## 36118 Applied Natural Language Processing

# Assignment 1 - Text Analysis

## Name – Shivansh Pandey

## Student ID – 24627453

## Date – 9/04/2024

### Overview

The 29 plain text files, each containing a transcript of an Australian radio talkback show, make up the dataset used for this assignment. The sources of these transcripts are mostly divided into two groups of radio stations:

1. ABC Radio Stations:

NAT: ABC National Radio

ABCE: ABC Radio broadcasts to eastern Australia

ABCNE: ABC Radio broadcasts to southern and western Australia

This collection consists of fourteen transcripts from various public broadcasters that offer a variety of conversations that are reflective of both national and local audiences.

1. Commercial Radio Stations:

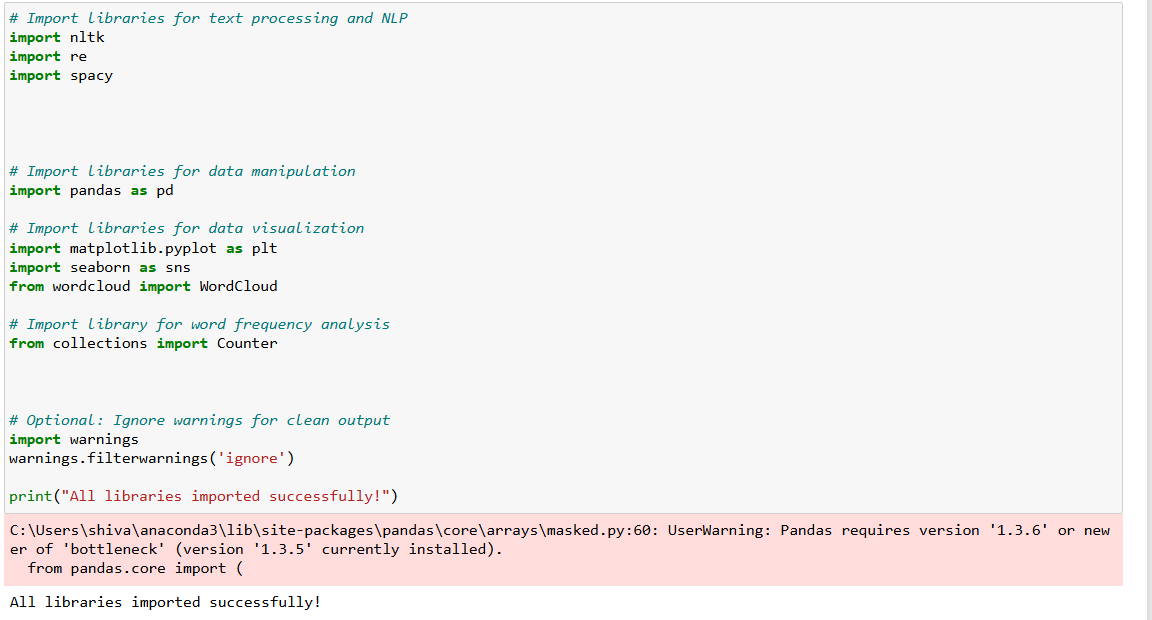
COME: Eastern Australia is served by commercial stations.  
COMNE: Western and Southern Australia's commercial broadcast stations  
Fifteen transcripts from commercial stations are included in the collection, providing information about the subjects and material covered in these areas.

Prior to its management by Griffith University from 2012 to 2023, this collection was a part of the Australian National Corpus (AusNC), a linguistic resource. A rich, publicly accessible library of Australian English that captures the language as it is spoken in different circumstances around the nation was the goal of the AusNC program.

I'll be using this dataset for topic modeling and text analysis in this project. The objective is to identify the common patterns that emerge from public and commercial broadcasters in order to compare the fundamental subjects presented across these transcripts. The disparities in programming concentration between regional commercial stations and national public radio will be clarified by this investigation.

### Step 1 : Import and install required libraries, packages and loading dataset

Some packages will be installed as per the requirements, below you can see the common imports and installation snap from the notebook.



A close-up of a computer code

Description automatically generated

The program extracts text files from a given directory, sorts the contents according to alphabetic prefixes in the filenames, and then saves the files. After that, it saves this data in a list of dictionaries, which is then transformed into a DataFrame for simple analysis. Debugging lines are also added to ensure that the files and categories are processed correctly.

A screenshot of a computer program

Description automatically generated

### Step 2 : Data Exploration/ Data Understanding

A number of crucial actions are taken during the data understanding phase to guarantee that the dataset is ready for analysis. To make sure the dataset is complete and has no gaps that could interfere with the analysis, the first step is to look for any missing values. To swiftly confirm that the data has been loaded correctly and is in the desired format, the first few rows of the DataFrame are printed next. We then look at the distinct categories in the dataset to make sure that all of the desired categories were accurately recognized during the classification process. 'Unknown' categories are indicated for more analysis to resolve any problems encountered during the classification process.

The actual categories found in the dataset are compared with a prepared list of expected categories in order to further confirm the categorization. This comparison guarantees the accuracy and consistency of the dataset's categorization. The frequency of each category within the dataset is then counted and visualized to assess the distribution of categories, reveal which categories are more common, and offer insights about the dataset's balance.

Lastly, an effort is made to produce the dataset's descriptive statistics. Although the majority of the dataset is textual, which may limit the applicability of statistical measures like mean or standard deviation, this phase aids in locating any potential numerical columns and gives a broad picture of the dataset's organization. All things considered, these actions guarantee a thorough comprehension of the dataset and lay the groundwork for the analysis that follows.

**Steps are explained below –**

Look for missing values: This code looks for any missing values in the dataset to make sure all the data is there and complete, ready for analysis.

Print the DataFrame's initial few rows: This allows for a fast confirmation that the data has been loaded correctly and is formatted according to expectations by displaying a tiny sample of the data.

Check unique categories: This code describes the unique categories that are present in the dataset in order to confirm that the intended categories have been appropriately detected throughout the classification process.

'Unknown' categories: This highlights any problems with the categorization procedure for additional examination by identifying and displaying any data entries that were unable to be categorized.

A screenshot of a computer

Description automatically generated

Comparing with a reference list of expected categories: This code creates a list of expected categories and compares it with the categories that are present in the dataset. The result demonstrates that the categories that were actually taken from the dataset match the categories that were predicted, proving that the categorization procedure was accurate and successful.

Distribution of categories: This code counts and shows how many times each category appears in the dataset. This provides information on the dataset's balance by showing that transcripts from some categories—like 'COME' and 'NAT'—are more prevalent in the dataset than transcripts from others—like 'ABCNE'.

Descriptive statistics of the dataset: This code uses the describe() function to try and produce a summary of the dataset's statistics, but the result isn't displayed here. For numerical columns, df.describe() often returns statistical metrics (such as count, mean, and standard deviation). However, unless there are numerical columns available, this method could fail to provide relevant output because the dataset mostly consists of text data.

A screenshot of a computer

Description automatically generated

### Step 3 :Text – Preprocessing

These actions were taken in order to fully comprehend the dataset and get it ready for a more thorough examination. While examining the distribution of categories offers insights into the structure and balance of the dataset, combining text and generating word clouds aids in discovering major terms and themes throughout all transcripts. By creating distinct word clouds for every category, it is possible to compare the language used by various groups. It is possible to perform more in-depth text analysis, including frequency analysis or pattern recognition, by tokenizing the text into individual words. Last but not least, examining bigrams and trigrams reveals frequent word combinations that provide distinct conversational patterns within each category. This information is essential for comprehending the subtle differences in language usage among various radio stations.

Word count Freq - This code initially creates a complete collection of words by joining all the text data from the dataset into a single string. The next thing it does is create a word cloud, which is a picture where each word's size represents how frequently that term occurs in the dataset. Lastly, Matplotlib is used to display the word cloud, making it possible to quickly identify the most prevalent and important terms throughout all of the transcripts visually.

A screen shot of a computer screen

Description automatically generated

Bar chart - To visually depict the distribution of categories within the dataset, this code creates a bar plot. It shows the number of occurrences for each category using Seaborn's count plot, where the y-axis indicates the counts for each category and the x-axis represents the various categories. With titles and labels added to the plot, it offers a clear visual representation of the transcript distribution throughout the dataset's many categories.

A screen shot of a graph

Description automatically generated

Word frequency for category - For every category in the dataset, a distinct word cloud is created and shown by this code. It first creates a single string out of all the text connected to each distinct category by iterating through them. The most common terms within each category are then visually highlighted by creating a word cloud for that particular category. Lastly, a title identifying the category each word cloud represents is shown, enabling a visual comparison of the terms that are prevalent in various categories.

A screenshot of a computer

Description automatically generated

A collage of words

Description automatically generated

A close up of words

Description automatically generated

Tokenization - The text in each row of the dataset is tokenized by this code, which separates the text into discrete words (tokens) and stores the tokens in a new column named "tokens." Every row's text is subjected to the NLTK word tokenize function, which makes it easier to conduct additional word-level text analysis. The result validates that the tokenization process was finished, and the new column is operational by demonstrating that the 'tokens' column has been successfully added to the DataFrame alongside the other columns.

A screen shot of a computer code

Description automatically generated

N-gram - In order to examine common word patterns, this function creates bigrams (two-word sequences) and trigrams (three-word sequences) for each category in the dataset. The text in each category is first tokenized, after which n-grams (bigrams and trigrams) are produced. The frequency of these n-grams is determined by the code using a Counter. Next, for each category, the top 10 most frequent bigrams and trigrams are shown, exposing common word combinations unique to each radio station group. The results indicate that certain phrases—like "it's" and "I think"—are used often in all categories, while other phrases—like "a little bit" or "don't know"—highlight certain conversational tendencies among various groups. Understanding the common linguistic patterns and catchphrases utilized in various radio programs is made easier by this study.

A screenshot of a computer code

Description automatically generated

A white background with black and white lines

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph of a number of words

Description automatically generated

### Step 4: Data Preparation

In order to prepare the data for precise and insightful analysis, this text cleaning phase is essential. Noise in raw text data frequently includes punctuation, stopwords, and word variants that aren't necessary to comprehend the primary themes or patterns. The data is made more consistent and focused by normalizing the text, eliminating superfluous parts, and lemmatizing words. This makes it easier for algorithms and models to recognize important phrases, patterns, and subjects. This preprocessing makes sure that the analysis that comes after—be it machine learning, topic modeling, or frequency analysis—is founded on accurate and pertinent data, producing more perceptive outcomes.

Below code uses multiple steps to clean and treat text data inside of a DataFrame. The required resources, such as tokenizers, stopwords, and lemmatizers, are first downloaded from NLTK. Next, the function clean\_text is developed, which standardizes text by filtering out stopwords and non-alphabetic words, tokenizing the text into terms, converting it to lowercase, and eliminating punctuation. Additionally, the words are lemmatized, or reduced to their most basic forms. To further improve text cleanliness, additional domain-specific phrases like "um" and "uh" are added to the stopwords list. Finally, the cleaning function is used on every text element in the 'text' column of the DataFrame. This creates a new column called 'cleaned\_text', which holds the processed text and prepares it for additional analysis.

A screenshot of a computer program

Description automatically generated

This code first determines which words are most frequently used in the text that has been cleaned in order to assess the efficacy of the text cleaning procedure. The cleaning procedure is repeated after adding more frequently occurring but uninformative words to the stopwords list. After making sure the text has been appropriately cleaned and is now more focused on important material, the code reexamines the most common words to see what is still there. This better prepares the text for additional analysis. This stage aids in confirming that the majority of superfluous or filler words have been eliminated, leaving the crucial terms that are important for further investigation.

A screenshot of a computer program

Description automatically generated

### Step 5 : model – Topic modelling using LDA

1. Text Vectorization with TfidfVectorizer : Vectorization was a crucial initial step in getting your text data ready for subject modeling. In doing so, the textual data is transformed into a numerical format known as a document-term matrix (DTM), in which each row denotes a document (a transcript in this instance) and each column a term or word. Based on a word's frequency throughout all papers, the matrix's values represent each word's relevance.

experienced an issue at first since the stop\_words parameter needs to be formatted as a list correctly. This was fixed by turning the stopword set into a list, and the vectorization procedure produced a DTM. This stage was crucial since vectorization makes it possible to extract significant patterns from text data, and the LDA model requires numerical input.

1. Validation of Cleaned Text: Had to make sure that the text data was in a format appropriate for vectorization by verifying that the cleaned\_text column was correctly formatted as a string before moving on to the whole dataset. Moreover, the vectorization procedure was applied to a tiny sample of the data to make sure it functioned well on a smaller scale. This phase confirmed that the pretreatment stages were successful and helped avoid potential problems when working with the entire dataset.
2. Final Vectorization: then, feeling secure that the text data and stopword list had been handled appropriately, applied the TfidfVectorizer to the complete dataset. As a result, a finished DTM that was prepared for the topic modeling procedure was produced. To make sure the matrix was correctly generated, the DTM's shape was examined, and each transcript was represented by a set of weighted terms.

**Latent Dirichlet Allocation (LDA) Model:**

Using the Latent Dirichlet Allocation (LDA) model was the last step after the DTM was ready. Using the assumption that every document is a mixture of subjects and that every word in the document can be associated with one of these topics, LDA is a generative statistical model that finds a set of themes within a collection of documents. Based on the analysis, five subjects were chosen, and the LDA model was fitted to the DTM. The most pertinent terms related to each issue were then determined by the LDA model and printed out. These topics, each of which is defined by a group of important phrases, stand in for the underlying themes found in the text data.

Five different subjects were identified in the dataset via the Latent Dirichlet Allocation (LDA) model's output, each of which was identified by a set of important terms:

* Topic 1 - terms like "plant," "garden," "tree," "bird," and "house," which are associated with gardening and nature, particularly in Queensland, are the subject of Topic 1. These terms also indicate conversations about outdoor activities and environmental issues.
* Topic 2 - Terms like "doctor," "blood," "golf," and "salt," which are linked to health, medicine, and sports, are highlighted. These terms indicate discussions about sports, medical advice, or potentially even specific people like "Graham."

A close up of words

Description automatically generated

* Both Topics 3 and 4 place a strong emphasis on words like "person," "remember," "wanted," and "case," which could point to conversations about anecdotes, recollections, or talkback exchanges involving people like "Trevor."

**A screenshot of a computer

Description automatically generated**

* Topic 5 - includes terms like "flower," "paint," "book," and "green," which are associated with art, literature, and nature and indicate subjects that may involve artistic endeavors or readings.

**A close-up of words

Description automatically generated**

**Reference**

* **Class slides, lab notebooks**
* **W3school**
* **Anaconda.org**
* [**NLP: Extracting the main topics from your dataset using LDA in minutes | by Priya Dwivedi | Towards Data Science**](https://towardsdatascience.com/nlp-extracting-the-main-topics-from-your-dataset-using-lda-in-minutes-21486f5aa925)
* **Previous lab notebooks for python libraries**
* **Google search for troubleshooting**
* **Edureka**
* **Youtube**