Logo, company name

Description automatically generated

GROUP ASSIGNMENT

CT107-3-3-TXSA

TEXT ANALYTICS AND SENTIMENT ANALYSIS

APU3F2211CS(DA)

HAND IN DATE: 7 APRIL 2023

GROUP MEMBERS:

* GAVIN OWEN SUSANTO (TP059661)
* MARCELL AGUNG WAHYUDI (TP058650)
* PIPIN SETIAWAN (TP058845)
* YONATHAN VINCENT XAVIER (TP059346)

WEIGHTAGE:

INSTRUCTIONS TO CANDIDATES:

**1 Submit your assignment via Moodle.**

**2** **Students are advised to underpin their answers with the use of references**

**(cited using the 7th Edition of APA Referencing Style).**

**3 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**

**4 Cases of plagiarism will be penalized.**

**5 You must obtain 50% overall to pass this module.**

Contents

[Introduction 3](#_Toc131696801)

[Preprocessing 4](#_Toc131696802)

[Question 1: Unigram Language Model 5](#_Toc131696803)

[Unsmoothed Unigram Language Model 5](#_Toc131696804)

[Smoothed Unigram Language Model 7](#_Toc131696805)

[Question 2: Bigram Language Model 9](#_Toc131696806)

[Unsmoothed Bigram Language Model 9](#_Toc131696807)

[Smoothed Bigram Language Model 10](#_Toc131696808)

[Question 3: Sentence Probability 12](#_Toc131696809)

[1) Manual computation of unigram model sentence probability 12](#_Toc131696810)

[2) Manual computation of bigram model sentence probability 13](#_Toc131696811)

[3) Justification on which language model is more suitable for sentence probability calculation. 15](#_Toc131696812)

[4) Sentence Probability in Python (Using Unigram and Bigram models) 15](#_Toc131696813)

[Question 4: Supervised Text Classification 18](#_Toc131696814)

[4.1 18](#_Toc131696815)

[4.2 19](#_Toc131696816)

# Introduction

This report served as one of the necessary deliverables for the group assignment under the Text Analytics & Sentiment Analysis module. It discusses the use of Python to apply text analytics approaches and methodologies. Jupyter Notebook, a web-based interactive computing notebook environment that enables users to modify and run human-readable documents while explaining the data analysis, was used for the overall implementation. In order to achieve the goals of this project members must understand well what Natural Language Processing (NLP) is, the goal of natural language processing (NLP), a subfield of computer science and artificial intelligence, is to enable computers to comprehend and interpret spoken and written human language in a manner that is comparable to that of humans. Not only to understand but members also need to apply the logic into implementation of model such as language model. The first two sessions of this documentation are implementation establishment a unigram and bigram language model with a text corpus that provided. The code implemented in this documentation will be accompanied by a justification to determine or express which model is the most suitable. For the last section in this documentation, we will be working on the supervised text classification with a csv file that was provided before. To accurately calculate model performance, the NLTK library will be used by team members so they can calculate more accurately and precisely. To perform supervised text classification on the sentiment team members will use SAS Enterprise Miner where the last part will describe the flow process of SAS Text Miner on the model performance measures. The following report includes references for every piece of evidence and thorough citations of all supporting arguments (Lutkevich, 2023).

# Preprocessing

The following code is for all the libraries used for the codes in this assignment.

Graphical user interface, text, application

Description automatically generated

Figure 1: Imported Libraries

Graphical user interface, text, application

Description automatically generated

Figure 2: Imported Txt File

The following code is the pre-processing function used in most of the later functions. The code takes in string as input, then turns all the words into lower case, and removes non-alphanumeric characters, tokenizes the words using nltk tokenize library, and returns the words.

Graphical user interface, text, application

Description automatically generated

# Question 1: Unigram Language Model

## Unsmoothed Unigram Language Model

The code is comprised of two functions. The training part, which trains the unigram, and lastly the prediction function which will predict the probability.

Graphical user interface, text, application, email

Description automatically generated

Figure 3: Unsmoothed Unigram Language Model Code

Then more processing is done so the words printed are unique and does not print the same words, while also removing <s>, and </s>. Output for the code:

Graphical user interface, text, application

Description automatically generated

Figure 4: Smoothed Unigram Language Model Output

## Smoothed Unigram Language Model

Like the Unsmoothed Unigram Language Model, the smoothed unigram language model will first be trained, turning the text file into dictionary, then a prediction (float) will be calculated using the dictionary alongside the text file as input.

