**Mohammad Mahmudul Hoque**

# AN ANALYTICAL APPROACH TO ANALYZE THE POPULAR WORD SEARCH FROM NINETEEN-YEAR NEWS DATASET USING NATURAL LANGUAGE PROCESSING TECHNIQUE

**Thesis**
**CENTRIA UNIVERSITY OF APPLIED SCIENCES**
**Business Intelligence Technologies**
**June 2023**

**ABSTRACT**

| | | |
|---|---|---|
| **Centria University of Applied Sciences** | **Date** June 2023 | **Author** Mohammad Mahmudul Hoque |
| **Degree Programme** Business Intelligence Technologies | | |
| **Name of thesis** AN ANALYTICAL APPROACH TO ANALYZE THE POPULAR WORD SEARCH FROM NINETEEN-YEAR NEWS DATASET USING NATURAL LANGUAGE PROCESSING TECHNIQUE | | |
| **Centria supervisor** Jari Isohanni | | **Pages** 5 + 34 |
| **Instructor representing commissioning institution or company.** Mitha Rachel Jose | | |

Natural language processing (NLP) has seen significant growth in 2022 because of the growing availability of digital data. With the development of big data, it is now important to analyze and obtain valuable conclusions from massive databases. Natural Language Processing (NLP) is a subfield of computer science and artificial intelligence that deals with the interaction between human language and computers. It involves the development of algorithms and computational models that can analyze, understand, and generate human language. This study focuses on an algorithm to find the most popular word from nineteen-year news data. There are many different approaches that can be used to implement word search games using NLP, and the specific approach used will depend on factors such as the size of the grid, the complexity of the clues. This thesis study explores the use of NLP tools to identify and find out the most popular words in the nineteen-year news datasets. The study will employ a range of NLP techniques such as vectorization, stemming, tokenization, stop word removal and many to preprocess the data. After preprocessing the data, the study will use frequency analysis (Term Frequency-Inverse Document Frequency: TF-IDF) to identify the most commonly occurring words in the dataset. The study will then rank the words based on their frequency and identify the most popular words.

**Key Word**
Big data, NLP, Vectorization, Stemming, Tokenization, Stop words, TF-IDF

## CONCEPT DEFINITIONS

**NLP**

Natural Language Processing (NLP) is a branch of Data Science which deals with Text data

**Big Data**

A collection of extremely large and complex data.

**Vectorization**

It refers to the process of performing operations on entire arrays or metrices of data, instead of processing them element by element.

**Stemming**

It is an algorithm which typically uses linguistic rules to identify and remove common suffixes and prefixes from words.

**Tokenization**

It is the process of breaking up a large piece of text into smaller units called tokens, which can be words, phrases, symbols, or other meaningful elements.

**Stop Words**

Stop words are commonly occurring words in a language considered of little value in discerning a text's meaning. Stop words in English are, “the, is, of, that, in, it, and, with”.

**TF-IDF (Term Frequency-Inverse Document Frequency)**

It is a technique used in NLP to evaluate the importance of words or terms in a text corpus.

**FIGURES**

**CODES**

# 1 INTRODUCTION

Data science is an interdisciplinary (Donoho 2017) field that involves the use of statistical, mathematical, and computational techniques to extract insights and knowledge from data. It combines elements of statistics, machine learning, computer science, and domain expertise to solve complex problems. There are many different techniques that data scientists use to analyze and make sense of data. Some of the most common data science techniques are Descriptive statistics, Inferential statistics, Data visualization, Regression analysis, Time series analysis, Clustering, Classification, Natural language processing and Deep learning.

The process of data science typically involves some specific steps which starts with data collection, and it involves gathering the relevant data from various sources, including databases, APIs, sensors, and other data sources. Data preprocessing involves cleaning, transforming, and organizing the data into a format suitable for analysis. This may involve handling missing values, outliers, and data errors. For example, some may not be comfortable sharing information about their salary, drinking, and smoking habits. It is not possible to get information regarding certain questions in a data collection survey. For example, some may not be comfortable sharing information about their salary, drinking, and smoking habits.These are left out intentionally by the population. In some cases, data is accumulated from various past records available and not directly. In this case, data corruption is a major issue. Due to low maintenance, some parts of data are corrupted giving rise to missing data. Inaccuracies during the data collection process also contribute to missing data. For example, in manual data entry, it is difficult to completely avoid human errors, equipment inconsistencies leading to faulty measurements, which in turn cannot be used.

Exploratory data analysis use tools like data visualization, descriptive statistics, and others to gather insights into the data and spot trends and connections. This method is enhanced by feature engineering, which involves choosing and producing features or variables that are relevant to the issue at hand. These features or variables may be created by combining, altering, or creating new variables. Following the engineering of the features, a suitable machine learning algorithm is selected and trained on the data to produce a predictive model. This process is known as model selection and training. Model evaluation and testing involves the use of an invalid dataset or cross-validation methods to make sure the model is efficient and generalizable to new data. The end-to-end process is then completed by deploying the successful model into a real-world setting where it may be used to make predictions or influence decisions.

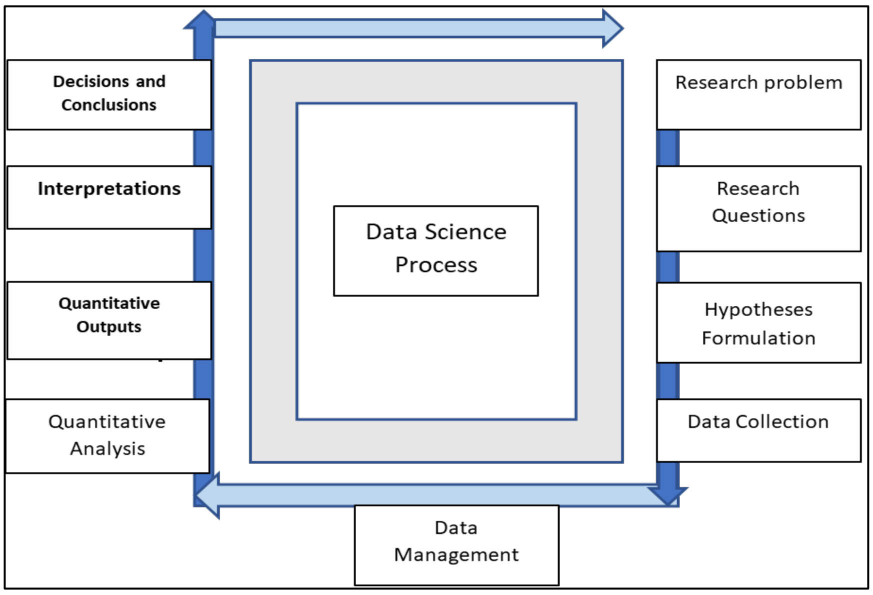


FIGURE 1. Data Flow Diagram

The study is based on popular word searches from news headlines so that it needs an explanation to the related topic. News headlines are a major source of information for many people worldwide in the fast-paced world of today. Therefore, analyzing the most frequently used words in news headlines can shed light on the subjects that are on people's minds on a personal, social, and even national level. In this study, it shows what are the most frequently searched terms and phrases can tell us about the current situation around the globe by analyzing a dataset of news headlines from a variety of sources. The dataset employed to conduct this project is composed of 1048574 samples. And the whole dataset has two columns, mainly "Date: when the article was published' and 'Headline Text: title of the article in English'. The texts from which need to search for the most popular words occurred in the total number of headlines. The dataset contains data on news headlines published over a period of nineteen years. Based on information from the reputable Australian news company ABC (Australian Broadcasting Corporation). The main aim of this project is to dig into the keywords (the most popular words) so that one can see all the important episodes shaping the last decade and how they evolved over time.

Therefore to analyze the most popular word search, usually used Natural Language Processing Technique i.e TF-IDF (Term Frequency-Inverse Document Frequency: TF-IDF) to identify the most commonly occurring words in the dataset. TF-IDF Vectorizer is a measure of originality of a word by comparing the number of times a word appears in document with the number of documents the word appears in (Chaudhary M 2020). The study will then rank the words based on their frequency and identify the most popular words.

# 2 BACKGROUND TO THE STUDY

Data science has become an effective tool in recent years for evaluating huge datasets, particularly text data. A branch of data science called "Natural Language Processing" (NLP) belongs to the study of human language. Executing popular word searches from a 19-year news dataset as the project example, looking at some of the important research articles on applying NLP approaches to evaluate text data.

Manning et al.'s (2008) study was one of the first ones in the field, which presented a system for gathering and evaluating semantic data from massive text databases using NLP methods. They analyzed a significant amount of news stories to show the effect of their method ( Manning, Raghavan, & Schütze 2008). Researchers have more recently used NLP approaches to study word usage and frequency in news articles. For instance, Petrovic et al (2013) employed NLP methods in their study to examine the frequency of terms associated with political and economic events in a sizable collection of news stories. They found that variations in keyword frequency were closely tied to things like political elections and economic crises (Petrovic, Osborne, & Lavrenko 2013). The usage of NLP approaches to evaluate text data has grown in popularity recently. These methods have been used by researchers to examine the frequency of keywords associated with a wide range of news story themes, such as political events, economic crises, social difficulties, and public health problems. These studies show how data science and NLP techniques have the potential to offer important insights into the study of text data.

## 2.1 NLP

NLP is a subfield of artificial intelligence (AI) and linguistics that focuses on the interaction between computers and human language. NLP aims to enable computers to understand, interpret, and generate natural language text or speech, allowing them to communicate with humans more effectively. NLP involves various techniques, algorithms, and models to process and analyze human language data.The study of language processing is called NLP. People who are deeply involved in the study of language are linguists, while the term ‘computational linguist’ applies to the study of processing languages with computer application of computation (Hardeniya, Perkins, Chopra 2016). The natural language processing are using machine learning algorithms to analyze and understand human language. This is the technique used to analyze the most popular word search. NLP involves a range of tasks, including morphological analysis, syntactic analysis, semantic analysis, and discourse analysis. Morphological analysis deals with the structure of words and their component parts, while syntactic analysis deals with the structure of sentences and the relationships between words. Semantic analysis deals with the meaning of words and sentences, and discourse analysis deals with the larger context in which language is used. Natural Language Processing (NLP) techniques can be utilized to develop algorithms for solving word search puzzles automatically or to aid players in finding the words more efficiently. There are several Python libraries that can be used for implementing natural language processing (NLP) techniques in word search puzzles. NLP techniques rely on machine learning, deep learning, statistical modelling, linguistic rules, and other approaches to process and analyze language data. It involves tasks at various levels of linguistic analysis, including morphological, syntactic, semantic, and pragmatic analysis. The libraries used to find the most popular word search are explained below.

