requirements	7
Stakeholders Impact.....	7
Data Precautions	8
Developer Expertise	8
Prompt B.....	9
Project Proposal	9
PROBLEM STATEMENT	9
CUSTOMER SUMMARY.....	9
EXISTING SYSTEM ANALYSIS.....	9
DATA.....	9
PROJECT METHODOLOGY.....	9
PROJECT OUTCOMES.....	10
IMPLEMENTATION PLAN	10
EVALUATION PLAN	11
RESOURCES AND COSTS	11
TIMELINE AND MILESTONES	12
Prompt C.....	13
Application Files	13

Prompt D.....	14
Post-implementation Report.....	14
Project purpose	14
Datasets.....	14
Data product code.....	17
Hypothesis verification.....	20
Effective visualizations and reporting	20
Accuracy analysis.....	22
Application testing.....	23
Appendices	24
Installation Guide	24
User Guide.....	25
Summation of Learning Experience	27
References	28

Prompt A

Letter of Transmittal

Dustin Yansberg
Machine Learning Engineer
BURGERBURGERBURGER
123 Sesame St.
New York, NY 10001
July 12, 2023

Hughey Hubert
Chief Technical Officer
BURGERBURGERBURGER
123 Sesame St.
New York, NY 10001

Subject: Project Proposal for Sentiment Analysis of Reviews

Dear Mr. Hubert,

I hope this letter finds you and other members of the executive team well. I am writing to you to propose a project that could impact our company's ability to analyze and respond to reviews posted on platforms such as Yelp and Google Reviews. This project, centered on a type of machine learning called sentimental analysis, has the potential to lead to a great improvement in customer satisfaction as well as an increase in our ability to manage our online brand reputation.

Sentimental Analysis uses advanced natural language processing techniques to build a machine learning model that can extract sentiment information from a given text. In essence, if we feed negative and positive reviews into a model with the goal of performing sentimental analysis, the model will be able to identify if the review is positive, negative, or neutral. With this information in hand, our marketing teams here at BURGERBURGERBURGER can quickly and accurately identify the sentiment of a review posted online at several platforms the moment they are posted, which will enable them to apply their expertise in addressing the review accordingly.

Attached to this letter is a more in depth look at the elements of this proposal. I will provide a clear outline of the problem summary, a description of the application I intend to deliver along with its benefits, a description of the data to be used in this project, clear

objectives, how the project will impact stakeholders, its funding requirements, and any data precautions to consider, along with a wealth of other good-to-know information.

Here at BURGERBURGERBURGER, we take our customer relations seriously, and for that reason, I believe this project is not only in-line with our vision and culture, but also has great potential to see us move into a brand-new phase of growth. Please feel free to browse this proposal at your leisure, and should any questions arise, do not hesitate to reach out to me or any member of my team to discuss it.

Sincerely,

Dustin Yansberg

Project Recommendation

Problem Summary

There are many review sites where customers may find information about our business. To name a few, there are Yelp, TripAdvisor, Google Reviews, and Facebook Reviews. When a customer posts a review for BURGERBURGERBURGER on any of these platforms, it is imperative that we are equipped to respond as quickly as possible. As it currently stands, days may pass before we know a new review is posted, and that doesn't even take into consideration how long it might take us to analyze the contents of the review and respond to it. In short, we need to analyze and respond to our reviews in a timelier manner.

Application Benefits

- Immediate analysis of posted reviews
- Better customer relations
- Better online reputation management

Application Description

- When a review is pasted into the appropriate field and the application is run, the application will determine whether the application is negative or positive.

Data Description

- The data used for this application is a collection of publicly posted reviews on Yelp curated into a public Kaggle dataset.

Objective and Hypothesis

- The objective is to ensure that the user of this software can quickly assess the sentiment of an online review for BURGERBURGERBURGER quickly.
- My hypothesis is that using a multinomial Naïve Bayes classifier model trained on reviews classified as positive and negative, I can predict with relative accuracy the sentiment of a newly input review.

Methodology

I have decided to use the waterfall methodology for this project. This methodology works best because the requirements for this project are well defined, and this type of project has ample resources available for understanding and research. The model we will be using is a multinomial naïve bayes model since we'll primarily be dealing with word frequencies as features.

Funding Requirements

This project can easily be completed within two 40-hour work weeks by one employee using the equipment and software tools that have already been provided to the employee, so the relative cost of this project is low. The table below provides a more detailed breakdown of the project costs:

Resource	Description	Cost per unit	Totals
Work laptop	Computer Laptop employee already has	\$0	\$0
Work hours	The estimated number of hours needed to complete task	\$50 X 80 hrs.	\$4000
PyCharm	IDE for running Python scripts -- cost is per user first year	\$249.00 X 1	\$250
Python	Programming language already running on work laptop	\$0	\$0
Maintenance	The annual hours needed to maintain the software	\$50 X 20 hrs.	\$1000
		Total	\$5250

Stakeholders Impact

There are many stakeholders to our company, and all of them will be impacted positively by the implementation of a tool such as the one described in this document. Customer's will be able to be heard and have a more accessible open dialogue with BURGERBURGERBURGER because of our increased ability to respond to their unique and individual concerns. Marketing team members will be able to more effectively address positive and negative reviews accordingly in a timely manner. Investors in our company will see potential revenue growth from our improved online reputation management.

Data Precautions

The data for this product is publicly available from Kaggle.com and it consists of legally acquired public reviews on the Yelp website. The data contains no personal information about the posters of the reviews. It simply has a column with the text of the review and a column of the text's sentiment score. Since this data is so anonymized and made publicly available, there are no ethical or legal concerns with the development and use of this software.

Developer Expertise

The developer building this software is experienced in building and maintaining machine learning products, and I have no hesitation in recommending them for the development of this tool. This developer has been with BURGERBURGERBURGER as a Machine Learning Engineer for more than five years and has produced several products on their own in the past.

Prompt B

Project Proposal

PROBLEM STATEMENT

Our marketing department is effective at responding to reviews online, however their response rate is much slower than would be optimal. Sometimes days might pass before marketing can read and digest a review, let alone respond to it. A large part of this issue stems from the inability to quickly detect whether a review is negative or positive. A data product that analyzes reviews for our marketing team automatically could change that.

CUSTOMER SUMMARY

The customers that will be affected by this software mostly comprises of BURGERBURGERBURGER's marketing team. Their department handles our social media and online reputation management. With a new tool allowing them to gauge reviews as positive or negative, they can prepare appropriate actions for responding to the review.

EXISTING SYSTEM ANALYSIS

At this current point in time, our existing system for handling reviews relies on members of marketing fully reading a review before being able to anticipate and craft a response, if it is necessary.

DATA

The data used for this application is a collection of publicly posted reviews on Yelp curated into a public Kaggle dataset.

PROJECT METHODOLOGY

A project of this nature is not breaking new ground and will have clearly definable requirements. This project is also being solo developed by a seasoned machine learning

engineer. As such, the waterfall methodology is the best methodology for the project's implementation. Waterfall methodology provides a linear approach to the software development lifecycle and our approach to it will be outlined below.

1. Requirement gathering
 - a. Speak with members of marketing that oversee online reputation management.
 - b. Develop a requirements document that outlines explicit goals of the software deliverable.
2. Design
 - a. Taking the requirements document from the previous phase, build a design document that outlines how the software will work.
3. Implementation
 - a. Using the design document as a guide, develop the software and prepare it for integration and testing.
4. Integration and testing
 - a. Integrate the new software into a testing environment, and beta test it with select members of marketing.
 - b. Gather feedback from team members and repeat steps 3 and 4, if necessary.
5. Deployment
 - a. With the finished product, make it available for all relevant members of marketing to use.
6. Maintenance
 - a. Be open to troubleshooting issues that arise in the future and updating the model as new and more accurate techniques come out.

PROJECT OUTCOMES

The primary outcome of this project will be a deliverable in the form of a software that the marketing team can use to analyze the sentiment of reviews.

IMPLEMENTATION PLAN

- A. Source public datasets from Kaggle and load them into the python environment.
 - a. All datasets used will be publicly available and contain no sensitive or confidential information. Everything will be anonymized.
- B. Perform data preprocessing.
 - a. Lowercasing

- b. Punctuation and stop word removal.
 - c. Remove any rows with null values
- C. Perform exploratory data analysis.
 - a. Distribution of dataset
 - b. Most common words bar chart
 - c. Most common bigrams and trigrams bar chart
 - d. Length of comments box plot
 - e. Word count box plot
- D. Feature extraction, which will involve transforming the data into a number format that a ML algorithm can understand.
 - a. This can be done with sklearn's CountVectorizer.fit_transform function.
- E. Train the Naïve Bayes classifier on my dataset
- F. Evaluate the performance of the model. This may result in adjusting the model parameters, choosing a new model, preprocessing data differently, etc.
- G. Prediction and application! In this final stage the developer can use the model to identify new bodies of text as carrying positive or negative sentiment.

EVALUATION PLAN

The evaluation of this project will be based on an accuracy score given by the built-in .score function of the MultinomialNB class in the sklearn Python library. Further evaluation of this project will be determined by feedback from the users of the product.

RESOURCES AND COSTS

The table below provides a detailed breakdown of the project resource requirements and costs:

Resource	Description	Cost per unit	Totals
Work laptop	Computer Laptop employee already has	\$0	\$0
Work hours	The estimated number of hours needed to complete task	\$50 X 80 hrs.	\$4000
PyCharm	IDE for running Python scripts -- cost is per user first year	\$249.00 X 1	\$250
Python	Programming language already running on work laptop	\$0	\$0

Maintenance	The annual hours needed to maintain the software	\$50 X 20 hrs.	\$1000
		Total	\$5250

TIMELINE AND MILESTONES

<ul style="list-style-type: none"> - Source public dataset(s) from Kaggle - Perform preprocessing such as lowercasing, normalizing, removal of stop words, and adding a 0-1 sentiment scoring of reviews in dataset. 	July 10, 2023
<ul style="list-style-type: none"> - Perform exploratory data analysis to better understand the distribution of the dataset, and to identify any potential for pitfalls in the data 	July 11, 2023 – July 14, 2023
<ul style="list-style-type: none"> - Split dataset into a training set and a testing set - Train the Naïve Bayes classifier on the training data set. - test the resulting trained model on the test set for accuracy 	July 17, 2023 – July 18, 2023
<ul style="list-style-type: none"> - Develop a simple webapp that allows the user to input a string into a text field to determine if the string is of positive or negative sentiment 	July 19, 2023 – July 20, 2023

Prompt C

Application Files

The application has been stored as a functioning Google Collab. It can be found at the following URL:

<https://colab.research.google.com/drive/1XBKC2QRihLQc2Lf9-N46Z2nzRf4yyxt7>

Prompt D

Post-implementation Report

Project purpose

The purpose of this project is to build a machine learning model that can predict whether a body of text carries a positive or negative sentiment. This model is meant to be used to quickly analyze online reviews of the company called BURGERBURGERBURGER.

Datasets

The dataset used is publicly available yelp reviews. This dataset is anonymized and was provided for free on Kaggle.com. The dataset can be found at the following URL:

<https://www.kaggle.com/datasets/marklvl/sentiment-labelled-sentences-data-set>

This dataset contained more files than was necessary, so ultimately I hosted the one file I needed on my GitHub repository which can be found here:

https://raw.githubusercontent.com/DustinYansberg/Product_Review_Sentiment_Analysis/main/yelp_labelled.csv

The dataset is initially structured such that each row contains a review, and a sentiment score for that review where a score of 1 is positive, and score of 2 is negative. However, some reviews were broken up into multiple columns which created a lot of NaN values.

	col1	col2	col3	col4	Unnamed: 4	Unnamed: 5
0	Wow... Loved this place.	1	NaN	NaN	NaN	NaN
1	Crust is not good.	0	NaN	NaN	NaN	NaN
2	Not tasty and the texture was just nasty.	0	NaN	NaN	NaN	NaN
3	Stopped by during the late May bank holiday of...	1	NaN	NaN	NaN	NaN
4	The selection on the menu was great and so wer...	1	NaN	NaN	NaN	NaN

To correct this, I wrote a function to flatten each row into one column, and then pulled the number from the end of the row and place them in a separate column.

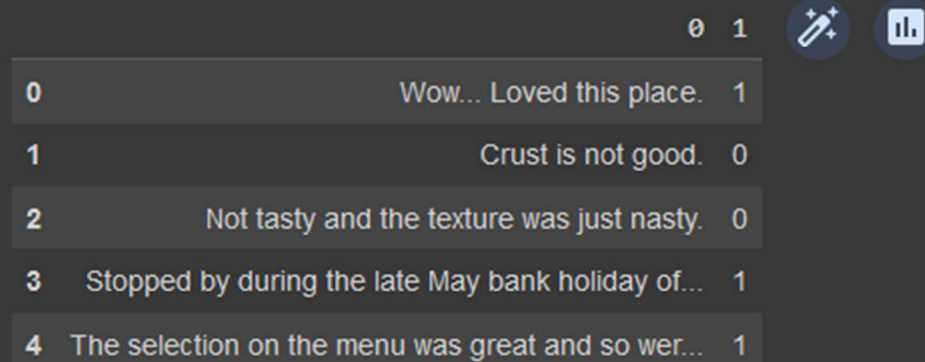
```
# then let's write a function to merge the reviews into the first column.
def merge_text_and_extract_number(row):
    # Drop NA values
    non_na_values = row.dropna()

    # The last value should be number
    number = non_na_values[-1]

    # All prior values should be text
    text = ' '.join(non_na_values[:-1])

    return pd.Series([text, number])
```

```
data = data.apply(merge_text_and_extract_number, axis=1)
data.head()
```



The image shows a Jupyter Notebook interface. At the top, there are two icons: a pencil and a bar chart. Below the icons is a table with two columns, labeled '0' and '1'. The table contains five rows of data. The first row has the index '0' and the text 'Wow... Loved this place.' in column '0' and the value '1' in column '1'. The second row has the index '1' and the text 'Crust is not good.' in column '0' and the value '0' in column '1'. The third row has the index '2' and the text 'Not tasty and the texture was just nasty.' in column '0' and the value '0' in column '1'. The fourth row has the index '3' and the text 'Stopped by during the late May bank holiday of...' in column '0' and the value '1' in column '1'. The fifth row has the index '4' and the text 'The selection on the menu was great and so wer...' in column '0' and the value '1' in column '1'.

	0	1
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1

For ease of use, I labeled the columns as 'review' and 'value.' To see if there were any empty rows, I then ordered the dataset in ascending order on the review column.

```
[7] data.columns = ['review', 'value']
```

```
[8] data.sort_values(by=['review'])
```

	review	value
602	!...THE OWNERS REALLY REALLY need to quit bei...	0
219	#NAME?	1
26	#NAME?	0
896	#NAME?	0
71	#NAME? really good rice all the time.	1
...
934	very slow at seating even with reservation.	0
251	very tough and very short on flavor!	0
81	walked in and the place smelled like an old gr...	0
684	will definitely be back!	1
190	you can watch them preparing the delicious food!)	1

992 rows × 2 columns

I am not sure why the first result of this ordering was a large string, but it did identify some rows that did not contain any notable text. I wrote a function 'clean()' to clean these rows. This function performed the bulk of the remaining preprocessing. It lower cased the text, removed the '#NAME?' strings, removed punctuation, and removed stopwords.

```
9 def clean(comment):  
    comment = comment.lower()  
    comment = re.sub('#name?', '', comment)  
    comment = re.sub('[^a-z A-Z 0-9-]+', '', comment)  
    comment = " ".join([word for word in comment.split() if word not in stopwords.words('english')])  
  
    return comment
```

After this, I ran the function on the entire dataset, and then reset the index values since I knew some indices would be removed entirely.


```
[10] data['review'] = data['review'].apply(clean)
data = data[data['review'].str.strip() != '']
# since we will be removing entire rows where the values are empty after
# removing '#name?' we want to make sure we reset the index values to avoid
# issues while iterating through the reviews in the future.
data = data.reset_index(drop=True)
```

The result was a fully preprocessed dataset that was ready for EDA. A sample of the cleaned dataset is seen below.

data.head(27)

	review	value
0	wow loved place	1
1	crust good	0
2	tasty texture nasty	0
3	stopped late may bank holiday rick steve recom...	1
4	selection menu great prices	1
5	getting angry want damn pho	0
6	honeslty didnt taste fresh	0
7	potatoes like rubber could tell made ahead tim...	0
8	fries great	1

Data product code

Descriptive Method

I used several descriptive methods to analyze my data. My goals here were to:

1. Ensure that the data was evenly distributed between positive and negative reviews.
 - a. I did this by writing code to build a bar chart with two bars. One for the count of negative reviews, and one for the count of positive reviews.

```
fig, ax = plt.subplots()

values = ['positive', 'negative']
counts = [data['value'].value_counts()[1], data['value'].value_counts()[0]]
bar_labels = ['positive', 'negative']
bar_colors = ['tab:green', 'tab:red']

ax.bar(values, counts, label=bar_labels, color=bar_colors)

ax.set_ylabel('Total Count')
ax.set_title('Distribution of positive and negative reviews')
ax.legend(title='Legend')

plt.show()
```

2. Identify any words that may cause confusion for the model.
 - a. I did this by splitting the dataset into two DataFrames and identified the ten most common words for each of them:

```
pos_words = [word for word_list in pos_reviews['word_list'] for word in word_list]
counter = Counter(pos_words)
common_pos_words = counter.most_common(10)

neg_words = [word for word_list in neg_reviews['word_list'] for word in word_list]
counter = Counter(neg_words)
common_neg_words = counter.most_common(10)
```

3. Analyze the length of the reviews to see if any outliers could cause trouble for the model.
 - a. I did this by building a box plot to demonstrate the review length distribution of both reviews.

```
[36] plt.figure(figsize=(16,8))
      plt.subplot(1,2,1)
      sns.boxplot(y=pos_reviews['word_count'])
      plt.ylabel('Word Count of Positive Reviews', labelpad=5)
      plt.subplot(1,2,2)
      sns.boxplot(y=neg_reviews['word_count'])
      plt.ylabel('Word Count of Negative Reviews', labelpad=5)
      plt.show()
```

The results of these EDA's can be found in the 'Effective visualizations and reporting' section of this document.

Predictive Method:

The predictive method I used for this project is a Multinomial Naïve Bayes algorithm. Following my EDA of the dataset, I returned to using the dataset that included all reviews. I first used sklearn's `CountVectorizer().fit_transform` on the review column to transform the reviews into vectors the model could understand. I then assigned the vectorized reviews to variable 'X' and the sentiment values to variable 'y'.

```
[37] cv1 = CountVectorizer()  
      X = cv1.fit_transform(data['review'])  
      y = data['value']
```

Then I split the dataset into training and testing data sets making the test set to be 20% of the dataset and the training set 80% of it.

```
[38] X_train, X_test, y_train, y_test = train_test_split(  
      X, y, test_size=0.20, random_state=80  
      )
```

I then fit the model to the training set.

```
[39] model = MultinomialNB().fit(X_train, y_train)
```

And then got my accuracy score for the model on the test set.

```
[46] model.score(X_test, y_test)  
  
0.8080808080808081
```

Finally, I got the accuracy score on my training set to check for fitting of the model.

```
[42] model.score(X_train, y_train)  
  
0.9532237673830595
```

Hypothesis verification

My original hypothesis stated that by “using a multinomial Naïve Bayes classifier model trained on reviews classified as positive and negative, I can predict with relative accuracy the sentiment of a newly input review.” The model was able to predict the sentiment of the test cases with roughly 80% accuracy. Based on this, my initial hypothesis was correct, though there could be work done to improve the accuracy of the model.

Effective visualizations and reporting

I chose several visualizations in my EDA to understand the dataset better to identify possible issues that may negatively impact the model’s ability to predict accurately.

Dataset Distribution

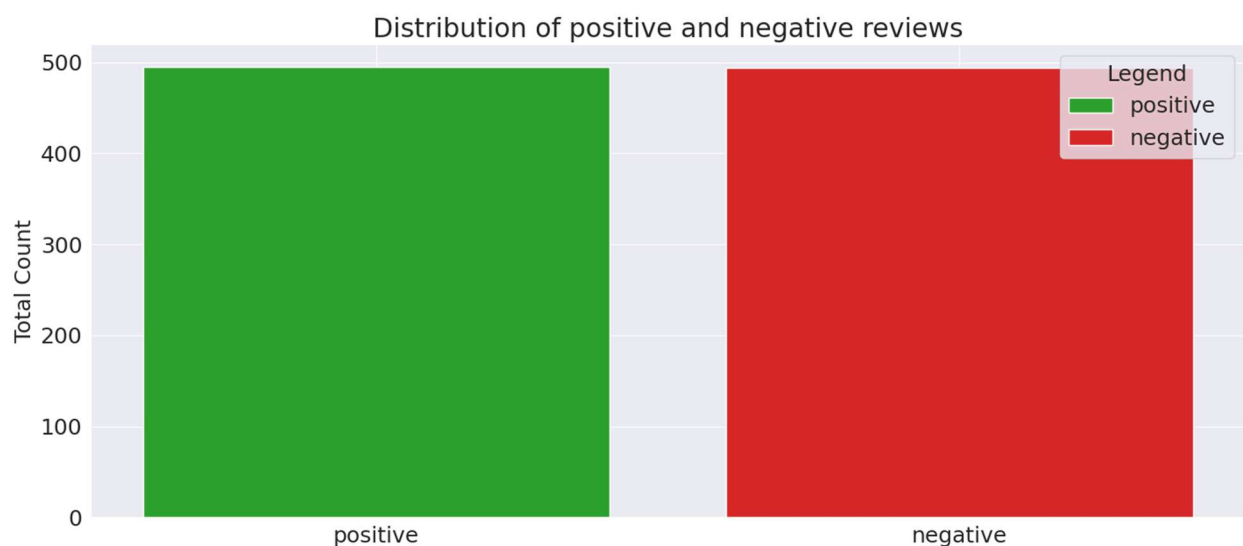


Figure 1: This bar graph shows the distribution of positive and negative reviews. Demonstrated here is the fact that the dataset is evenly distributed between positive and negative reviews.

The dataset is evenly distributed between positive and negative reviews. The dataset is well balanced and now adjustments need to be made for this aspect.

Word Frequency:

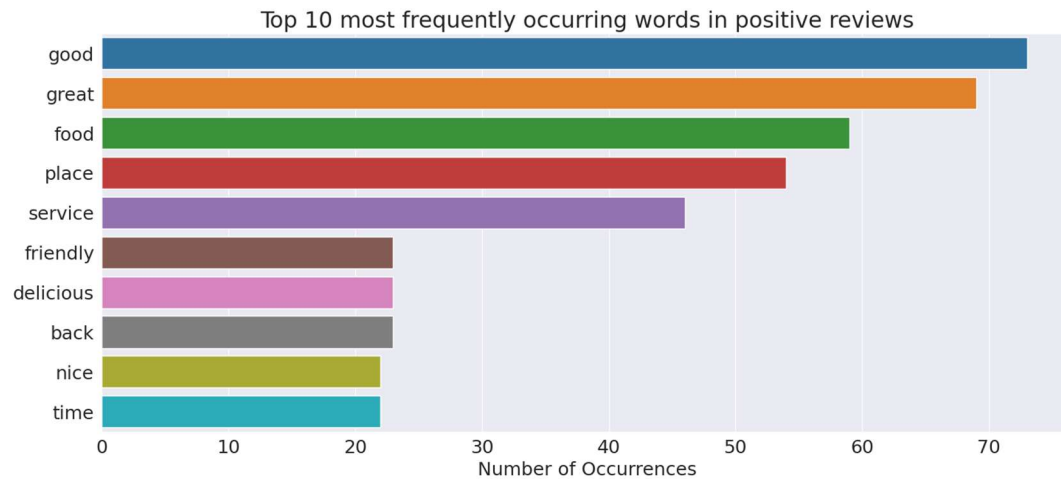


Figure 2: This bar graph shows the ten most frequent words in positive reviews. Notably, food, service, back, and place are all common words found in this figure as well as Figure 3.