Text

Description automatically generated

Figure 5: Smoothed Unigram Language Model Code

Output:

Graphical user interface, text

Description automatically generated

Figure 6: Smoothed Unigram Language Model Code Output

# Question 2: Bigram Language Model

## Unsmoothed Bigram Language Model

To calculate the unsmoothed bigram model, firstly the text is pre-processed by making all words lowercase and removing all non-alphanumeric characters, then the bigram can be generated, the next step is counting the frequency of every bigram, and lastly the calculation of the bigram probability:

Text

Description automatically generated

Figure 7: Unsmoothed Bigram Language Model Code

For the output, a data frame is created using pandas library for easier visualization:

A picture containing text

Description automatically generated

Figure 8: Unsmoothed Bigram Language Model Code Output

## Smoothed Bigram Language Model

To calculate the smoothed bigram model, firstly the text is pre-processed by making all words lowercase and removing all non-alphanumeric characters, then the bigram can be generated, the next step is counting the frequency of every bigram, and lastly the calculation of the bigram probability:

Graphical user interface, text, application

Description automatically generated

Figure 9: Smoothed Bigram Language Model Code

For the output, a data frame is created using pandas library for easier visualization:

A picture containing text

Description automatically generated

Figure 10: Smoothed Bigram Language Model Code Output

# Question 3: Sentence Probability

## Manual computation of unigram model sentence probability

Unigram Model Formula:

Text

Description automatically generated

Figure 11 Unigram Model Formula (Flowers, 2016)

This formula can be used to determine the possible smoothed unigram expressions. With the unigram language model, sentence probability is the sum of the probabilities of each word (p (Wi ) ). Sentence boundaries (such as <s> and </s>) will be dropped from the vocabulary when determining possible whole unigrams, according to language modelling conventions.

Table

Description automatically generated

Figure 12 Manual computation of unigram model

Table

Description automatically generated

Figure 13 Manual computation of unigram model

Table

Description automatically generated

Figure 14 Manual computation of unigram model

Text

Description automatically generated

Figure 15 Manual computation of unigram model

Table above shows the sentence probability of Sentences: <s> He read a book </s>, <s> I read a different book </s>, and <s> He read a book my Danielle </s>. It shows the probability of each token sentence token and also the total probability of the sentences. It shows for the first sentence the total probability is 0.000579. The total probability of the second sentence is 0.000032 and the last sentence is 0.000004.

## Manual computation of bigram model sentence probability

Bigram Model Formula:

Company name

Description automatically generated with low confidence

Figure 16 Bigram Model Formula (Kapadia, 2019)

Bigram model has a formula as above with this formula it can detect whether a word contains a prior word, the algorithm simply counts the instances of bigrams (pairs of two consecutive words) in each text corpus and divides by the quantity of preceding words and also determine whether or not to calculate the likelihood.

A screenshot of a computer

Description automatically generated with low confidence

Figure 17: Manual computation of bigram model

A screenshot of a computer

Description automatically generated with low confidence

Figure 18: Manual computation of bigram model

A screenshot of a computer

Description automatically generated with low confidence

Figure 19: Manual computation of bigram model

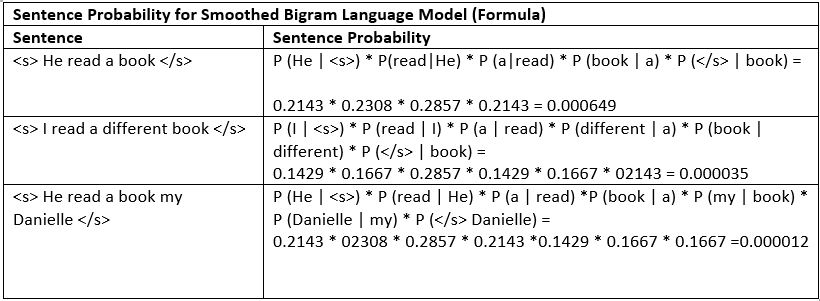


Figure20: Manual computation of bigram model

From the table above we can see that the table shows the probability of sentences such as <s> He read a book </s>, <s> I read a different book </s>, <s> He read a book my Danielle </s> where the table also shows the total probability of the sentences and probability of each token sentence. For example, the total probability of first sentence is 0.000649, the second sentence is 0.000035 and the last sentence is 0.000012.