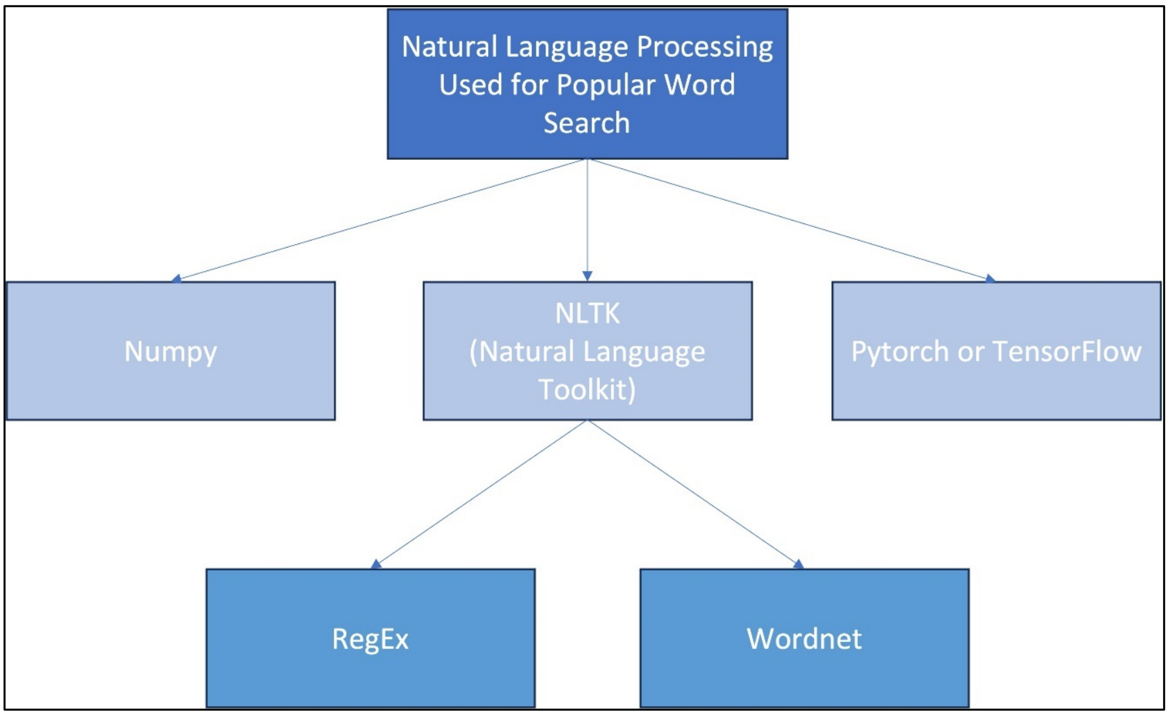

Natural Language Processing Used for Popular Word Search
Numpy
NLTK (Natural Language Toolkit)
Pytorch or TensorFlow
RegEx
Wordnet

FIGURE 2. Natural Language Process Diagram

Depending on the specific requirements and complexity of the word search puzzle, other libraries or custom implementations may also be needed. It is important to choose the appropriate libraries based on the specific tasks and functionalities needed in your word search puzzle implementation. The quantity of the puzzle and the capabilities required can be carefully considered by developers to ensure that the libraries they select match their needs and provide the word search puzzle with the best possible answer.

NLTK (Natural Language Toolkit) is a popular Python library for NLP tasks, including text segmentation, tokenization, and string matching. It provides a wide range of functionalities for working with text data, including word and sentence tokenization, part-of-speech tagging, and more, which can be useful in processing word search puzzles. NLTK is a standard python library that provides a set of diverse algorithms for NLP. It is one of the most used libraries for NLP and Computational Linguistics.The process of cleaning unstructured text data, so that it can be used to predict, analyze, and extract information.These are some of the methods to process the text data in NLP such as tokenization, frequency distribution of words, filtering stop words, stemming, lemmatization , parts of speech(pos) tagging, named entity recognition and wordnet.

Regular Expressions (RegEx) are powerful tools for pattern matching in text data. Python's built-in re module provides support for regular expressions, which can be used to implement string matching and pattern matching functionalities in word search puzzles.For instance If you wanted to tokenize the string into word and non-word chars, you could use \w+|\W+ regex. However, in your case, you want to match word character chunks that are optionally followed with ' that is followed with 1+ word characters, and any other single characters that are not whitespace.

NumPy is a popular numerical computing library for Python. It provides efficient and fast operations on arrays, which can be used for grid manipulation and data processing in word search puzzles. NumPy can be used to represent and manipulate the grid of letters in the word search puzzle, making it easier to implement algorithms for word identification and orientation (NumPy, 2023) . NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

Pytorch or TensorFlow are popular deep learning libraries that can be used for more advanced NLP tasks, such as training and using neural networks for word search puzzles. Deep learning models, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), can be trained to identify words or patterns in the grid of letters in word search puzzles. (PyTorch , 2023). PyTorch is a relatively new deep learning framework based on Torch. It is Developed by Facebook’s AI research group and open-sourced on GitHub in 2017, it’s used for natural language processing applications.
PyTorch has a reputation for simplicity, ease of use, flexibility, efficient memory usage, and dynamic computational graphs. It also feels native, making coding more manageable and increasing processing speed.TensorFlow is a symbolic math library used for neural networks and is best suited for dataflow programming across a range of tasks.It offers multiple abstraction levels for building and training

models. A promising and fast-growing entry in the world of deep learning, TensorFlow offers a flexible, comprehensive ecosystem of community resources, libraries, and tools that facilitate building and deploying machine learning apps. Also, as mentioned before, TensorFlow has adopted Keras, which makes comparing the two seem problematic. Nevertheless, we will still compare the two frameworks for the sake of completeness, especially since Keras users don't necessarily have to use TensorFlow.

WordNet is a lexical database that provides information about the meanings, synonyms, and relationships between words. The nltk library includes WordNet, which can be used for semantic analysis or providing contextual clues in word search puzzles (WordNet, 2023). WordNet has been used for a number of purposes in information systems, including word-sense disambiguation, information retrieval, automatic text classification, automatic text summarization, machine translation and even automatic crossword puzzle generation. WordNet is a lexical database of semantic relations between words in more than 200 languages. WordNet links words into semantic relations including synonyms, hyponyms, and meronyms. The synonyms are grouped into synsets with short definitions and usage examples. WordNet can thus be seen as a combination and extension of a dictionary and thesaurus. While it is accessible to human users via a web browser, its primary use is in automatic text analysis and artificial intelligence applications. WordNet was first created in the English language and the English WordNet database and software tools have been released under a BSD style license and are freely available for download from that WordNet website.

## 2.2 TF-IDF

The number of times a specific word appears in the document is referred to as the term frequency (TF). Because the IDF tends to favour short files, this number is typically normalized (the numerator is typically less than the denominator).The inverse document frequency (IDF) is a gauge of a word's overall significance. TF-IDF is a method of information retrieval that is used to rank the importance of words in a document. It is based on the idea that words that appear in a document more often are more relevant to the document. TF-IDF is the product of Term Frequency and Inverse Document Frequency. A high-weight TF-IDF can be produced by the combination of a word's high frequency in one file and its low file frequency over the full set of files. Consequently, TF-IDF tends to keep important words and filter out common words. ( Fan & Qin Y. 2018). TF-IDF Vectorizer is a measure of originality of a word by comparing the number of times a word appears in document with the number of documents the word appears in formula for TF-IDF is (Chaudhary M 2020), TF-IDF=TF(t,d)xIDF(t),

where, TF (t, d) = Number of times term "t" appears in a document "d".

IDF(t) = Inverse document frequency of the term t.

The TfidfVectorizer converts a collection of raw documents into a matrix of TF-IDF features. Term Frequency is the measure of the frequency of words in a document. It is the ratio of the number of times the word appears in a document compared to the total number of words in that document. Inverse Document Frequency is the measure of how much information the word provides about the topic of the document. It is the log of the ratio of the number of documents to the number of documents containing the word. TF-IDF method removes the drawbacks faced by the Bag of Words model. It does not assign equal value to all the words, hence important words that occur a few times will be assigned high weights. Bag of Words just creates a set of vectors containing the count of word occurrences in the document (reviews), while the TF-IDF model contains information on the more important words and the less important ones as well.

# 3 REQUIREMENTS AND METHODOLOGY

It is important to understand the significance of such a technique before exploring into the specifications and method for applying natural language processing (NLP) methods to analyze the well-known word search from a nineteen-year news dataset. News articles are vital in today's information-rich environment for shaping public opinion and reflecting societal trends. We can gain valuable insights from a large collection of news data by utilizing NLP, and we can achieve this by identifying the most common terms that have dominated headlines for over 20 years. Through this analytical process, we may better comprehend the discourse as it develops, define important issues and events, and understand the larger narrative that has influenced how people see the world.

## 3.1 Requirements

Nineteen-year news dataset are required for this project and this dataset of news articles spanning over 19 years to analyze the popular word search. This dataset should be in a suitable format that can be read and processed using data science tools and techniques. This contains data of news headlines published over a period of nineteen years and it is sourced from the reputable Australian news source ABC (Australian Broadcasting Corporation). With a volume of two hundred articles per day and a good focus on international news, we can be fairly certain that every event of significance has been captured here. Digging into the keywords, one can see all the important episodes shaping the last decade and how they evolved over time.

Programming skills are required and without programming skills this project cannot be proceed, and to do this project Python programming language best to use in data science to write and execute the code for data preprocessing, analysis, and visualization. One of the main reasons why data analytics using Python has become the most preferred and popular mode of data analysis is that it provides a range of libraries. The python programming language is scalable and flexible. It has a vast collection of libraries for numerical computation and data manipulation. It provides libraries for graphics and data visualization to build plots.

Natural Language Processing (NLP) tools to preprocess the text data, such as tokenization, stop word removal, stemming/lemmatization, and so on. For this preprocess steps need to use NLP libraries like

NLTK, SpaCy for this purpose. Many open-source programs are available to uncover insightful information in the unstructured text (or another natural language) and resolve various issues. Although by no means comprehensive, the list of frameworks presented below is a wonderful place to start for anyone or any business interested in using natural language processing in their projects. The most popular frameworks for Natural Language Processing (NLP) tasks are NLTK, Stanford CoreNLP,SpaCy,GPT-3, Apache OpenNLP, Google Cloud, Text Blob, Amazon Comprehend,Word2Vec and so on.

Word frequency analysis like word counts, the TF-IDF method is chosen for this project to analyze the frequency of words in the dataset. The popular words that appear frequently in the news articles will be easier to recognize according to this exploration. Search engines use word frequency to establish the subject of web pages. They developed complex linguistic analysis in order to classify pages by subject without human intervention. In turn, webmasters do the same, to try to fool search engines into assigning high keyword relevance to the pages they create. For instance, using a word with a 3% frequency gives a text good relevance on that word (or keyword, in a search engine context). A 10% frequency is still OK, but it is close to “keyword stuffing”, a technique used by webmasters who try to force their websites into the top places of the search engines. Keyword stuffing is penalized by the search engines, and needs to be prevented by smart use of synonyms. Either with synonymizer software or good writing skills.

Data visualization is a field in data analysis that deals with visual representation of data. It graphically plots data and is an effective way to communicate inferences from data. Using data visualization, we can get a visual summary of our data. With pictures, maps and graphs, the human mind has an easier time processing and understanding any given data. Data visualization plays a significant role in the representation of both small and large data sets, but it is especially useful when we have large data sets, in which it is impossible to see all of our data, let alone process and understand it manually. Python offers several plotting libraries, namely Matplotlib, Seaborn and many other such data visualization packages with different features for creating informative, customized, and appealing plots to present data in the most simple and effective way. Data visualization tools like Matplotlib are frequently used to make charts, graphs, and word clouds to show the results of the study.

## 3.2 Project Development Tools

By using these project development tools, the popular word search from the 19-year news dataset may be analyzed efficiently with natural language processing techniques, enabling the extraction of useful information from a huge amount of news data. The tools required for this system are python language, Google Collaboratory, some python libraries, or packages such as Numpy, pandas, seaborn, sklearn, scipy, matplotlib.
Python is a general-purpose, object-oriented programming language that has several implications across the software, web development, data science and automation environments. The language's dynamic semantics, high-level built in data structures, dynamic typing and dynamic binding make it one of the most useful languages for rapid application development (Corbo, 2022) although being simple to understand and use, Python can be scaled up and employed for large, challenging tasks like gathering big amounts of data and running machine learning & deep learning algorithms. The Python programming language is used in this work to implement the machine learning & deep learning methods.

Jupyter Notebook is an open-source web application which enables us to write and exchange codes and documents. It offers a setting where one can write code, execute it, examine the results, display data, and observe the outcomes all without exiting the setting. As a result, it is a useful instrument for carrying out end-to-end data science workflows, including data cleaning, statistical modelling, building, and training machine learning models, visualizing data, and a few other tasks (Cardoso, Leitao & Teixeria 2019).

Scikit-learn (Sklearn) is the most useful and robust machine learning library. Scikit-learn exposes a wide variety of machine learning algorithms, both supervised and unsupervised, using a consistent, task-oriented interface, thus enabling easy comparison of methods for a given application (Pedregosa, Varoquaux, Gramfort, Bertrand Thirion 2011). It uses a Python consistent interface to provide a set of efficient tools for machine learning and statistical models, such as classification, regression, clustering, and dimensionality reduction. NumPy, SciPy, and Matplotlib are the foundations of Scikit-learn, which is mostly written in Python.

TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. Its computational model is based on dataflow graphs with mutable state (Abadi 2016). It has a flexible ecosystem of tools, libraries, and community resources that allow researchers to extend the state-of-the-art in machine learning and developers to quickly build and implement ML applications.

Pandas is a software library created in Python that is used for data processing and analysis. Panda’s provides rich data structures and functions designed to make working with structured data set fast, easy, and expressive. It combines NumPy's high performance array computing characteristics with relational databases' and spreadsheets' flexible data manipulation features. The name, Pandas is derived from the term "panel data" which comes from econometrics and refers to data sets that comprise observations for the same persons over several periods (McKinney, 2013).

# 4 IMPLEMENTATION

This section outlines the working procedure. Firstly, need to collect the dataset and this dataset contains data on news headlines published over a period of nineteen years. Based on information from the reputable Australian news source ABC (Australian Broadcasting Corporation). Firstly, need to describe the dataset by showing the data head and tail and describe the dataset column. The, pre-processed the data that is needed to provide clean text for conducting the project. Then the TF-IDF vectorizer is applied to the clean text to get the result, which will be shown in plot view as well as converted to strings to show results in word cloud view.

Preprocessing data is a key step in many data analysis and machine learning tasks. In Python, it uses various libraries and techniques to preprocess your data. The steps followed for pre-processing the news data describes as follows,

Step 1: Import the necessary libraries.

Importing the necessary libraries is essential before starting to execute the analytical approach for analyzing a popular word search from a 19-year news dataset using natural language processing techniques. These libraries consist of the tools and techniques needed for statistical analysis, data manipulation, text processing, and visualization.

Step 2: Load the data

The popular word search from the 19-year news dataset will be examined after the appropriate libraries have been imported and the data has been loaded into the project. Accessing and importing the news dataset, which acts as the main source for analysis, are required steps in loading the data.

Step 3: Clean the data

After loading the news dataset, the next step in the analysis of the popular word search is to clean the data. Data cleaning involves preprocessing and transforming the dataset to ensure its quality, consistency, and readiness for further analysis.

Step 4: Text preprocessing

The news dataset must be prepared for analysis through text preparation. It involves changing unformatted raw text input into a format that has been defined suitable for tasks using natural language processing.

Step 5: Scaling or normalization

A preprocessing process called scaling or normalization wants to convert numerical data or variables to a similar scale or range. Through this method, it assures every defining provides equally to the analysis and that no feature controls the outcomes as a result of its higher quantity.

Step 6: Encoding categorical variables

When working with machine learning or data analysis tasks involving categorical data, encoding categorical variables is an essential preprocessing step. Categorical variables show qualitative or nominal information that cannot be simply translated into numerical values, such as gender, color, or nationality.

The Australian news open dataset is a collection of text data that includes news articles, headlines, and other textual content related to news and events in Australia. It encompasses a wide range of topics, including politics, economy, sports, entertainment, technology, and more. These datasets are typically used for various purposes, such as research, natural language processing (NLP), sentiment analysis, topic modelling, and data analysis.

The dataset may contain the full text of news articles published by various Australian news sources. Each article may consist of the article content, publication date, author information, and additional metadata. The dataset may include headlines or short summaries of news articles. Headlines provide a concise representation of the main points or topics covered in the corresponding articles. The dataset may include information about the news sources, such as the name of the news organization, website, or publication that published the article. Additional metadata may be available, including article IDs, categories or topics associated with the articles, geographic location, and other relevant information.

An overview of proposed method for popular word searches is also explained as a flow diagram. Figure 3. shows the procedural view of the work.

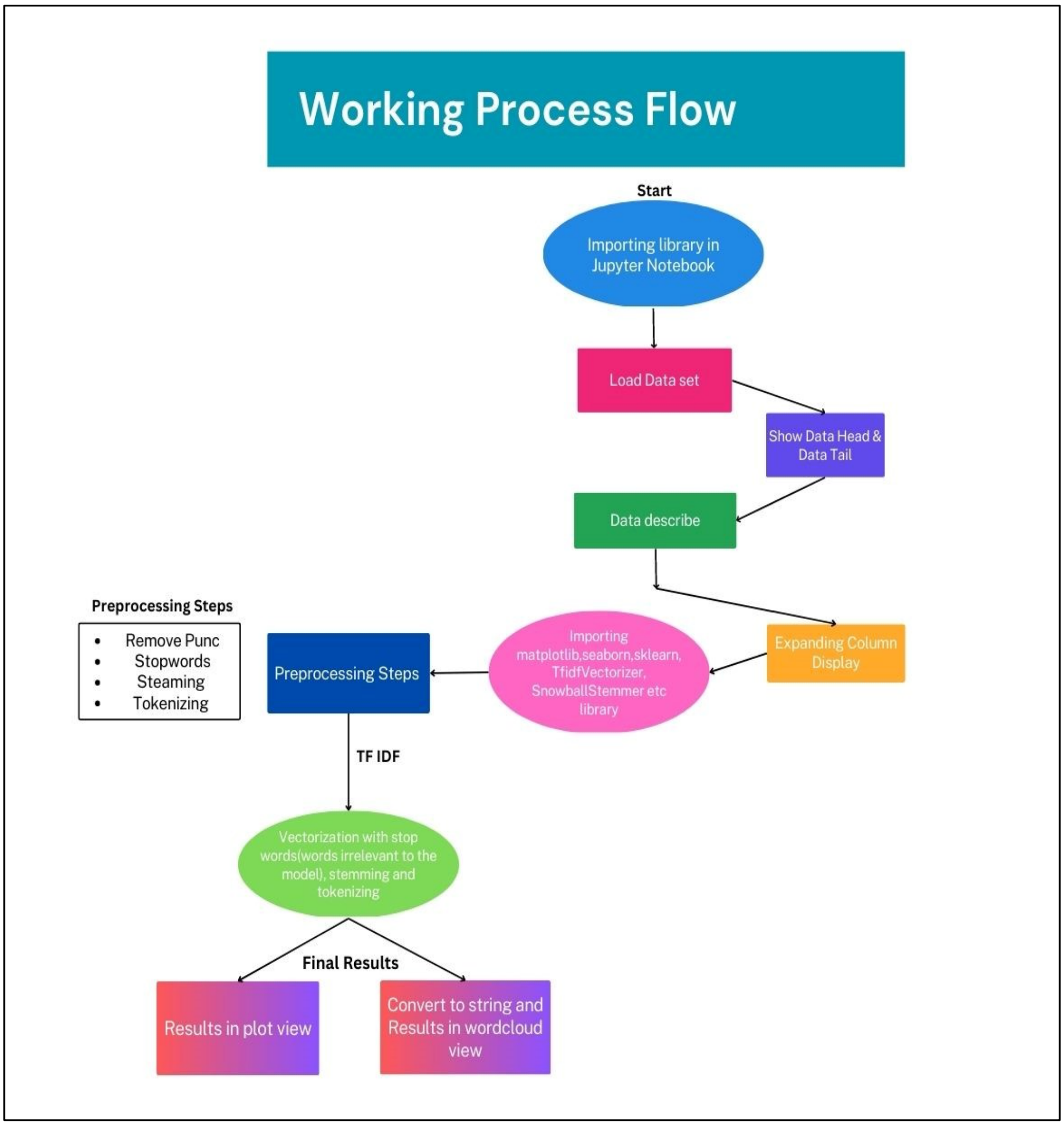

Working Process Flow
Start
Importing library in Jupyter Notebook
Load Data set
Show Data Head & Data Tail
Data describe
Expanding Column Display
Importing matplotlib,seaborn,sklearn, TfidfVectorizer, SnowballStemmer etc library
Preprocessing Steps
Preprocessing Steps
Remove Punc
Stopwords
Steaming
Tokenizing
TF IDF
Vectorization with stop words(words irrelevant to the model), stemming and tokenizing
Final Results
Results in plot view
Convert to string and Results in wordcloud view

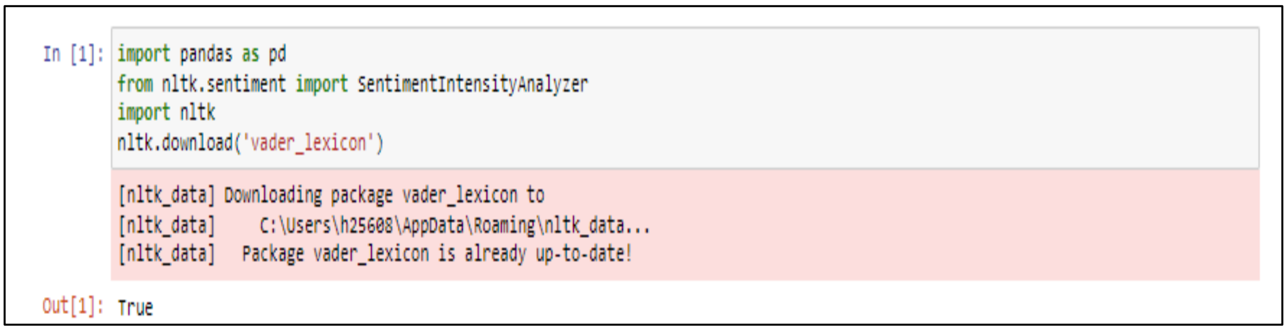
FIGURE 3. Working Process Flow

## 4.1 Environment setup

Using a Jupyter Notebook is the recommended approach to run the project. An open-source web tool called Jupyter Notebook enables users to create and share documents with real-time code, equations, visuals, and text. It offers a computer environment that is interactive and supports many programming languages, including Python, R, Julia, and others. Installing Jupyter Notebook locally or using cloud-based services like JupyterLab or Google Colab are the only options for using it. As soon as it is installed, one may run the notebook server, start new notebooks, and begin coding and documenting data science projects in a flexible and interactive environment.