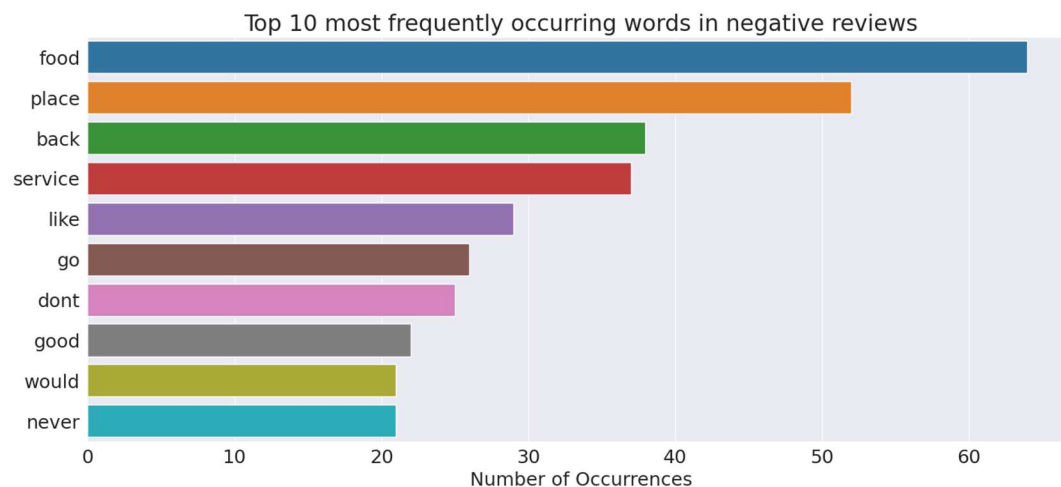


Figure 3: This bar graph shows the ten most frequent words in negative reviews. Notably, food, service, back, and place are all common words found in this figure as well as Figure 2.

From the above bar graphs, it appears that there is some overlap in the top ten words of positive and negative reviews. Notably, "food," "service," "back," and "place" are common in both the negative and positive reviews. In fact, other than the word "back," they are all in the top five most common words. It may be beneficial to treat them as stop words and remove them from the dataset to improve the accuracy of our model.

Word Count Distribution

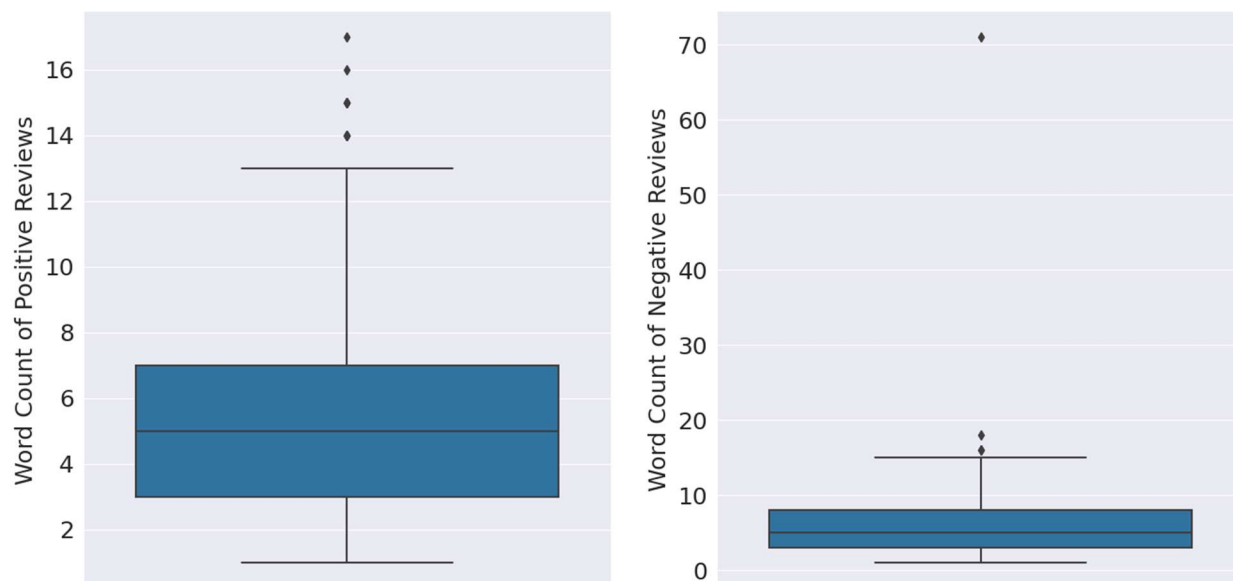


Figure 4: A distribution of word counts between positive (left) and negative (right) reviews.

From the graphs in Figure 4 we can see that most of the reviews, both positive and negative, have less than ten words in them. Both sets of reviews have some outliers in the upper bounds, with negative reviews having one standout review with over 70 words. I don't think this is telling us anything too shocking and would recommend no adjustment to the dataset to compensate for this distribution.

Accuracy analysis

Python's sklearn library includes a function called `score()` which gives the developer an accuracy score of their model on a particular dataset. By splitting the dataset into train and test sets, I was able to train the model on a portion of the dataset and then test it on another portion.

```
[37] cv1 = CountVectorizer()
      X = cv1.fit_transform(data['review'])
      y = data['value']

[38] X_train, X_test, y_train, y_test = train_test_split(
      X, y, test_size=0.20, random_state=80
      )

[39] model = MultinomialNB().fit(X_train, y_train)
```

```
[46] model.score(X_test, y_test)

0.8080808080808081
```

The accuracy of the model on our test set was roughly 80.8%. To analyze the fitting of the model, I then used the `score()` function on the training set as well.

```
[42] model.score(X_train, y_train)

0.9532237673830595
```

Our model's accuracy on the training dataset is ~ 95.32%. This is significantly higher than our model's accuracy on the test dataset. This suggests that our model is overfitted to our training set. Ideally, to avoid overfitting, we would like the scores to be similar. From my EDA, I identified some words that could cause confusion in the model, resulting in a lower accuracy. Removing these words from the dataset entirely could lead to a better accuracy score. Furthermore, our dataset is relatively small with only 1000 reviews. If we were to get a larger dataset, we may be able to positively impact the accuracy of our model. However, for our purposes this accuracy and fitting is acceptable since the reviews being flagged will be read by our marketing team anyway, and they can make executive decisions on what is and isn't important or accurate.

Application testing

Using the waterfall methodology allowed for testing to be done throughout the development lifecycle of this project. Using each phase's output and the next phases input allowed me to test a phase's output as the following phases input before moving on to development of the following phase. However, when testing presented a potential issue from an earlier phases output, the entire development lifecycle needed to return to the phase where the potential issue arose. For instance, when moving the project to Google Colab from my work computer, the source files could no longer be located. This meant that no portion of the project would work until I returned to the beginning of the implementation phase and found a solution for sourcing the project files in a way that Google Colab could understand. Once that was done, every portion of the implementation, integration and testing, and deployment phase had to be retested before I could finalize the deployment phase. In hindsight, unit testing in the final development environment throughout the development life cycle would have made the development process more efficient.

Appendices

Installation Guide

Google Colab is a browser-based interpreter through which Jupyter Notebooks can be run without any software installation beyond your web browser. As this is the case, there is no need for an installation guide. Proceed to the user guide for detailed instructions on running and testing the code.

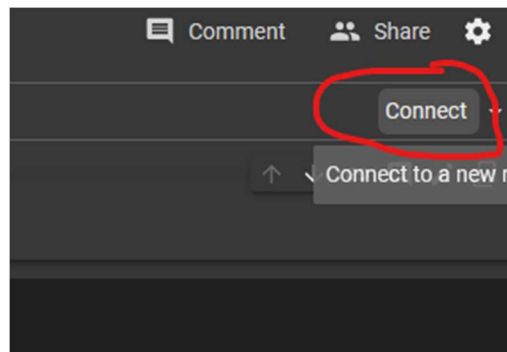
User Guide

The following steps lead to a successful test of the provided application:

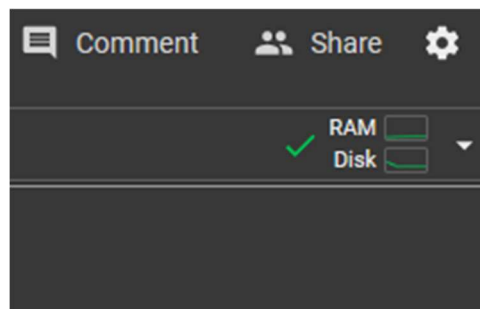
1. In a browser* of your choice, navigate to the listed URL:
 - a. <https://colab.research.google.com/drive/1XBKC2QRihLQc2Lf9-N46Z2nzRf4yyxt7>

*Ensure you are logged into a google account on this browser instance to run the code properly.

2. At the top right of the page:
 - a. select 'Connect' and wait for the process to finish.

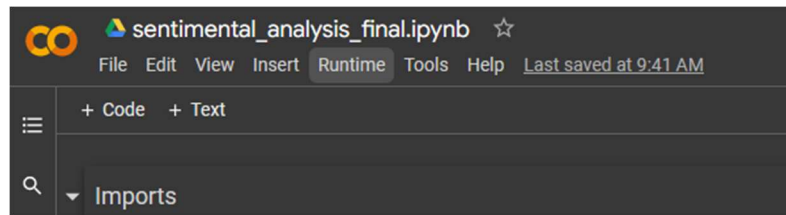


- b. Wait a few moments for the connection to complete. When it is connected, it should look like this:

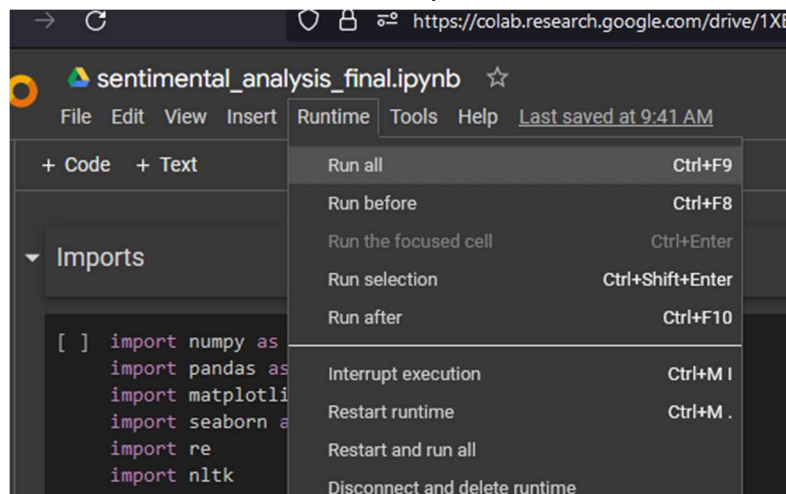


3. With the connection completed, navigate to the top left of the page and select 'Runtime.'

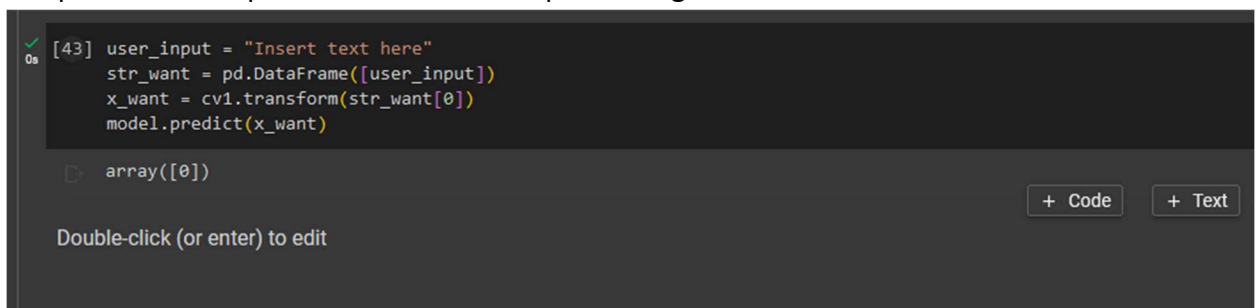
- a. Select Runtime:



- b. Then select Run All from the drop-down menu:



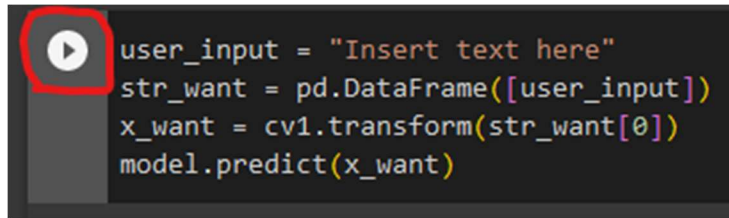
4. Scroll to the bottom of the page and wait for the program to finish initializing. This may take a few moments. It took approximately 25 seconds on my machine. You will know the process is completed when the last input has a green checkmark next to it:



5. Replace "Insert text here" with a body of text of your choice, but do not delete the quotation marks.

6. Press 'Shift + Enter' on your keyboard to run the input.

- a. Alternatively, click on the play button to the left of the input line.



```
user_input = "Insert text here"
str_want = pd.DataFrame([user_input])
x_want = cv1.transform(str_want[0])
model.predict(x_want)
```

- b. The results will output directly below this block of code.

7. Understanding the results:

- a. **Array([0]):** If the output below the input reads “array([0])” then the text is predicted to have negative sentiment.
- b. **Array([1]):** If the output below the input reads “array([1])” then the text is predicted to have positive sentiment.

Summation of Learning Experience

This project was challenging but fun. Having not performed any machine learning projects on my own, there was a lot for me to learn before I could even begin. Sourcing a dataset, preprocessing it, performing exploratory data analysis, and training a model were all brand new concepts to me. However, now that I have completed this task, I feel that a new area of computer science has been made available for me to explore and I look forward to creating personal ML projects on my own in the future as I continue to explore and learn.

References

Kaghazgarian, M. (2018, April 24). *Sentiment labelled sentences data set*. Kaggle.
<https://www.kaggle.com/datasets/marklvl/sentiment-labelled-sentences-data-set>