## Justification on which language model is more suitable for sentence probability calculation.

Table

Description automatically generated

Figure : Unigram and Bigram Probability Comparison

We can conclude from the table above that the bigram language model is more suitable for calculating sentence probabilities. This is since the bigram model has a greater sentence probability than the unigram model when comparing smoothed unigrams and bigrams. Moreover, the bigram language model uses order to determine sentence probabilities, whereas the unigram language model only uses individual words in a phrase. Bigram is more precise because it considers the context of words in phrases, which can significantly improve the accuracy of language models.

## Sentence Probability in Python (Using Unigram and Bigram models)

To calculate sentence probability using unigram and bigram, first the unigram and bigram training functions has to be created:

Graphical user interface, text, application

Description automatically generated

Figure : Unigram and Bigram Training Function

After the training function is completed, the sentence probability of each language model is made:

Graphical user interface, text, application

Description automatically generated

Figure : Sentence Probability Function for Unigram and Bigram

Code Output:

Graphical user interface, text, application

Description automatically generated

Figure : Sentence Probability Code Output

# Question 4: Supervised Text Classification

## 4.1

Text

Description automatically generated

Figure 21 Import Library

The Code above is used to import the required library such as nltk and pandas.

Graphical user interface, text

Description automatically generated

Figure 22 Read Data Set

The code above is used to read the Musical\_Instruments\_Reviews.csv data set.

Text

Description automatically generated

Figure 23 Labeling Predicted Sentiments

The code above is used to label the predicted three level of sentiments which are neutral, positive and negative by using the library SentimentIntensityAnalyzer

A picture containing text

Description automatically generated

Figure 24 Export Predicted Resulting Data Set

The code above is used to export the resulting data set into a new csv file called “Musical\_Instruments\_Reviews\_Result.csv”

Graphical user interface

Description automatically generated with low confidence

Figure 25 Print Polarity Scores

Finally, the code above shows the polarity scores of the data set. Negative sentiment has a polarity of 0.085, neutral sentiments have a polarity of 0.727, and positive sentiment have a polarity of 0.188.

## 4.2

Text

Description automatically generated

Figure 26 Import Library

The code above shows the required library that is required. One of the highlighted library will be the NaiveBayesClassifier from nltk.classify, this library will be used to build a NaiveBayes model

Graphical user interface, text

Description automatically generated

Figure 27 Read Data Set

The code above is used to import the predicted resulting data set from Q4.1

Graphical user interface, text

Description automatically generated with medium confidence

Figure 28 Amount of Reviews

The code above is used to get the amount of positive, negative, and neutral reviews. The data set contain 9070 positive reviews, 996 negative reviews, and 188 neutral reviews.

Text

Description automatically generated

Figure 29 Data Pre-Processing

The code above shows the data pre-processing process. During this phase, work tokenizer is used to break text into word for creating the positive, negative and neutral features. After creating these features, a training and testing data set will be created for the purpose of model building to train the data set.

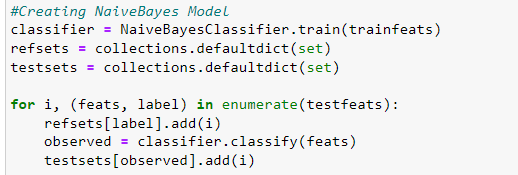


Figure 30 Creating NaiveBayes Model

The code above shows the creation of the NaiveBayes model and data training using the NaiveBayesClassifier library.

Graphical user interface, application, Word

Description automatically generated

Figure 31 Performance Measures

The code above shows the performance measures of the NaïveBayes Model. From the figure above, we can see that the model has an accuracy of 87.9%, precision of 89%, recall of 98%, and F1 Score of 93.6%

# Bibliography

Flowers, L. (2016). *N-gram Language Models*. Retrieved from https://slideplayer.com/slide/7929208/

Kapadia, S. (26 March, 2019). *Language Models: N-Gram*. Retrieved from Towards Data Science: https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9

Lutkevich, B. (January, 2023). *natural language processing (NLP)* . Retrieved from TechTarget: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP