

**School of InfoComm Technology**

**Applied Analytics Assignment**

Diploma in Cybersecurity & Digital Forensics

Diploma in Financial Informatics

Diploma in Information Technology

Year 2/3 (20212/2023), Semester ⅗

**TEAM/INDIVIDUAL ASSIGNMENT**

(40% of AA Module)

**Deadline for Submission:**

**Presentation: 31th July 2022 (Sunday),23:59hrs**

**Report & Code: 14th August 2022 (Sunday),23:59hrs**

|  |  |  |  |
| --- | --- | --- | --- |
| Tutorial Group | : | P02 | |
| Team Number | : | Group 2 | |
| Tutor | : | Mr Toh Ser Chye | |
| Members | : | Student No. | Student Name |
|  |  | **S10226797** | **Lim Kai Chong** |
|  |  | S10222894 | Dominic Lee |
|  |  | S10221858 | Kendric |
|  |  | S10222871 | Ian |
|  |  | S10223426 | Lor Jing Xiang Eugene |

**Penalty for late submission:**

10% of the marks will be deducted every day after the deadline.

**NO** submission will be accepted after 21st August 2022, 23:59 hrs.

Table of Contents

[Introduction 4](#_Toc111347389)

[Problem 4](#_Toc111347390)

[Approach 4](#_Toc111347391)

[Text Data Pre-Processing 5](#_Toc111347392)

[Data Pre-processing 5](#_Toc111347393)

[Cleansing 6](#_Toc111347394)

[Tokenization 8](#_Toc111347395)

[Removing stop words 8](#_Toc111347396)

[Stemming and Lemmatization 9](#_Toc111347397)

[Data Transformation 12](#_Toc111347398)

[Bag of Words Model 12](#_Toc111347399)

[TF-IDF Model 14](#_Toc111347400)

[Text Data Understanding 17](#_Toc111347401)

[Extracting Keywords from TF-IDF Matrix 17](#_Toc111347402)

[Data Formatting 21](#_Toc111347403)

[Data Visualization and Exploration 23](#_Toc111347404)

[Using Word Clouds 23](#_Toc111347405)

[Using a frequency bar graph 24](#_Toc111347406)

[Using feature engineering 24](#_Toc111347407)

[Generating Frequent Itemsets using the Apriori algorithm 26](#_Toc111347408)

[Mining Association Rules - Overall 27](#_Toc111347409)

[Using Minimum Confidence Thresholds 28](#_Toc111347410)

[Using Minimum Lift Thresholds 31](#_Toc111347411)

[Using both Minimum Confidence and Minimum Support Thresholds 32](#_Toc111347412)

[Mining Association Rules – By Category 33](#_Toc111347413)

[Sports 33](#_Toc111347414)

[Tech 36](#_Toc111347415)

[Politics 37](#_Toc111347416)

[Business 38](#_Toc111347417)

[Entertainment 39](#_Toc111347418)

[Summary and Further Improvements 40](#_Toc111347419)

[Summary 40](#_Toc111347420)

[Further Improvements 40](#_Toc111347421)

[Distributional Semantics 40](#_Toc111347422)

[Count matrix vs TF-IDF matrix 41](#_Toc111347423)

[Module Reflection 41](#_Toc111347424)

Applied Analytics Problem 1 Individual Report

**Lim Kai Chong S10226797 P02**

# Introduction

## Problem

Unstructured data, especially in the form of natural language text, can contain large amounts of meaningful information that require the usage of Natural Language Processing (NLP) in order to be extracted. Natural Language Processing is all about leveraging tools, techniques and algorithms to process and understand unstructured language-based data. In this assignment, text analysis will be used on a dataset that contains 2225 documents from the BBC news website corresponding to stories in five tropical areas from 2004-2005, with the intention of extracting meaningful information from this unstructured dataset, through the use of various text analysis techniques and algorithms that will be covered later in the report.

## Approach

To perform text analysis, a methodological approach will be used. Firstly, since the textual data retrieved from the dataset is highly unstructured, the textual data will first need to be pre-processed and parsed into more easy-to-interpret forms. The first part of text data pre-processing involves cleaning and standardization of text to remove unnecessary content from the textual data. This step will involve the use of various techniques such as cleansing, removing tags, tokenizing and removing unnecessary tokens that do not provide semantic meaning (stop words). The next and final part of text data pre-processing involves text normalization, the process of transforming text into a single canonical form. In this step, normalization techniques such as stemming and lemmatizing will be applied. After text data is pre-processed, data can then be transformed and vectorized so that it can be eventually used as input for the classification models for document classification. Vectorizing is the process of encoding text as integers to create feature vectors. To convert text to a numeric vector, techniques such as Bag of Word and Term Frequency-Inverse Document Frequency (TF-IDF) will be considered. After text data is pre-processed and transformed into feature vectors, the text data needs to be analysed and explored before it can be used in the classification models. For exploration and analysis of data, extraction of feature keywords will first need to be performed using the TF-IDF matrix. Only after the feature keywords are extracted, can frequent item sets or sets of keywords be generated using the Apriori algorithm for association rules to be mined. Again, this is so that the text data can be uncovered and understood, so that we are able to make a priori hypotheses and expectations of what we want the data to test when we train our classification models. Once all of the data is successfully pre-processed, transformed and explored, the data will be ready to be used by the classification models. Since the classification models are only built in Problem 2 of this assignment, a summary of the findings from the textual data will be recorded instead, along with further improvements and recommendations that can be considered for future tasks involving text analysis. Figure 1 shows the blueprint of the approach used for Problem 1 of this assignment:

Diagram

Description automatically generated

Figure – Blueprint of the approach used for Problem 1

# Text Data Pre-Processing

## Data Pre-processing

Data pre-processing refers to manipulation or dropping of data before it is used in order to enhance performance. In data mining and text analytics, data pre-processing is an essential part because removing irrelevant components and noise helps the classification models to perform more complex analyses including learning patterns and extracting information. Before any data pre-processing can be performed, data will first need to be loaded. Using the Python pandas library, we are able to load the data and store it as a data frame, to perform simple descriptive analytics using head() and info().

Graphical user interface, text

Description automatically generated

Figure – Loading the text data and performing simple descriptive analytics

Next, we check the textual data to account for null values:

Text

Description automatically generated

Figure – Checking for nulls in textual data before data pre-processing

### Cleansing

Following the approach mentioned, we first pre-process the data by performing cleansing. By defining a function pre\_process(text), we are able to use this function whenever we require a text corpus to be cleansed. In the function, the text parameter passed in will be cleansed by first reducing all characters to lowercase form. Next, using the Regular Expressions re Python module, we can remove tags, special characters, digits and punctuations from the text so that the text can be tokenized. Once these operations are done, we return the text and we create a new column ‘text\_clean’ in the data frame, which is the cleansed version of the initial text data.

Graphical user interface, text, application

Description automatically generated

Figure – Process of cleansing the initial text data

### Tokenization

Once the text is cleansed, i.e. the text no longer has special characters or punctuations, we can proceed to tokenize our text. The process of tokenization converts text documents into more meaningful components called tokens, which are independent textual components meant to separate words from given sentences or input. This helps to break raw text documents into tokens, which can be used in sequential techniques to better understand the content of the text document.

Text

Description automatically generated

A picture containing text

Description automatically generated

Figure – Process of tokenizing the text data

Figure 5 shows the process of tokenizing the text data. To perform tokenizing, we will create a function called tokenize(text) that will tokenize the text parameter passed into the function and return a tokenized text. Similarly, the tokenizing process will be done using the Regular Expressions re Python module. The use of the ‘\W+’ regular expression indicates that it will split whenever it sees one or more non-word characters in the text, so that words in a sentence that are separated by spaces will be split into individual tokens. Once the text is tokenized, we create a new column in the data frame called ‘text\_tokenized’, where our tokenized text is stored.

### Removing stop words

Once our text is separated into tokens, we can proceed to remove stop words. Stop words are unnecessary tokens in our text that do not provide any semantic meaning. As such, they can be removed to limit the number of tokens to give more focus to important and relevant information in the text data.

A picture containing graphical user interface

Description automatically generated

Figure – Process of loading stop words to remove stop words from text data

Figure 6 shows the process of removing stop words from the text data. To remove stop words from the text data, we first create a function to load and return a list of stop words given to us under “stopwords.txt”. Once we have loaded a list of stop words, we can then pass the list of stop words through our text and remove any tokens in our text that matches any stop words stored in the ‘stopwords’ list. Similarly, once the entire process is done, we can store the text data containing no stop words in a new column called ‘text\_nostop’ in the data frame. After cleansing, tokenizing and removing stop words, we can proceed with text normalization, where techniques such as stemming and lemmatization will be used to normalize the text data.

### Stemming and Lemmatization

Text normalization is the process of reducing inflectional forms and sometimes derivationally related forms of a word to a common base form. In any NLP problem, text normalization is an important step in the data pre-processing phase because by converting tokens to their base form, the number of unique tokens in the text will be reduced, and variations in the text will be removed. This will in turn help to normalize the data and reduce the amount of unnecessary information that will need to be learnt, resulting in better quality data with less overall noise, and potentially better accuracies in the models that the data will eventually be fed to. In this section, two text normalization techniques will be explored, namely Stemming and Lemmatization.

To understand the process of stemming, we would need to understand what ***word stems*** are and what the word ***stem*** represents. ***Word stems*** are often known as the base form of a word, and we can create new words by attaching affixes to them. This process is known as ***inflection***. On the contrary, the reverse of this is obtaining the base form of a word from its inflected form, and this process is known as ***stemming***. Consider the word “HOP”, you can add affixes to it and form several new words like “HOPS” or “HOPPING”. Conversely, removing the affixes will give you the base word, or the word stem, “HOP”. In text analytics, stemming helps to standardize words to their base stem irrespective of their inflections, which helps many applications classify text, or retrieve information from text. To perform stemming, a stemmer can be implemented from the Natural Language Toolkit (NLTK) package. In this case, we will use the Porter stemmer, which is a popular stemmer offered in the NLTK package.

Graphical user interface, text

Description automatically generated

Figure – Process of stemming the text data

Using the Porter stemmer implementation from the NLTK package, we are able to create a function stemming(text) that stems each token in the text passed in through the text parameter to its base form, or word stem, and return the stemmed text. After stemming the text, we add a new column ‘text\_stemmed’ to the data frame and store the stemmed text data there. With reference to Figure 7, we can observe that the word ‘hands’, the third token in the first row (row 0) of ‘text\_nostop’ was reduced to its base form ‘hand’ after stemming was performed.

Next, we will perform lemmatization on the text data as well. The process of lemmatization is very similar to stemming, where affixes are removed to get the base form of a word. However, in this case, the base form is also known as the ***root word*** but **not the root stem**. The difference between the two is that **the root stem may not always be a lexicographically correct word**, i.e., it may not be present in the dictionary but **the root word, or the lemma, will always be present in the dictionary**. In terms of operations, the lemmatization process is considerably slower than the stemming process because the affix of a word is only removed if the lemma is present in the dictionary, creating an additional step in the process. For lemmatization, we will use a robust lemmatization module that uses WordNet and the word’s semantics to get the root word or lemma. Similar to Porter stemmer, this module can be imported from the NLTK package.

Text

Description automatically generated

Figure – Process of lemmatizing the text data

As mentioned, using the WordNet lemmatization module imported from the NLTK package, we are able to create a function lemmatizing(text) to lemmatize each token in the text passed in through the text parameter to its root word, or lemma, and return the lemmatized text. After lemmatizing the text, we add a new column ‘text\_lemmatized’ to the data frame and store the lemmatized text data there. With reference to Figures 7 and 8, we can see that the word ‘future’, the second token in the first row (row 0) of ‘text\_nostop’ was reduced to its word stem ‘futur’ in ‘text\_stemmed’ but was retained as ‘future’, its root word in ‘text\_lemmatized’, depicting the difference in the process of both normalization techniques, where although ‘futur’ is not a lexicographically correct word, it was still reduced to its base form, or word stem. Between lemmatization and stemming, there is currently not a universally preferred normalization technique. While stemming is straightforward and faster to implement, lemmatizing produces slightly more accurate results as it produces real, dictionary words. From a qualitative point of view, since both techniques are already implemented, and lemmatization has been known to be the slightly more accurate technique between the two, we will use the lemmatized text ‘text\_lemmatized’ as the text data for which data transformation will be performed.

## Data Transformation

Data Transformation will be performed using traditional vector space models for unstructured data such as the ***Bag of Words*** model and the ***Term Frequency-Inverse Document Frequency******(TF-IDF)*** model. A vector space model is a mathematical model to represent unstructured data or text as numeric vectors, such that each dimension of the vector is a special feature/attribute. Essentially, in this section, we are **transforming the unstructured text into numeric vectors** to achieve better text understanding through techniques that will be covered later in the report.

### Bag of Words Model

The Bag of Words model represents each text document as a numeric vector where each dimension is a specific word from the text data and the value represents **the occurrence of the specific word in the document (denoted by 1 or 0)**. As such, each vector is an array with the length of the total number of extracted word features, and how each feature is extracted from text to a numeric vector is done through a process known as ***vectorization***.

Graphical user interface, text, application, email

Description automatically generated

Figure – Process of vectorizing features and creating a feature matrix

Figure 9 shows the process of vectorizing features and creating a feature matrix, wherein the feature vectors will be stored. Using the sklearn.feature\_extraction module, we can import CountVectorizer and instantiate a count vectorizer object under count\_vect. In the CountVectorizer() constructor, we specify a parameter max\_df=0.15 to state that words that appear in 15% or more of documents need to be removed because words that occur too frequently are not influential and therefore not relevant. Additionally, we pass in the ‘stopwords’ list again just for good measure to remove any stopwords that could still exist in the text data. Lastly, we set the maximum number of features to 5000 to limit the transformations to only 5000 features instead of the entire corpus. It can be noted that such a parameter is optional in the vectorization process. After the feature matrix is created, we can see that the shape of the feature matrix is 2225 by 5000. This suggests that the feature matrix contains records of 2225 BBC news stories and 5000 feature words in the form of numeric vectors. Thereafter, words\_freq maps each word/feature to its respective frequency of occurrence obtained from the feature matrix, stores the key/value pairs in a dictionary and sorts the dictionary in descending order of frequency of occurrence.

Text

Description automatically generated Text

Description automatically generated

Figures 10 and 11 – Frequency of words

We can see that now the feature words are interpretable, as the words\_freq dictionary is able to display words with the highest frequencies and lowest frequencies. With reference to Figures 10 and 11, we can see that the word ‘film’ has the highest frequency of 1148 and the word ‘lasting’ along with 9 other words, has the lowest frequency of 12.

The Bag of Words feature matrix:

Table

Description automatically generated with low confidence

Figure 11 – The Bag of Words feature matrix

### TF-IDF Model

There are some problems that might arise with the Bag of Words model, especially when used on large datasets. Since the feature vectors are based on ***absolute term frequencies***, there might be cases where some terms that occur frequently across all documents overshadow other terms in the feature set. Specifically, this affects words that do not occur as frequently but are more interesting as features to identify specific categories or the semantic of a document. As such, to reflect the ***relevance*** of a word to a document, TF-IDF vectorization should be considered instead. The TF-IDF model is a combination of two metrics, ***term frequency (TF)*** and ***inverse document frequency (IDF)****.* Term frequency in any document vector is denoted by the raw frequency value of that term in a particular document. Conversely, as its name implies, inverse document frequency is the inverse of the document frequency of each term. Therefore, the TF-IDF can be computed by multiplying these two values, and is denoted as follows:

Graphical user interface, application

Description automatically generated

Figure 12 – Formula to compute TF-IDF score of a feature

Graphical user interface, text, application, email

Description automatically generated

Figure 13 – Obtaining TF-IDF based feature vectors

With reference to Figure 13, the TF-IDF based feature vectors can be obtained from TfidfTransformer, an implementation that can be imported from the sklearn.feature\_extraction module. Since feature extraction and vectorization has already been implemented and stored in ‘text\_counts’ when we instantiated the Bag of Words model, there is no need to vectorize the text again. However, it is worth noting that the TfidfVectorizer by Scikit-Learn can enable direct computation of the TF-IDF vectors by internally computing the term frequencies and inverse document frequencies of the raw document inputs, eliminating the need to use CountVectorizer to compute the term frequencies based on the Bag of Words model. After the TF-IDF based feature vectors are obtained, we can see that we are now able to also access the IDF of all feature vectors, through the use of .idf\_, since the IDFs of all feature vectors are now computed.

From this, we are able to obtain the TF-IDF matrix and the features with the lowest/highest TF-IDF scores:

Calendar

Description automatically generated

Figure 14 – Generating the TF-IDF matrix and obtaining features with highest/lowest TF-IDF scores

With this, the text data has been successfully transformed into feature vectors, and from obtaining the TF-IDF feature matrix, we will be able to perform text data understanding in the next section.

# Text Data Understanding

Text Data Understanding is the last but one of the most crucial steps in Machine Learning before building the models. In order to generate any useful a priori hypotheses that we would like our data to test, we would first need to be familiar with the text data to have salient a priori notions about what the data might uncover. In such cases, exploratory methods and further analysis needs to be conducted to allow understanding of:

* Individual Features/Extracted Keywords
* Associations or correlation among these extracted keywords/variables

As such, with the information that we have obtained from previous sections, we can perform text data understanding by extracting keywords and using various techniques such as the ***Apriori algorithm*** and ***association rule mining*** to further explore the text data before the data is used for the models.

## Extracting Keywords from TF-IDF Matrix

With the TF-IDF matrix that we have generated from the previous section, we can extract the keywords from its numeric vector representation and store the extracted keywords in a new dataset to perform other forms of data visualizations and rule mining. Text

Description automatically generated

Figure 15 – Mapping the first news story to the TF-IDF numeric vector row representation

To extract keywords from the TF-IDF matrix, we must first map every word in each news story to its respective TF-IDF numeric vector representations. In the cell of code shown in Figure 15, we use a common index of 0 to list out the first news story in its unstructured form and the first row of the TF-IDF matrix.

Thereafter, the ‘temp’ dictionary sorts the row of numeric vectors from row 1 of the TF-IDF matrix in descending order of their TF-IDF score to determine the feature numbers of the keywords with the highest TF-IDF scores for the first news story:

Graphical user interface

Description automatically generated with low confidence

Figure 16 – Obtaining a data frame containing the feature number and TF-IDF score for row 1 of TF-IDF matrix

This allows us to generate the top keywords that are the most relevant for each news story across the entire text data. Next, with the intention of extracting the 5 most relevant keywords from news story 1, we use a for loop to loop through the feature numbers of the 5 most relevant keywords for news story 1 to map the feature numbers to their respective feature names:

Text

Description automatically generated

Figure 17 – Mapping the top 5 feature numbers in term of TF-IDF scores to feature names



Figure 18 – The top 5 keywords from news story 1 mapped to their TF-IDF scores

With this, we can see that from Figure 18, we have successfully managed to map and extract the top 5 keywords for news story 1 to their respective numeric vectors and TF-IDF scores. With reference to Figure 16 as well, it can be observed that the feature number with the top performing TF-IDF score of 0.457, is correctly mapped to the feature name/keyword ‘tv’, with the same TF-IDF score.

Graphical user interface, text, application, email

Description automatically generated

Figure 19 – Process of extracting the top 5 most relevant keywords from the entire text data

With reference to Figure 19, now that we have extracted the top 5 most relevant keywords for the first news story, we loop this process through every news story in our text data, to extract the top 5 most relevant keywords from each news story. After the process of extracting the top 5 most relevant keywords from all news stories is completed, we can see that there is a total of **2225** extracted keywords.

Text, letter

Description automatically generated

Figure 20 – Storing the extracted keywords into a separate csv file

With reference to Figure 20, since ‘results’ is a list of data frames stored as a list object, it cannot be directly exported as a csv file. As such, the extracted keywords is first stored in the current text data frame by adding a new column ‘keywords’, and then this column is retrieved through the variable ‘extracted\_keywords’, which can then be used to export the extracted keywords into a new csv file ‘keywords.csv’. Now that we have managed to obtain a text data file containing all of the extracted keywords, we can begin to perform rule mining and other data visualisation techniques to understand the text data.

## Data Formatting

Data Formatting is a crucial step in text data understanding because text stored in data frames must be precisely formatted before it can be used for further exploration and analysis.

Text

Description automatically generated

Figure 21 – Reading the csv file of the extracted keywords

As shown in Figure 21, upon reading the extracted keywords file that was exported in the previous section, we can see that the data frame is not properly formatted and thus, cannot be properly used yet. Therefore, in order for further exploration and understanding of the keywords, the TF-IDF scores that are mapped to each keyword need to be removed. This can be done by using the Regular Expressions re Python module and the cleansing text function pre\_process(text) we defined earlier in our data pre-processing phase:

Graphical user interface, text, email

Description automatically generated

Figure 22 – Formatting the keywords to remove the TF-IDF scores from the keywords dictionary pair

As seen from the new column ‘formatted\_keywords’ in Figure 22, the keywords are no longer mapped to a TF-IDF score, and now exist as an entire cleansed text corpus. Next, to obtain each keyword as an independent variable, we would need to separate the keywords into individual entities that can be individually appended to a keyword list, so that the keywords can be stored in the data frame as independent objects:

Text

Description automatically generated with low confidence

Figure 23 – Keywords are now independently stored as variables in the data frame

As seen in Figure 23, the keywords are now independent variables stored in the data frame. To obtain a data frame that purely contains the keywords, we can use the .iloc[] method provided by the Python Pandas library to index our data frame and remove the unnecessary rows/columns.

Graphical user interface, application, table

Description automatically generated

Figure 24 – Each keyword is now properly formatted and ready to be used for future exploration

In Figure 24, after performing a series of .iloc[] indexing operations, we can see that the data frame is now perfectly formatted and filled with only independent keywords in their own individual columns. Therefore, we can now start to explore and analyse the extracted keywords.

## Data Visualization and Exploration

### Using Word Clouds

The first step to understanding textual data, or any form of data that is already processed, is to visualize the data through the use of various data visualization techniques. A good way to visualize textual data is through ***Word Clouds***. A word cloud is a visual representation of words, used to highlight popular words and phrases based on frequency and relevance. Word clouds are effective in providing quick and simple visual insights that can lead to more in-depth analyses. With the use of the Python WordCloud library, we are able to create word clouds for the extracted keywords data frame that we have:

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 25 – Word cloud used to visualise our extracted keywords in an interpretable form

From the word cloud shown in Figure 25, we are able to finally note some observations from the extracted keywords data. For example, Figure 25 suggests that the keywords ‘election, ‘film’ and ‘blair’ are the 3 most popular keywords, as they are depicted to be the 3 largest keywords in the word cloud, which already is able to provide a good amount of information of the type of news documents in the textual data.

### Using a frequency bar graph

Another effective way to visualize unstructured data and observe the frequencies of the most used keywords is by using a frequency bar graph. Using the matplotlib visualization library in Python and the .value\_counts() pandas function that returns a series containing counts of unique values, we are able to obtain and visualize the frequencies of the unique keywords by plotting a graph: Chart, bar chart

Description automatically generated

Figure 26 – Bar graph depicting the frequency of the top 50 most used keywords

From Figure 26, we can see that the most used keyword by far is the word ‘film’, being used over 100 times, followed by words like ‘election’ and ‘blair’, being used 60 times or less.

### Using feature engineering

Lastly, feature engineering can be an efficient way in generating new features, so that data can be more interpretable. While usually implemented to provide more information for better training processes in machine learning models, feature engineering can also be used in the data visualization phase to see if there are any interesting features that can be engineering or extracted from raw data, to be used in predictive modelling. In this case, feature engineering will be used to see if the length of a given text has any correlation to its category:

A picture containing text

Description automatically generated

Figure 27 – Adding a new column that stores the length of the news story

With reference to Figure 27, to first test this hypothesis, a new column that stores the length of each news story will be made.

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

Chart, histogram

Description automatically generated Chart, histogram

Description automatically generated Chart, histogram

Description automatically generated

Figure 28 – Histograms plotting the distribution of length of news stories

Thereafter, as seen in Figure 28, histograms can be plotted to show the distribution of length of news stories. As seen from the histograms plotted, most categories have a mode length of roughly 0-2000 words in their news stories. In the case of entertainment and tech, some outliers exist, with a word length of between 7000-8000 words.

## Generating Frequent Itemsets using the Apriori algorithm

Association rules and its algorithms seek to uncover associations and patterns among variables and take the form “If *antecedent,* then *consequent*,” along with measures such as ***support*** and ***confidence*** to deduce the strength of an association rule. However, the problem that awaits such an algorithm is the problem of *dimensionality*, that is, the number of possible association rules grows exponentially given the number of attributes. Consider that in market basket analysis, which is a typical application of the association rules algorithm, there may be thousands of items, each with thousands of combinations, making extracting associations between the items a daunting task. Therefore, it is essential to implement the ***a priori algorithm*** before mining association rules, which takes advantage of the lattice structure of the item sets to reduce the search problem to a more manageable and interpretable size. This is done through the ***a priori principle***, which states that “If an item set is frequent, then all its subset items will be frequent”. Therefore, by leveraging such a principle, the a priori algorithm helps to extract frequent item sets by using the concept of a ***minimum support threshold***. The support for an association rule *X* ⇒ *Y*  is the proportion of transactions in the set of transactions that contain both *X* and Y. As a result, by setting an appropriate support threshold, frequent item sets can be generated because only an item set that is frequently occurring will have a high support value.

Graphical user interface, text, application, email

Description automatically generated

Figure 29 – Applying the a priori algorithm and setting a minimum support threshold of 1%

Figure 29 shows the process of applying the a priori algorithm and setting a minimum support threshold of 1% on the one-hot encoded NumPy array containing the extracted keywords. By using the describe() method provided by the Python pandas library, we see that a total of 47 items and item sets out of 2962 keywords satisfied a minimum support threshold of 1%. However, in order to mine more interesting association rules, 47 items/item sets are not enough. Therefore, we will decrease the minimum support threshold to 0.5%, to see if more items/item sets can be generated:

Graphical user interface, text, application, email

Description automatically generated

Figure 30 – Applying the a priori algorithm and reducing the minimum support threshold to 0.5%

As seen from Figure 30, by reducing the minimum support threshold to 0.5%, we are able to generate 175 items/item sets, which would likely be a sufficient amount to generate a good number of association rules.

## Mining Association Rules - Overall

As mentioned, association analysis is a useful way to measure the strength of co-occurrence between one item and another. In text data understanding, association rules mining is an unsupervised learning technique conducted to discover hidden and usable patterns in the co-occurrences of items, to obtain greater understanding of the text data and relationship between items, through the expression of easily recognizable rules. To understand the process of how association rule mining works, one must first understand the concepts of ***confidence*** and ***lift.*** The confidence of a rule *X* ⇒ *Y*  is a measure of the accuracy of the rule as determined by the percentage of transactions within the set of transactions containing *X*  that also contain *Y* . In other words, the confidence of a rule *X* ⇒ *Y* is denoted by:

Text

Description automatically generated with medium confidence

Equation 1 – Confidence of rule X->Y

Therefore, in regard to Equation 1, we can say that if out of 6 transactions, 5 of them contain apples and 4 of them contain both apples and oranges, the confidence of the rule {apple} -> {orange} would be 4/5 = 80%.

On the other hand, the lift of a rule is a measure of how much more often the antecedent and the consequent of a rule *X* ⇒ *Y* occur together than we would expect if they were statistically independent. In other words**, the higher the lift of a rule *X* ⇒ *Y ,* the more interesting the rule is**, and such a metric can be denoted by:

Text, letter

Description automatically generated

Equation 2 – Lift of rule X->Y

Therefore, in regard to Equation 2, we can say that if out of 6 transactions, 4 of them contain apples, 2 of them contain oranges and 2 of them contain both apples and oranges, the lift of the rule {apple} -> {orange} would be (2/6) / (4/6 \* 2/6) = 1.5. Hence, using the measures of lift and confidence, and the concept of minimum thresholds, we are able to mine rules with strong associations for a given set of transactions.

### Using Minimum Confidence Thresholds

One way to generate association rules is to use minimum confidence thresholds as the measure of strength between association rules. In this assignment, a minimum confidence threshold of 30% is initially used.

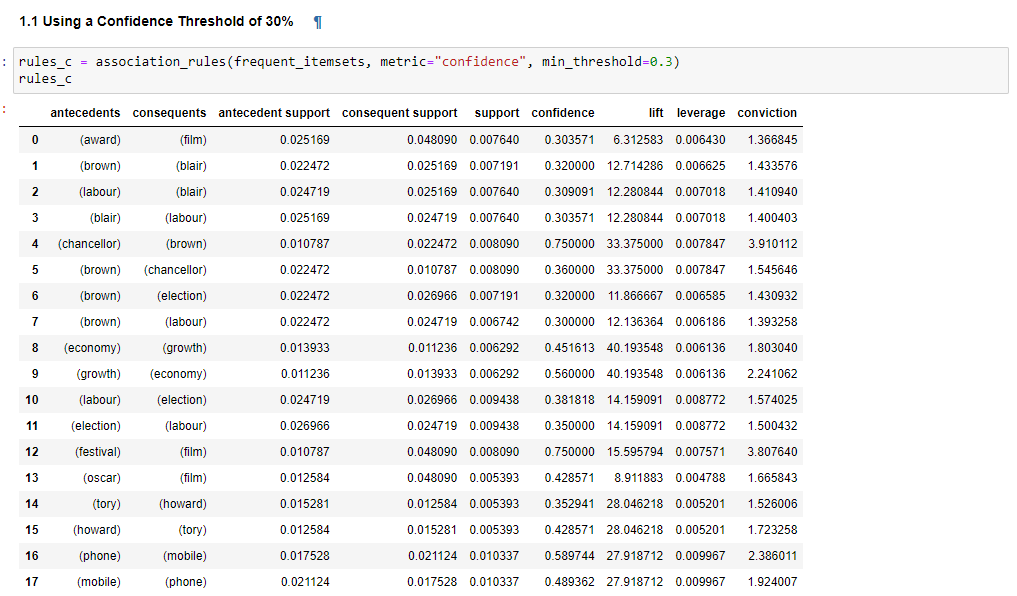


Figure 31 – Using a minimum confidence threshold of 30% to generate association rules

However, as seen from Figure 31, upon using a minimum confidence threshold of 30%, we can see that there are 17 different sets of association rules that satisfy the aforementioned threshold. Thus, with the intention of reducing the scope to more highly associated rules, we will try increasing the minimum confidence threshold to 40%:

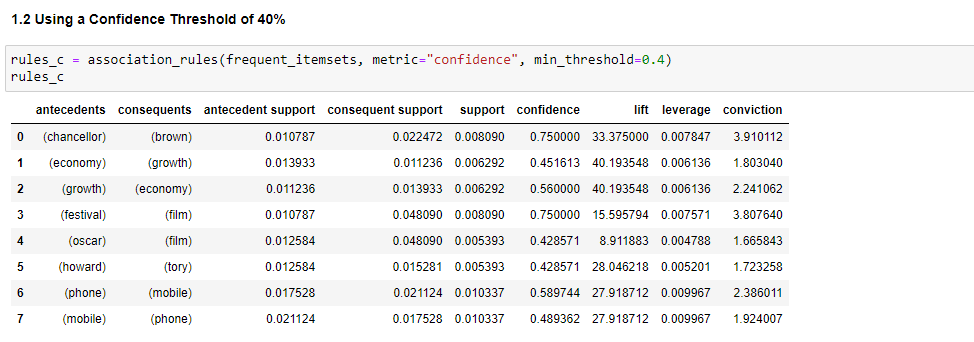


Figure 32 – Using a minimum confidence threshold of 40% to generate association rules

Figure 32 shows the association rules generated upon using a minimum confidence threshold of 40%. From the results shown, we can see that the number of rules have decreased to 8 rules. In addition, we can interpret that all of the above rules exist such that minimally, 40% of the time the antecedent occurs in the news story, the consequent will also occur in the same news story. In the case of the rules {chancellor} -> {brown} and {festival} -> {film}, the Figure also suggests that with a rule confidence of 75%, there is a 75% chance that the word ‘brown’ would appear in a news story that contains the word ‘chancellor’ and a 75% chance that the word ‘film’ will occur in the same news story that contains the word ‘festival’. Apart from this, some other valuable insights were:

1. The rules {economy} -> {growth} and {growth} -> {economy} has the highest lift value of 40, suggesting that the words/items ‘economy’ and ‘growth’ occur together 40 times more frequently than expected if the items were statistically independent. This may refer to the field of “Business” where the term “economic growth” is highly used as a phrase to describe the state of the economy at the given time
   1. This would also explain the high lift scores of these items as the words “economy” and “growth” can be assumed to be highly dependent on each other, especially if the economy was in a state of growth at the time
2. The itemset ‘mobile, phone’ and ‘phone, mobile’ have the highest support values of 1.03%, suggesting that the words ‘mobile’ and ‘phone’ has the highest co-occurrence in the entire collection of news stories, suggesting that mobile phones could be hot topics in the years 2004-2005
3. The words ‘film’ and ‘festival’ have a high likelihood (75%) of co-occurrence, and likely suggest that they occur most frequently in the ‘Entertainment’ industry, where there is frequent discussion of film festivals
4. The words ‘chancellor’ and ‘brown’ also have a high likelihood (75%) of co-occurrence, and likely refers to the former United Kingdom Prime Minister Chancellor Brown, suggesting the news stories that contain these words are highly relevant in the ‘Politics’ category

Chart, box and whisker chart

Description automatically generated

Figure 33 – Boxplot depicting the confidence distribution for the frequent itemsets

With reference to Figure 33, the boxplot for the confidence distribution of the rules that satisfy a minimum confidence threshold of 40% shows that on average, the rules have a mean confidence value of 52.5%, and most of the rules fall within the interquartile range of between 45% to 62.5%.

### Using Minimum Lift Thresholds

On the other hand, another way to generate association rules is to use a minimum lift threshold as the measure of interestingness between association rules. It can be noted that an appropriate lift threshold should always have a value higher than 1, as a lift of 1 suggests that the items are statistically independent and do not provide any interestingness. In the case of this assignment, a minimum lift threshold of 2 was initially used:

Table

Description automatically generated

Figure 34 – Using a minimum lift threshold of 2 to generate association rules

However, as seen from Figure 34, upon using a minimum lift threshold of 2, we can see that 24 rules were generated. From the figure, we can also see that the mean lift value is 18, suggesting that the minimum lift threshold should be increased to a value slightly larger than 18, to truly generate the most interesting rules.

Table

Description automatically generated

Figure 35 – Using a minimum lift threshold of 20 to generate association rules

With reference to Figure 35, a new minimum lift threshold of 20 was specified, that is, all antecedents and consequents in the rules that satisfy this threshold co-occur at least 20 times more than expected than if the antecedents and consequents were statistically independent. Some valuable insights obtained were:

1. The most interesting rules with the largest lift values of 40 belong to the rules {economy} -> {growth} and {growth} -> {economy}.
   1. As mentioned in the previous section, this suggests that the words/items ‘economy’ and ‘growth’ occur together 40 times more frequently than expected if the items were statistically independent.
   2. This is also a testament to how dependent the words/items ‘economy’ and ‘growth’ are on each other at the given time, potentially indicating the state of the economy at the time

### Using both Minimum Confidence and Minimum Support Thresholds

Lastly, another method to generate association rules is to use both minimum confidence and minimum support thresholds. By using this method, we get to mine the association rules that not only have the strongest co-occurrence and relationship, but also the rules that are the most interesting. Following the previous two sections, a minimum confidence threshold of 40% was used, and a minimum lift value of 20 was used:

A picture containing graphical user interface

Description automatically generated

Figure 36 – Using a minimum confidence threshold of 40% and a minimum lift threshold of 20 to generate rules

As seen from Figure 36 , upon using a minimum confidence threshold of 40% and a minimum lift threshold of 20, we can see that the same few rules such as {chancellor} -> {brown} and {economy} -> {growth} appear. This suggests that these rules are some of the rules with the strongest co-occurrence and the most interesting relationship out of all of the frequent item sets generated using the a priori algorithm. Some interesting insights are:

1. Only the one-sided rules {chancellor} -> {brown} and {howard} -> {tony} satisfy the minimum lift and confidence thresholds, but both sides of the rules {economy} -> {growth} and {mobile} -> {phone} satisfy the minimum lift and confidence thresholds
2. Even if the antecedents and consequents were flipped, the rules would still have the same lift. Therefore, for these one-sided rules, it can be assumed that their opposite rule had lower confidence scores, not satisfying the minimum confidence threshold. This suggests that in the case of the rules {chancellor} -> {brown} and {howard} -> {tony}, the order in which the words are presented in the document matters as it starkly affects the confidence scores of the rules depending on which word comes first

## Mining Association Rules – By Category

Apart from mining association rules for the overall set of frequent items from the extracted keywords dataset, association rules can also be mined for each category of news by creating separate datasets sorted by category and then repeating the entire process of association rule mining. By reducing the scope to each category, this section aims to understand the dataset from a categorical point of view, by analyzing each category as a separate dataset.

### Sports

To perform association rule mining on individual categories, we must first convert the keywords from the category into a one-hot encoded NumPy Boolean array. To do that, we must first use the .query() method provided by the Python pandas library to store the keywords of each category into a separate variable ‘sorted\_keywords’:

Text, letter

Description automatically generated

Figure 37 – Storing the keywords dictionary pair into ‘sorted\_values’ variable

Once the keywords dictionary pair is stored in ‘sorted\_values’, we can then append the values to a list so that we can encode the values into a one-hot encoded NumPy Boolean array:

Graphical user interface, text, application

Description automatically generated

Figure 38 – Encoding the keywords for the sports category into a one-hot encoded NumPy Boolean array

From Figure 38, we can see that there are a total of 511 news stories and 882 unique keywords for the ‘Sports’ category.

Graphical user interface, application, Word

Description automatically generated

Figure 39 – Number of rules generated upon using the same threshold values for support and confidence

It can be seen from Figure 39 that upon using the same minimum support threshold of 0.5% and minimum confidence threshold of 40%, there are a significantly larger number of rules generated, at 427, compared to 17 when association rule mining was performed on the overall extracted keywords dataset. This is because when the sample size decreases from all five categories to only one category, the average support and confidence value for the item sets and rules will inherently increase, as not only are they more relevant to the category, but there is a smaller sample size with less irrelevant documents. As such, to set appropriate minimum thresholds for support and confidence, the threshold values must be increased.

Graphical user interface, application

Description automatically generated with medium confidence

Figure 40 – Generating association rules by setting a support threshold of 15% and a confidence threshold of 80%

With reference to Figure 40, increasing the minimum support threshold to 15% and the minimum confidence threshold to 80%, we can see that now only 11 rules are generated. Some interesting insights are:

1. For the sports category, the best rule would be {greek} -> {thanou}, with a confidence of 90%, suggesting that the word ‘thanou’ appears 90% of the time the word ‘greek’ appears.
2. The rule also has a lift of 38, suggesting that the words ‘thanou’ and ‘greek’ co-occur 38 times more than expected if they were statistically independent items.
3. The words ‘thanou’ and ‘greek’ also have a support value of 17.6%, suggesting that they appear in 17.6% of all news stories that belong to the ‘Sports’ category.
4. Interestingly, Thanou is the name of a Greek Olympic athlete, giving reason as to why these keywords not only have a high likelihood of co-occurrence but also a large dependence on each other.
5. The names and corporations of many other athletes were mentioned in the rules as well, such as Gerrard, who was a former Liverpool player

### Tech

This process of converting the keywords by category to a one-hot encoded NumPy Boolean array and then setting appropriate minimum support and confidence thresholds is then repeated for the other four categories. In the case of Tech, a minimum support threshold of 13% and a minimum confidence threshold of 80% managed to generate 8 rules:

Table

Description automatically generated with low confidence

Figure 41 – Generating association rules for the ‘Tech’ category

From Figure 41, we can see that 3 rules manage to obtain a perfect confidence score of 100%. In such cases, it can be noted that the ***conviction*** of a rule, which is a measure that defines the dependence of the consequent on the antecedent of the rule will also be defined as ‘*inf’,* meaning *infinity*, because it suggests that the ***antecedent*** is **fully dependent** on the ***consequent.*** To further elaborate, the conviction of a rule can be denoted by:

Text

Description automatically generated

Equation 3 – Conviction of a rule

As such, looking at Figure 41 and Equation 3, when a perfect confidence score of 100% (i.e. confidence (A->C) = 1) is achieved, the conviction of the rule will be at its maximum range, denoted by *inf.* Therefore, as derived from Figure 41:

1. The rules with the strongest co-occurrence would be the 3 rules with a perfect confidence score
2. The rule that is the most interesting is the rule {spyware} -> {program, software}, with a lift value of 42.9, suggesting that many news stories write about the terms spyware, program and software dependently on each other, which could signify a growing relevance of anti-spyware programs/software that results in more documentation
3. The most frequent itemset would be {handset, mobile}, with a support of 17.5%, which suggests that many news stories in the tech category frequently advertise or mention mobile handsets
4. Spyware was a heavily discussed topic in the field of tech at the time, as seen from the high occurrence of the item “spyware” in most rules
5. Other rules point to other rising technologies at the time such as the Nintendo

### Politics

Next, association rule mining is performed on the ‘Politics’ category, by setting the appropriate minimum support threshold to 13%, and the minimum confidence threshold to 80%:

Table

Description automatically generated

Figure 42 – Generating association rules for the ‘Politics’ category

From Figure 42, similar to the ‘Tech’ category, we can see that there are a number of rules that achieve a perfect confidence score. As seen from the figure:

1. There are 7 rules in this case, that display a perfect co-occurrence of 100%, such that any time the antecedent of the rule occurs, the consequent of the rule will always occur in the same news story as well
2. In addition, 2 out of the 7 rules that achieve a perfect confidence score {id} -> {card} and {card} -> {id}, are the most interesting rules with a lift of 69.5, suggesting a high degree of dependence
3. The item set {silk, kilroy} is the most frequent item set, with a support value of 19%, alluding to the former English politician Robert Michael Kilroy-Silk as a commonly mentioned name in the politics category of news stories
4. The political news coverage at the time was also heavily euro-centred, with mentions of the Blair government and Chancellor Brown, likely talking about the Blair – Brown deal that surfaced in 1994

### Business

Next, association rule mining is performed on the ‘Business’ category, by setting the appropriate minimum support threshold to 15%, and the minimum confidence threshold to 80%:

A picture containing table

Description automatically generated

Figure 43 – Generating association rules for the ‘Business’ category

From Figure 43, we can see that a total of 6 rules were generated. Some other insights derived were:

1. For this category, the rule with the strongest likelihood of co-occurrence is {barrel} -> {oil}, with a perfect confidence score of 100%
2. Thereafter, the most interesting rules are {lse} -> {boerse} and {boerse} -> {lse}, both of which having a lift of 45.3%, eluding that the London Stock Exchange is highly dependent on ‘boerse’, which is the German translation of the words ‘stock market’, and are used in co-occurrence frequently in the ‘Business’ category
3. The item set {boerse, deutsche} is the most frequent item set, with a support value of 17.6%, referencing ‘Deutsche Boerse’, a capital market company that is implied to have a high degree of relevance and occurrence in the ‘Business’ category
4. Mention of oil and oil production subsidiaries like *Yuganskneftegaz* in Russia

### Entertainment

Last but not least, association rule mining is performed on the ‘Entertainment’ category, by setting the appropriate minimum support threshold to 15%, and the minimum confidence threshold to 80%:

A picture containing graphical user interface

Description automatically generated

Figure 44 – Generating association rules for the ‘Entertainment’ category

From Figure 44, we can see that a total of 5 rules were generated. Some other insights derived were:

1. For this category, there are two rules with the strongest co-occurrence {berlin} -> {film} and {berlin, festival} -> {film}, both of which having a perfect confidence score of 100%, suggesting that there is a high likelihood that the word ‘berlin’, or the words ‘berlin’ and ‘festival’ occurs together with the word film, alluding to the Berlin International Film Festival
2. Thereafter, the most interesting rule that satisfy the aforementioned thresholds belong to {berlin} -> {film, festival}, at a lift value of 18.7, which references the Berlin International Film Festival as well, and implies that the event was highly interesting and frequently mentioned together in the ‘Film’ category
3. Lastly, the item set {berlin, film}, is the most frequent itemset, with a support value of 20.7, also alluding to the same Berlin International Film Festival
4. In general, most news stories under the category of “Entertainment” were in discussion of the Berlin Film Festival

To summarise this section, data was first extracted into unique keywords, then the extracted keywords were visualised using various visualization techniques. Subsequently, using the a priori algorithm, frequent itemsets were generated and lastly, association rule mining was performed on the overall extracted keywords dataset, and for each individual category. To better visualise the overall insights obtained from this section, a summary table was made that recorded the notable rules and the general insights of each dataset rule mining was performed on:

Calendar

Description automatically generated

Figure 45 – Summary table from performing rule mining

# Summary and Further Improvements

## Summary

In summary, using the 3-part approach of pre-processing, transforming and understanding, we were able to perform text analysis to transform the unstructured textual data into a meaningful and interpretable form. In addition, we were able to see the methodology and logic behind each of the techniques applied at each step of the approach, and how they managed to play a part in collectively understanding the data. While this is merely a glimpse into the domain of Natural Language Processing, through this assignment, we get to see how some of the important frameworks and libraries associated with NLP such as the NTLK framework allow us to systematically dissect and interpret unstructured textual data. Lastly, following the universal workflow of machine learning, after we have successfully prepared, transformed and understood the data, we finally proceeded with building classification models to classify the data to their corresponding categories, which will be further explored in Problem 2 of this assignment.

## Further Improvements

While this was a demonstration of how text analytics can be performed, some considerations and improvements can be expanded upon in the future to make the overall process of analysing unstructured text more robust and meaningful:

### Distributional Semantics

Despite being effective methods for extracting features from text, Bag of Words and TF-IDF models are unable to capture additional information in a text document such as the semantics, structure, sequence and context around nearby words. As such, more sophisticated models could be explored such as Word2Vec, which is a predictive deep learning-based model to compute and generate high quality, distributed and continuous dense vector representations of words that capture contextual and semantic similarity. In addition, count-based methods such as Latent Semantic Analysis (LSA) can be used to calculate statistical measures of how often words occur with their neighboring words in a corpus and then build dense word vectors for each word, to build better quality word vectors based on semantic and contextual similarity.

### Count matrix vs TF-IDF matrix

Secondly, in this assignment, only the TF-IDF matrix was used to extract keywords and perform data understanding. To expand on this, the use of the count matrix could be considered as well, to produce a different set of extracted keywords. This makes it so that data understanding can be performed on two different sets of data to compare and see if any differences can be observed for better depth and greater understanding of the data.

# Module Reflection

Applied Analytics was a very insightful and useful module. Being particularly interested in the field of Data Science and Analytics, I decisively chose this module as one of my elective modules, the other being Deep Learning, with the aim of pursuing this interest and learning more about the study of Machine Learning and Artificial Intelligence. After completing this module, I have found the skills I learnt such as basic Excel, cluster analysis, text analysis and classification very useful and interesting. In particular, I have found this assignment on Natural Language Processing and text classification very fun and meaningful, and something I look forward to expanding into greater depth in the foreseeable future. Looking back, choosing this module as one of my elective modules was a decision I did not regret, and I am grateful for the experiences and skills I have gained from this module.