## 4.2 Importing the libraries

To start the implementation process in Jupyter notebook, need to import libraries.



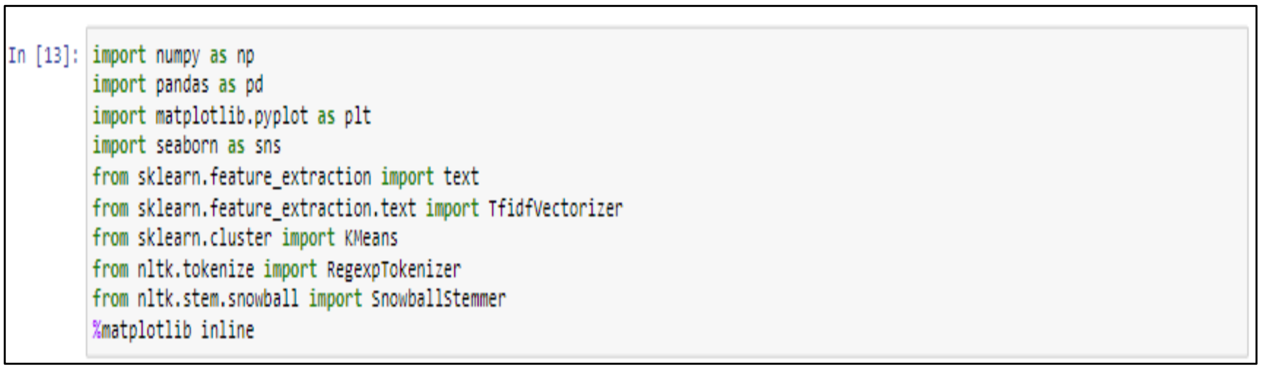
```
In [1]: import pandas as pd
        from nltk.sentiment import SentimentIntensityAnalyzer
        import nltk
        nltk.download('vader_lexicon')

        [nltk_data] Downloading package vader_lexicon to
        [nltk_data]     C:\Users\h25608\AppData\Roaming\nltk_data...
        [nltk_data]   Package vader_lexicon is already up-to-date!

Out[1]: True
```

CODE 1. Importing the Libraries

The CODE 1 describes as follows, import pandas as pd: This line imports the Pandas library and renames it as "pd" for ease of use. from nltk.sentiment import SentimentIntensityAnalyzer: This line imports the SentimentIntensityAnalyzer class from the NLTK library, which is used to perform sentiment analysis on text data.import nltk: This line imports the NLTK library, which provides tools for natural language processing.  nltk.download('vader_lexicon'): With this line, NLTK's pre-built sentiment analysis tool, the VADER lexicon, is downloaded. Based on a word or phrase's polarity (positive or negative) and intensity, the Vader lexicon assigns sentiment scores to it.



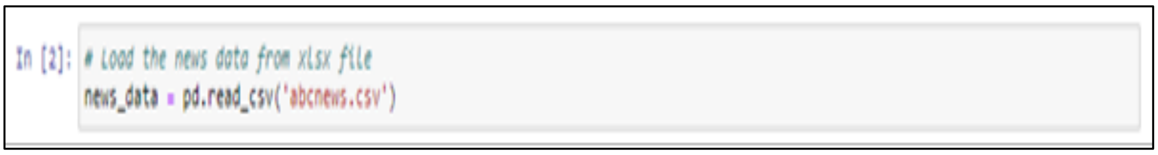
```
In [13]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.feature_extraction import text
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.cluster import KMeans
         from nltk.tokenize import RegexpTokenizer
         from nltk.stem.snowball import SnowballStemmer
         %matplotlib inline
```

CODE 2. Importing the Libraries

The CODE 2 describes as follows, import numpy as np: This line imports the NumPy library, which provides support for arrays and matrices. import pandas as pd: This line imports the Pandas library, which provides support for data manipulation and analysis. import matplotlib.pyplot as plt: This line imports the Matplotlib library, which provides support for creating visualizations. import seaborn as sns: This line imports the Seaborn library, which provides additional support for creating visualizations. from sklearn.feature_extraction import text: This line imports the text module from the Scikit-learn library, which provides support for text analysis. from sklearn.feature_extraction.text import TfidfVectorizer: This line imports the Scikit-Learn library's TfidfVectorizer class, which is used to convert text data into a matrix of TF-IDF characteristics. from sklearn.cluster import KMeans: This line imports the KMeans class from the Scikit-learn library, which is used to perform k-means clustering. from nltk.tokenize import RegexpTokenizer: This line imports the RegexpTokenizer class from the NLTK library, which is used to tokenize text into individual words. from nltk.stem.snowball import SnowballStemmer: This line imports the SnowballStemmer class from the NLTK library, which is used to perform stemming on words. %matplotlib inline: This line is a magic command for Jupyter notebooks that enables inline plotting of visualizations.

## 4.3 Load the news data from csv file

The process of gathering project-related data from diverse resources is known as data collection. This is the first step of the suggested procedure. The data in this work is of the text kind. And after collecting the expected data we load it to the system.

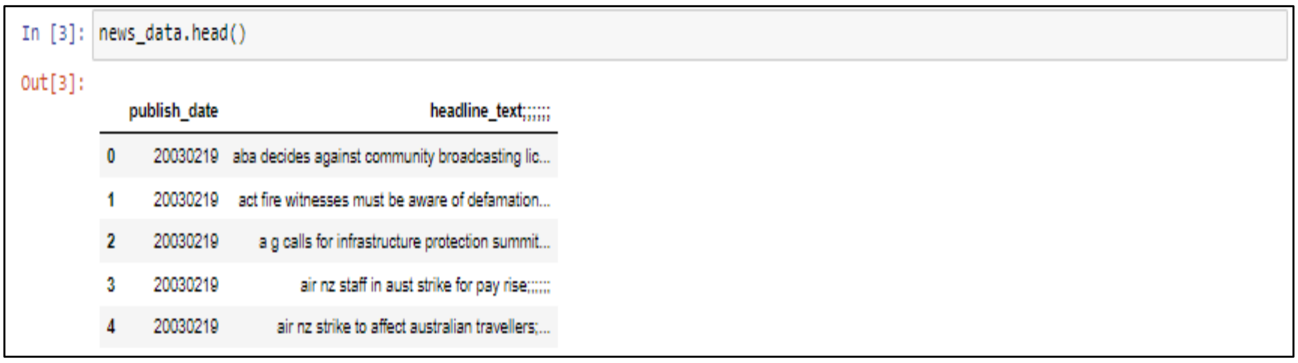


```
In [2]: # Load the news data from xlsx file
        news_data = pd.read_csv('abcnews.csv')
```

CODE 3. Load News Data

The CODE 3 describes as follows, pd is a reference to the Pandas library that was imported earlier. read_csv (), this is a function in the Pandas library that reads a CSV file and returns a DataFrame object. 'abcnews.csv'. This is the name of the CSV file to be loaded. It should be in the same directory as the Python script. news_data. This is the name given to the Pandas DataFrame object that will hold the data from the CSV file.

## 4.4 Displaying the few first lines of the dataset



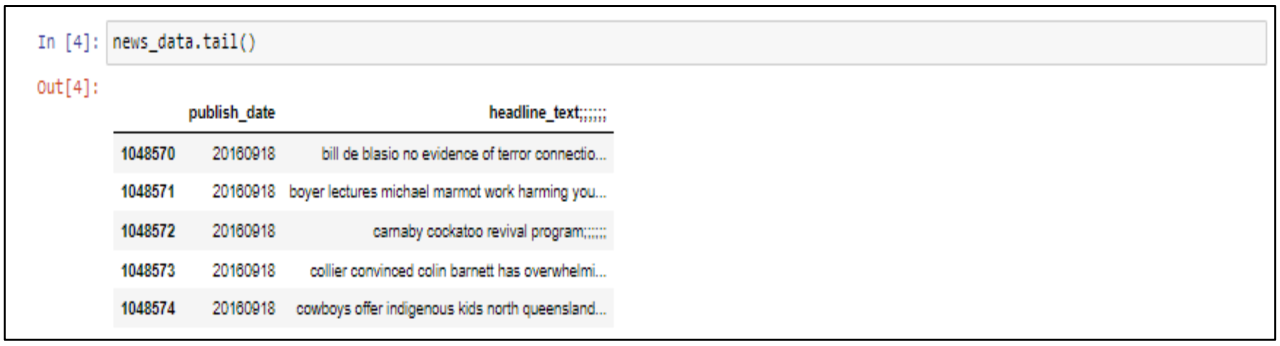
```
In [3]: news_data.head()
Out[3]:
```

| | publish_date | headline_text;;;;;; |
|---|---|---|
| 0 | 20030219 | aba decides against community broadcasting lic... |
| 1 | 20030219 | act fire witnesses must be aware of defamation... |
| 2 | 20030219 | a g calls for infrastructure protection summit... |
| 3 | 20030219 | air nz staff in aust strike for pay rise;;;;;; |
| 4 | 20030219 | air nz strike to affect australian travellers;... |

CODE 4. Display Head Data

The CODE 4 describes as follows, news_data. This is the name of the Pandas DataFrame object that was created earlier. head (10). This is a method in the Pandas library that returns the first 10 rows of the DataFrame. If no argument is passed to the method, it returns the first 5 rows by default.

## 4.5 Displaying the few last lines of the dataset

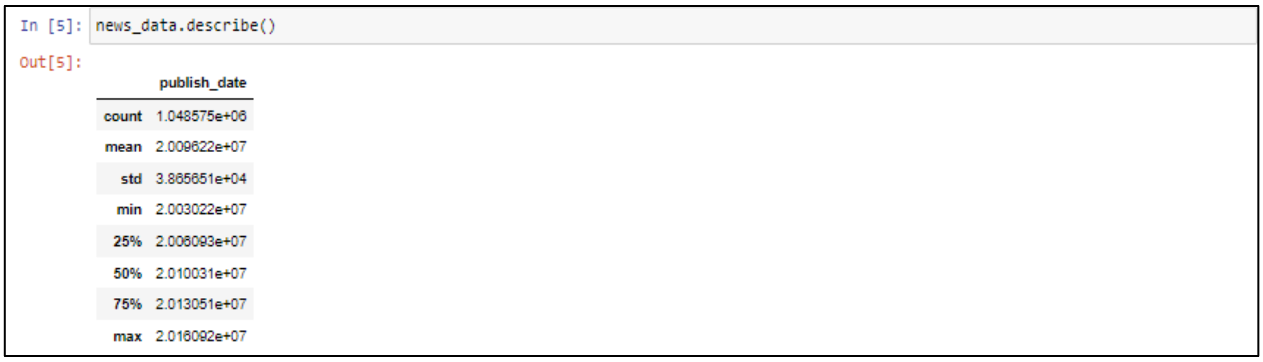


```
In [4]: news_data.tail()
Out[4]:
         publish_date                               headline_text;;;;;
1048570      20160918        bill de blasio no evidence of terror connectio...
1048571      20160918   boyer lectures michael marmot work harming you...
1048572      20160918                       carnaby cockatoo revival program;;;;;
1048573      20160918       collier convinced colin barnett has overwhelmi...
1048574      20160918      cowboys offer indigenous kids north queensland...
```

CODE 5. Display Tail Data

The CODE 5 describes as follows, news_data. This is the name of the Pandas DataFrame object that was created earlier. tail(). This is a method in the Pandas library that returns the last 5 rows of the DataFrame. If an argument is passed to the method, it returns the last n rows.

## 4.6 Describing the dataset

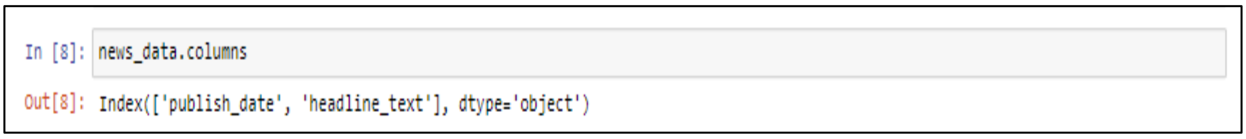


```
In [5]: news_data.describe()
Out[5]:
       publish_date
count  1.048575e+06
 mean  2.009622e+07
  std  3.865651e+04
  min  2.003022e+07
  25%  2.006093e+07
  50%  2.010031e+07
  75%  2.013051e+07
  max  2.016092e+07
```

CODE 6. Describing Dataset

The CODE 6 describes as follows, news_data. This is the name of the Pandas DataFrame object that was created earlier. describe (). This is a method in the Pandas library that returns a statistical summary of the DataFrame. It includes the count, mean, standard deviation, minimum, and maximum values for each numeric column in the DataFrame.

## 4.7 Column-wise description of the data



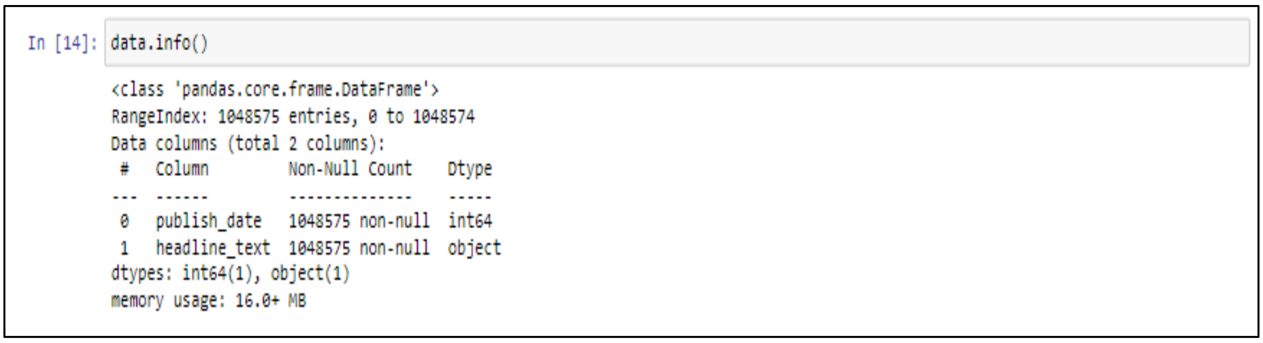
```
In [8]: news_data.columns

Out[8]: Index(['publish_date', 'headline_text'], dtype='object')
```

CODE 7. Column wise Description

The CODE 7 describes as follows, news_data. This is the name of the Pandas DataFrame object that was created earlier. columns. This is an attribute in the Pandas library that returns the column names of a DataFrame.

## 4.8 Display the Information on data



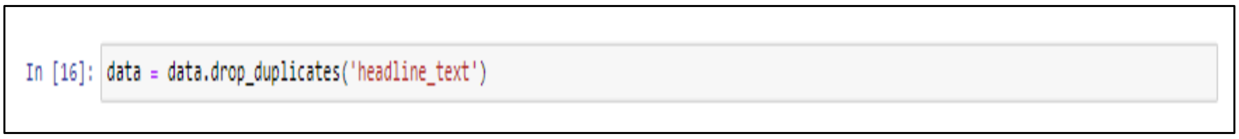
```
In [14]: data.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 1048575 entries, 0 to 1048574
         Data columns (total 2 columns):
          #   Column         Non-Null Count    Dtype
         ---  ------         --------------    -----
          0   publish_date   1048575 non-null  int64
          1   headline_text  1048575 non-null  object
         dtypes: int64(1), object(1)
         memory usage: 16.0+ MB
```

CODE 8. Display the Information Data

The CODE 8 describes as follows, news_data This is the name of the Pandas DataFrame object that was created earlier. info (). This is a method in the Pandas library that provides a summary of the Data Frame's metadata, including the number of non-null values in each column, the data type of each column, and the memory usage of the DataFrame.

## 4.9 Deleting duplicate headlines (if any)



```
In [16]: data = data.drop_duplicates('headline_text')
```

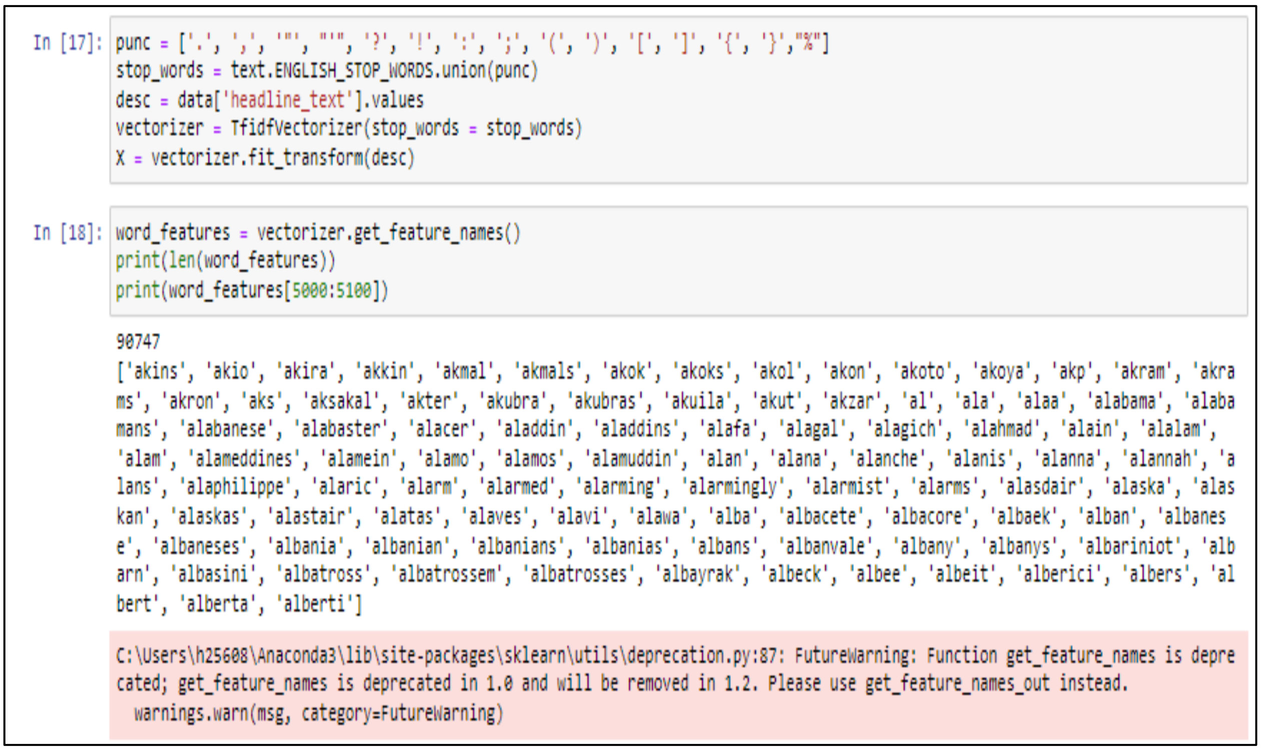
CODE 9. Deleting Duplicate Headlines

The CODE 9 describes as follows, news_data. This is the name of the Pandas DataFrame object that was created earlier. .drop_duplicates('headline_text').This drops duplicates from the DataFrame based on the 'headline_text' column using the drop_duplicates() method. data =. This assigns the resulting DataFrame back to the variable 'data'.

## 4.10 Application of NLP

Preparing data for vectorization: Vectorization is used to speed up the Python code without using loop. Using such a function can help in minimizing the running time of code efficiently. Various operations are being performed over vector such as dot product of vectors which is also known as scalar product as it produces single output, outer products which results in square matrix of dimension equal to length X length of the vectors, Element wise multiplication which products the element of same indexes and dimension of the matrix remain unchanged. It helps to convert the text data into numbers. This process is sometimes referred to as “embedding” or “vectorization”. In order to perform machine learning on text, we need to transform our documents into vector representations such that we can apply numeric machine learning. This process is called feature extraction or more simply, vectorization, and is an essential first step toward language-aware analysis. Representing documents numerically gives us the ability to perform meaningful analytics and also creates the instances on which machine learning algorithms operate. In text analysis, instances are entire documents or utterances, which can vary in length from quotes or tweets to entire books, but whose vectors are always of a uniform length. Each property of the vector representation is a feature. For text, features represent attributes and properties of documents—including its content as well as meta attributes, such as document length, author, source, and publication date. When considered together, the features of a document describe a multidimensional feature space on which machine learning methods can be applied. For this reason, must now make a critical shift in how we think about language from a sequence of words to points that occupy a high-dimensional semantic space. Points in space can be close together or far apart, tightly clustered or evenly distributed. Semantic space is therefore mapped in such a way where documents with similar meanings are closer together and those that are different are farther apart. By encoding similarity as distance, we can begin to derive the primary components of documents and draw decision boundaries in our semantic space. The simplest encoding of semantic space is the bag-of-words model, whose primary insight is that meaning and similarity are encoded in vocabulary.

Term Frequency–inverse document frequency (TF-IDF): In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modelling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Nowadays, tf-idf is one of the most popular term-weighting schemes; 83% of text-based recommender systems in the domain of digital libraries use tf-idf. (Qiaoyan Kuang; Xiaoming Xu 2010). Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.



```
In [17]: punc = ['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}',"%"]
         stop_words = text.ENGLISH_STOP_WORDS.union(punc)
         desc = data['headline_text'].values
         vectorizer = TfidfVectorizer(stop_words = stop_words)
         X = vectorizer.fit_transform(desc)

In [18]: word_features = vectorizer.get_feature_names()
         print(len(word_features))
         print(word_features[5000:5100])

         90747
         ['akins', 'akio', 'akira', 'akkin', 'akmal', 'akmals', 'akok', 'akoks', 'akol', 'akon', 'akoto', 'akoya', 'akp', 'akram', 'akra
         ms', 'akron', 'aks', 'aksakal', 'akter', 'akubra', 'akubras', 'akuila', 'akut', 'akzar', 'al', 'ala', 'alaa', 'alabama', 'alaba
         mans', 'alabanese', 'alabaster', 'alacer', 'aladdin', 'aladdins', 'alafa', 'alagal', 'alagich', 'alahmad', 'alain', 'alalam',
         'alam', 'alameddines', 'alamein', 'alamo', 'alamos', 'alamuddin', 'alan', 'alana', 'alanche', 'alanis', 'alanna', 'alannah', 'a
         lans', 'alaphilippe', 'alaric', 'alarm', 'alarmed', 'alarming', 'alarmingly', 'alarmist', 'alarms', 'alasdair', 'alaska', 'alas
         kan', 'alaskas', 'alastair', 'alatas', 'alaves', 'alavi', 'alawa', 'alba', 'albacete', 'albacore', 'albaek', 'alban', 'albanes
         e', 'albaneses', 'albania', 'albanian', 'albanians', 'albanias', 'albans', 'albanvale', 'albany', 'albanys', 'albariniot', 'alb
         arn', 'albasini', 'albatross', 'albatrossem', 'albatrosses', 'albayrak', 'albeck', 'albee', 'albeit', 'alberici', 'albers', 'al
         bert', 'alberta', 'alberti']

         C:\Users\h25608\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is depre
         cated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
           warnings.warn(msg, category=FutureWarning)
```

CODE 10. Preprocessing Steps

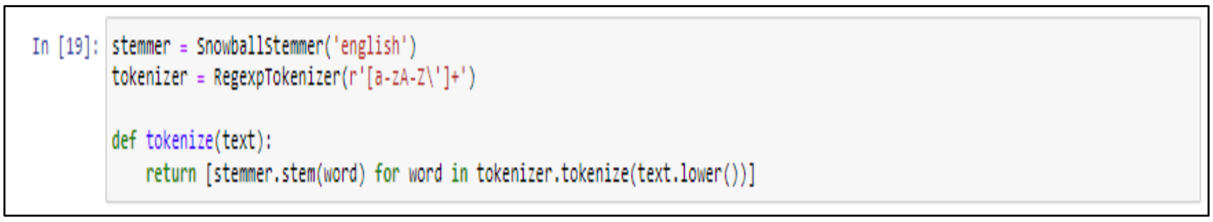
The CODE 10 describes as follows, punc. This is a list of common punctuation marks that are often excluded from text analysis. stop_words = text.ENGLISH_STOP_WORDS.union(punc). This combines the list of English stop words from the text module of the Scikit-learn library with the list of punctuation marks to create a new list of stop words to exclude from analysis. desc = data['headline_text']. Values. This creates a NumPy array of the headlines from the 'headline_text' column of the 'news_data' Data-Frame. vectorizer = TfidfVectorizer(stop_words = stop_words). This creates a TF-IDF vectorizer object from the Scikit-learn library with the custom list of stop words to use during analysis. X = vectorizer.fit_transform(desc). This applies the vectorizer to the corpus of headlines to create a sparse matrix of TF-IDF features for each headline. word_features = vectorizer.get_feature_names().This retrieves the list of unique words (features) from the TF-IDF vectorizer object created earlier and assigns it to the variable 'word_features'. print(len(word_features)). This prints out the total number of unique words (features) in the TF-IDF matrix. print(word_features[5000:5100]). This prints out a subset of the feature names, specifically the names from index 5000 to 5100.

## 4.11 Stemming, Vectorization and Tokenization

In natural language processing, stemming, tokenization, and vectorization are essential approaches. Tokenization divides text into smaller pieces, stemming reduces words to their base form, and vectorization turns textual information into numerical representations. Combining these methods allows us to process, evaluate, and get valuable insights from textual data for a variety of applications, including text categorization, sentiment analysis, and information retrieval.

### 4.11.1 Stemming

Stemming is the process of reducing words to their root or base form, known as the stem. This is useful for tasks such as text classification, where the goal is to identify the underlying meaning of the text, rather than the specific form of the words used. For example, the word "running" might be stemmed to "run", and the word "jumping" might be stemmed to "jump". Stemming can be accomplished using various algorithms, such as the Porter stemming algorithm, the Snowball stemming algorithm, or the Lancaster stemming algorithm. Stemming is the process of reducing a word into its stem, i.e., its root form. The root form is not necessarily a word by itself, but it can be used to generate words by concatenating the right suffix. For example, the words fish, fishes and fishing all stem into fish, which is a correct word. On the other side, the words study, studies and studying stems into study, which is not an English word (R. Méndez, E. L. Iglesias, F. Fdez-Riverola, F. Díaz & J. M. Corchado 2006).



```
In [19]: stemmer = SnowballStemmer('english')
         tokenizer = RegexpTokenizer(r'[a-zA-Z\']+')

         def tokenize(text):
             return [stemmer.stem(word) for word in tokenizer.tokenize(text.lower())]
```

CODE 11. Stemming

The CODE 11 describes as follows, stemmer = SnowballStemmer('english'). This creates a SnowballStemmer object from the NLTK library for the English language and assigns it to the variable 'stemmer'. tokenizer = RegexpTokenizer(r'[a-zA-Z\'] +'). This creates a RegexpTokenizer object from the NLTK library that matches all alphabetical characters and apostrophes and assigns it to the variable 'tokenizer'.def tokenize (text). This defines a new function called 'tokenize' that takes in a string of text as input. return [stemmer.stem(word) for word in tokenizer.tokenize(text.lower())]. This function splits the text into tokens using the RegexpTokenizer object, converts the tokens to lowercase, and applies stemming using the SnowballStemmer object. The resulting list of stemmed tokens is returned.

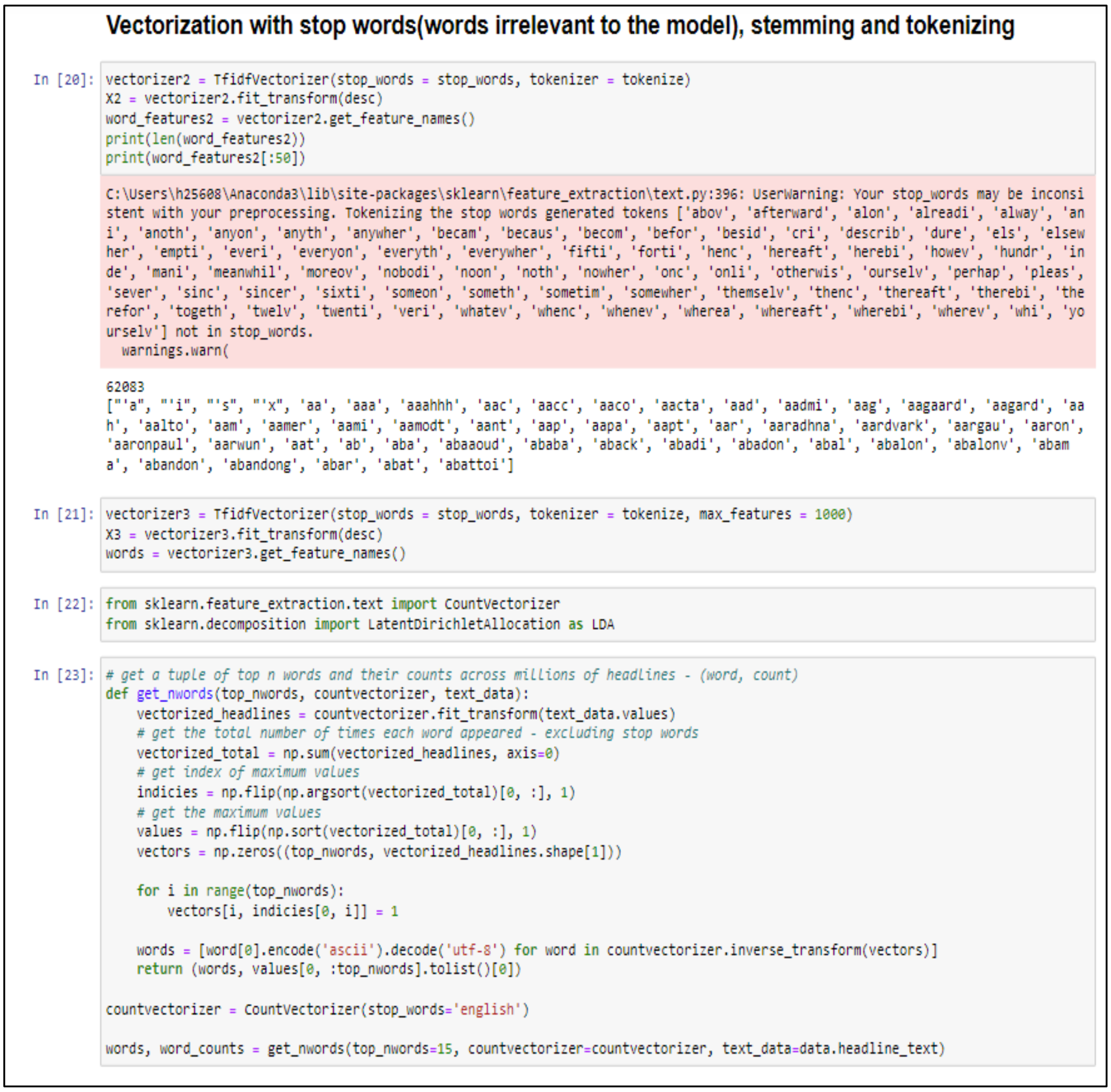
### 4.11.2 Vectorization

Vectorization in NLP refers to the process of converting text data into numerical vectors that can be used for machine learning algorithms or other mathematical operations. To perform most machine learning tasks on text data, it needs to be represented in a numerical form, as most machine learning algorithms require numerical inputs. Vectorization involves converting each document or sentence in a corpus of text into a vector of numerical features, where each feature represents some aspect of the text, such as the frequency of a particular word or the presence of a particular grammatical structure. The resulting vectors can then be used as inputs to machine learning algorithms, such as clustering, classification, or regression.

There are several different approaches to vectorization in NLP, including bag-of-words models, term frequency-inverse document frequency (TF-IDF) models, and word embedding models. Bag-of-words models represent each document as a vector of word counts, while TF-IDF models weight each word in a document by its importance in the overall corpus. Word embedding models represent each word in a document as a dense vector of continuous values, which capture semantic relationships between words. Overall, vectorization is a key preprocessing step in NLP, as it enables the use of machine learning algorithms on text data, allowing for tasks such as sentiment analysis, topic modelling, and natural language generation.

### 4.11.3 Tokenization

Tokenization is the process of breaking up text into smaller units, or tokens, such as individual words or punctuation marks. This is useful for tasks such as text analysis and information retrieval, where the goal is to extract specific information from the text. Tokenization can be accomplished using various techniques, such as whitespace tokenization, which splits the text on whitespace characters, or regular expression tokenization, which uses regular expressions to split the text based on more complex patterns. (R. Méndez, E. L. Iglesias, F. Fdez-Riverola, F. Díaz & J. M. Corchado 2006).



**Vectorization with stop words(words irrelevant to the model), stemming and tokenizing**

```
In [20]: vectorizer2 = TfidfVectorizer(stop_words = stop_words, tokenizer = tokenize)
         X2 = vectorizer2.fit_transform(desc)
         word_features2 = vectorizer2.get_feature_names()
         print(len(word_features2))
         print(word_features2[:50])
```

```
C:\Users\h25608\Anaconda3\lib\site-packages\sklearn\feature_extraction\text.py:396: UserWarning: Your stop_words may be inconsi
stent with your preprocessing. Tokenizing the stop words generated tokens ['abov', 'afterward', 'alon', 'alreadi', 'alway', 'an
i', 'anoth', 'anyon', 'anyth', 'anywher', 'becam', 'becaus', 'becom', 'befor', 'besid', 'cri', 'describ', 'dure', 'els', 'elsew
her', 'empti', 'everi', 'everyon', 'everyth', 'everywher', 'fifti', 'forti', 'henc', 'hereaft', 'herebi', 'howev', 'hundr', 'in
de', 'mani', 'meanwhil', 'moreov', 'nobodi', 'noon', 'noth', 'nowher', 'onc', 'onli', 'otherwis', 'ourselv', 'perhap', 'pleas',
'sever', 'sinc', 'sincer', 'sixti', 'someon', 'someth', 'sometim', 'somewher', 'themselv', 'thenc', 'thereaft', 'therebi', 'the
refor', 'togeth', 'twelv', 'twenti', 'veri', 'whatev', 'whenc', 'whenev', 'wherea', 'whereaft', 'wherebi', 'wherev', 'whi', 'yo
urselv'] not in stop_words.
  warnings.warn(

62083
["'a", "'i", "'s", "'x", 'aa', 'aaa', 'aaahhh', 'aac', 'aacc', 'aaco', 'aacta', 'aad', 'aadmi', 'aag', 'aagaard', 'aagard', 'aa
h', 'aalto', 'aam', 'aamer', 'aami', 'aamodt', 'aant', 'aap', 'aapa', 'aapt', 'aar', 'aaradhna', 'aardvark', 'aargau', 'aaron',
'aaronpaul', 'aarwun', 'aat', 'ab', 'aba', 'abaaoud', 'ababa', 'aback', 'abadi', 'abadon', 'abal', 'abalon', 'abalonv', 'abam
a', 'abandon', 'abandong', 'abar', 'abat', 'abattoi']
```

```
In [21]: vectorizer3 = TfidfVectorizer(stop_words = stop_words, tokenizer = tokenize, max_features = 1000)
         X3 = vectorizer3.fit_transform(desc)
         words = vectorizer3.get_feature_names()
```

```
In [22]: from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.decomposition import LatentDirichletAllocation as LDA
```

```
In [23]: # get a tuple of top n words and their counts across millions of headlines - (word, count)
         def get_nwords(top_nwords, countvectorizer, text_data):
             vectorized_headlines = countvectorizer.fit_transform(text_data.values)
             # get the total number of times each word appeared - excluding stop words
             vectorized_total = np.sum(vectorized_headlines, axis=0)
             # get index of maximum values
             indicies = np.flip(np.argsort(vectorized_total)[0, :], 1)
             # get the maximum values
             values = np.flip(np.sort(vectorized_total)[0, :], 1)
             vectors = np.zeros((top_nwords, vectorized_headlines.shape[1]))

             for i in range(top_nwords):
                 vectors[i, indicies[0, i]] = 1

             words = [word[0].encode('ascii').decode('utf-8') for word in countvectorizer.inverse_transform(vectors)]
             return (words, values[0, :top_nwords].tolist()[0])

         countvectorizer = CountVectorizer(stop_words='english')

         words, word_counts = get_nwords(top_nwords=15, countvectorizer=countvectorizer, text_data=data.headline_text)
```

CODE 12. Vectorization with Stop words

The CODE 12 describes as follows, the above code creates a new TF-IDF vectorizer object with the following parameters: 'stop_words' set to the 'stop_words' variable, 'tokenizer' set to the 'tokenize' function, and 'max_features' set to 1000. This means that the vectorizer will only consider the top 1000 most frequently occurring words in the headlines for analysis. Then, the new vectorizer object is applied to the 'desc' variable using the fit_transform method, and the resulting matrix is assigned to the variable 'X3'. Finally, the list of 1000 most frequently occurring words (features) is retrieved from the vectorizer object using the get_feature_names() method and assigned to the variable 'words'. By setting 'max_features' to 1000, this code limits the number of features used for analysis to the top 1000 most frequent words, which can improve the efficiency and accuracy of the analysis. CountVectorizer is used to convert a collection of text documents to a matrix of token counts, which can then be used to train a machine learning model.

LatentDirichletAllocation is a machine learning algorithm used for topic modelling, which involves identifying the underlying topics that exist within a collection of documents. In the context of this project, these classes can be used to perform topic modelling on the news headlines data, to identify the topics that are most frequently discussed over the 19-year period. In the CODE 12, define a function called get_nwords that takes three arguments, top_nwords, countvectorizer, and text_data. top_nwords specifies the number of top words to be extracted. countvectorizer is an instance of the CountVectorizer class from scikit-learn library that is used to transform the text data into a matrix of token counts. text_data is the text data on which the function will operate. Inside the function, the vectorized_headlines variable is initialized by transforming the text_data with the countvectorizer. Next, vectorized_total calculates the total number of times each word appeared in the text data, excluding the stop words. The indices variable contains the index of the maximum values in the vectorized_total array, which are the most frequent words in the text data. values variable stores the maximum values. Finally, the function loops through the top_nwords number of words and sets their values to 1 in the vectors variable. The words and their corresponding counts are then returned. In the last line of the code, the get_nwords function is called with the arguments top_nwords=15, countvectorizer=countvectorizer, and text_data=data.headline_text to obtain the top 15 most frequent words and their counts.

# 5 RESULTS AND DISCUSSION

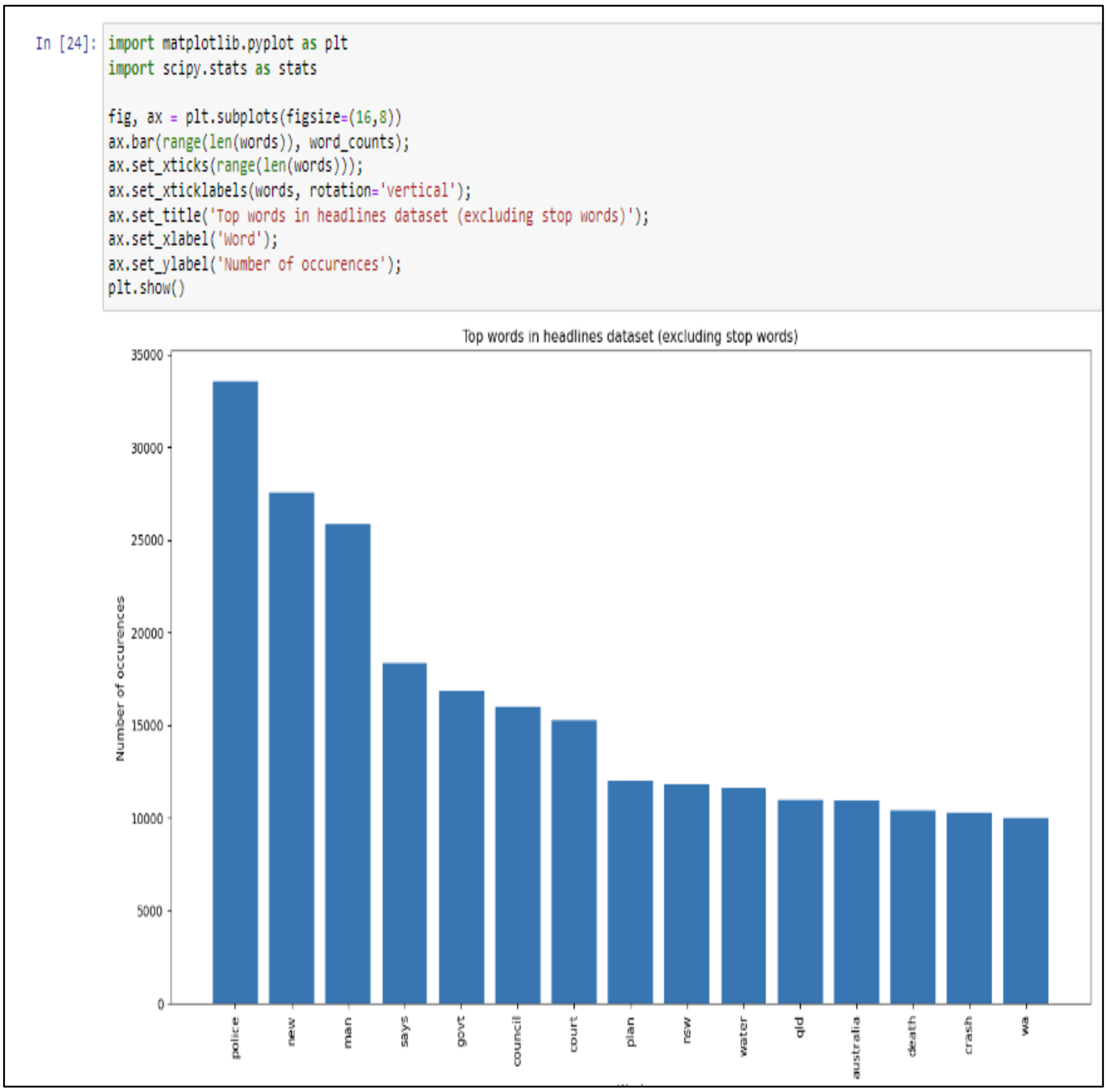
Visualization is done using two python Libraries matplotlib and WordCloud.

## 5.1 Visualization using matplotlib library

Matplotlib is a Python library used for creating visualizations and plots in various formats, such as line plots, scatter plots, bar plots, and histograms. It was originally created by John Hunter in 2003 and has since become a widely used and popular tool for data visualization in Python. It provides a wide range of tools and functions for creating high-quality plots, charts, and graphs to represent and analyze data. Matplotlib is highly customizable and offers a variety of plot types, allowing users to create visually appealing and informative visualizations. Matplotlib have extensive functionality, customization options, and integration with other scientific libraries. Matplotlib integrates seamlessly with NumPy, a numerical computing library, and Pandas, a data manipulation library. It also supports a range of file formats for exporting plots, such as PNG (pronounced ping), PDF, and SVG (Scalable Vector Graphics). One of the key features of Matplotlib is its ability to create complex and interactive visualizations, such as heatmaps, contour plots, and animations (Niyazi Ari 2014). Matplotlib also integrates well with other Python libraries, such as NumPy, Pandas, and Seaborn, making it a powerful tool for data analysis and exploration. Matplotlib is widely used in many fields, including data science, finance, and engineering, and is supported by a large community of contributors and users. Its versatility and ease of use make it a valuable tool for creating visualizations and exploring data in Python. Matplotlib supports interactive features to enhance the exploration of data. Users can add interactive elements like tooltips, zooming, panning, picking, and annotations to make plots more interactive and engaging. Users can animate plots to demonstrate changes over time or iterate through different views of the data, making it useful for presentations and conveying dynamic information.

CODE 13 shows a bar chart depicting the frequency of popular words. The x-axis represents the words, while the y-axis represents the frequency of occurrence. The bars are color-coded with blue and scaled according to the frequency, with the highest frequency words having the tallest bars. In the below figure, the chart clearly shows that the most frequent words are [insert words with highest frequency], while the least frequent words are [insert words with lowest frequency]. Overall, the chart provides a visual representation of the

most used words in the given dataset.



```
In [24]: import matplotlib.pyplot as plt
         import scipy.stats as stats

         fig, ax = plt.subplots(figsize=(16,8))
         ax.bar(range(len(words)), word_counts);
         ax.set_xticks(range(len(words)));
         ax.set_xticklabels(words, rotation='vertical');
         ax.set_title('Top words in headlines dataset (excluding stop words)');
         ax.set_xlabel('Word');
         ax.set_ylabel('Number of occurences');
         plt.show()
```

CODE 13. Matplotlib Visualization

The CODE 13 describes as follows, this code block creates a bar chart to visualize the top words in the headline's dataset, excluding stop words. The words and their corresponding frequencies are obtained from the words and word_counts variables, respectively, which were generated using the get_nwords() function earlier.

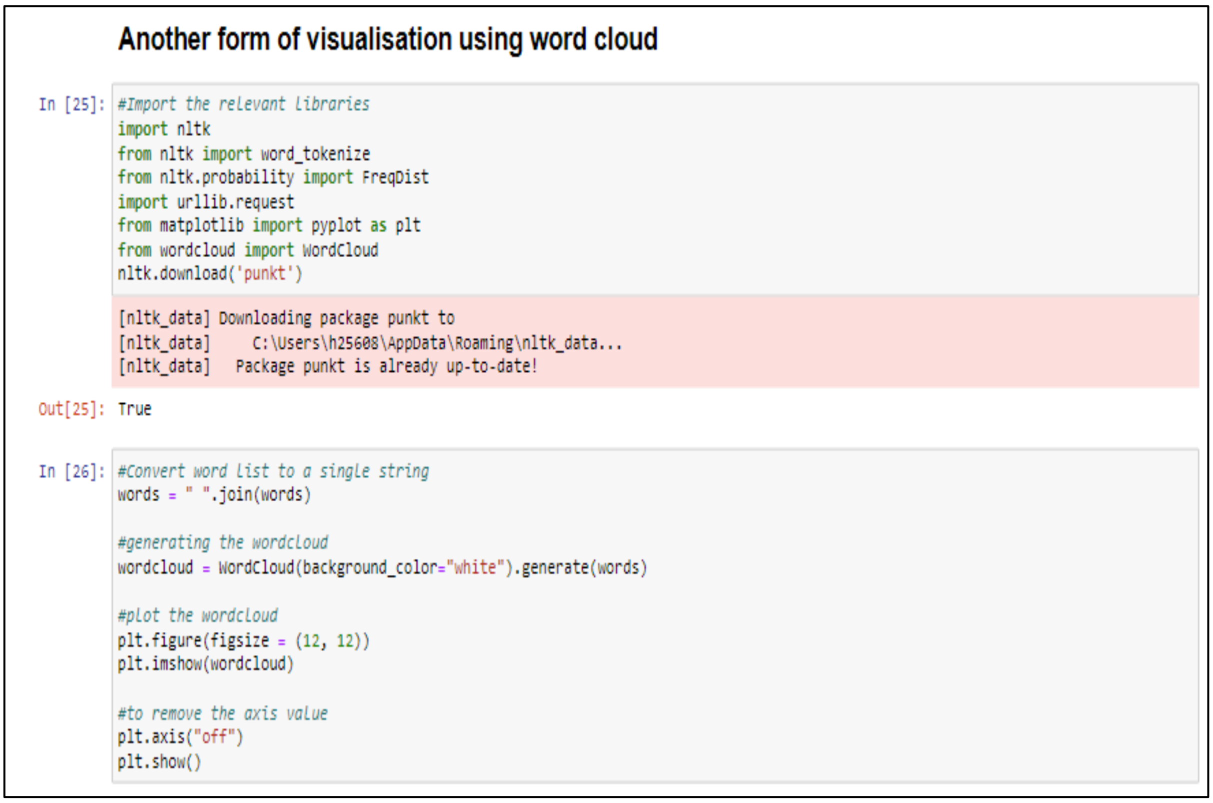
The figure, ax = plt.subplots(figsize= (16,8)) line initializes a figure and axis object with a size of 16x8 inches. The ax.bar(range(len(words)), word_counts) line creates a vertical bar chart with the range(len(words)) as the x-axis and the word_counts as the y-axis values. The ax.set_xticks(range(len(words))) and ax.set_xticklabels(words, rotation='vertical') lines set the x-axis tick locations and labels, respectively. The ax.set_title(), ax.set_xlabel(), and ax.set_ylabel() functions set the chart title, x-label, and y-label, respectively. Finally, the plt.show() line displays the chart.

## 5.2 Visualization using WordCloud

WordCloud is a Python library for creating word clouds, which are visual representations of the frequency of words in a text corpus. WordCloud provides a simple and flexible interface for generating word clouds from text data, such as articles, reviews, or social media posts. Word clouds are commonly used for exploratory data analysis and to gain insights from text data. They can reveal patterns, trends, and prominent themes within a collection of text documents.

The library allows you to customize the appearance of the word cloud, including the size, shape, color, and font of the words. WordCloud can also mask the words to create word clouds in custom shapes, such as logos or silhouettes. To use WordCloud, you first need to import the library and any other libraries that you may need, such as Numpy and Matplotlib. Then, you can create a WordCloud object and pass in the text data, along with any custom parameters that you want to use, such as the maximum number of words to display or the background color (Heimerl Florian, Lohmann Steffen, Lange Simon, Erti Thomas 2014). This will create a simple word cloud image from the provided text, with each word sized according to its frequency in the text. One can customize the appearance of the word cloud by passing in additional parameters to the WordCloud object, such as the maximum font size, the minimum word length, or the color map to use. Use a word cloud library, such as WordCloud in Python, to create the visual representation. The library takes the word frequency data and generates a cloud-like image where the size of each word represents its frequency. In a word cloud, words are displayed in a graphical format, typically in a random arrangement, where the size of each word is proportional to its frequency or relevance. More frequent words appear larger and bolder, while less frequent words are smaller and less prominent. The arrangement and colouring of words in the cloud are often designed to be visually appealing.

FIGURE 4 shows a wordcloud representing the frequency of popular words in the given dataset. The wordcloud is generated using various font sizes and colors to depict the frequency of occurrence of most popular word. The most popular words are shown in larger font sizes and darker colors, while less popular words are shown in smaller font sizes and lighter colors. The wordcloud effectively captures the most used words in the dataset, with [insert words with highest frequency] being the most prominent words. Overall, the wordcloud provides an easily interpretable visualization of the frequency distribution of popular words in the dataset.



**Another form of visualisation using word cloud**

```
In [25]: #Import the relevant Libraries
         import nltk
         from nltk import word_tokenize
         from nltk.probability import FreqDist
         import urllib.request
         from matplotlib import pyplot as plt
         from wordcloud import WordCloud
         nltk.download('punkt')

         [nltk_data] Downloading package punkt to
         [nltk_data]     C:\Users\h25608\AppData\Roaming\nltk_data...
         [nltk_data]   Package punkt is already up-to-date!

Out[25]: True

In [26]: #Convert word List to a single string
         words = " ".join(words)

         #generating the wordcloud
         wordcloud = WordCloud(background_color="white").generate(words)

         #plot the wordcloud
         plt.figure(figsize = (12, 12))
         plt.imshow(wordcloud)

         #to remove the axis value
         plt.axis("off")
         plt.show()
```

CODE 14. Word Cloud Visualization

The CODE 14 describes as follows, nltk is the Natural Language Toolkit, a library for working with human language data in Python. In this code snippet, nltk is being imported along with some specific modules that will be used later. The word_tokenize function and FreqDist class are being imported for text tokenization and frequency analysis. urllib.request is being imported to download data from a URL, and pyplot from matplotlib is being imported to create visualizations. Finally, the WordCloud class is being imported from the wordcloud library to generate word clouds. The nltk.download('punkt') line is downloading the required punkt package from nltk, which is used for tokenization of text data. Wordcloud using the wordcloud module from the wordcloud library. First, the word list words are joined together into a single string using the join () method. Then, a WordCloud object is created with a white background color. The generate () method is called on this object, passing in the words string to generate the wordcloud. Finally, the imshow () method is called on a plt (matplotlib.pyplot) object, passing in the wordcloud object as the image to display. The axis () method is then called with the argument "off" to remove the axis values from the plot. The show () method is called on the plt object to display the plot.

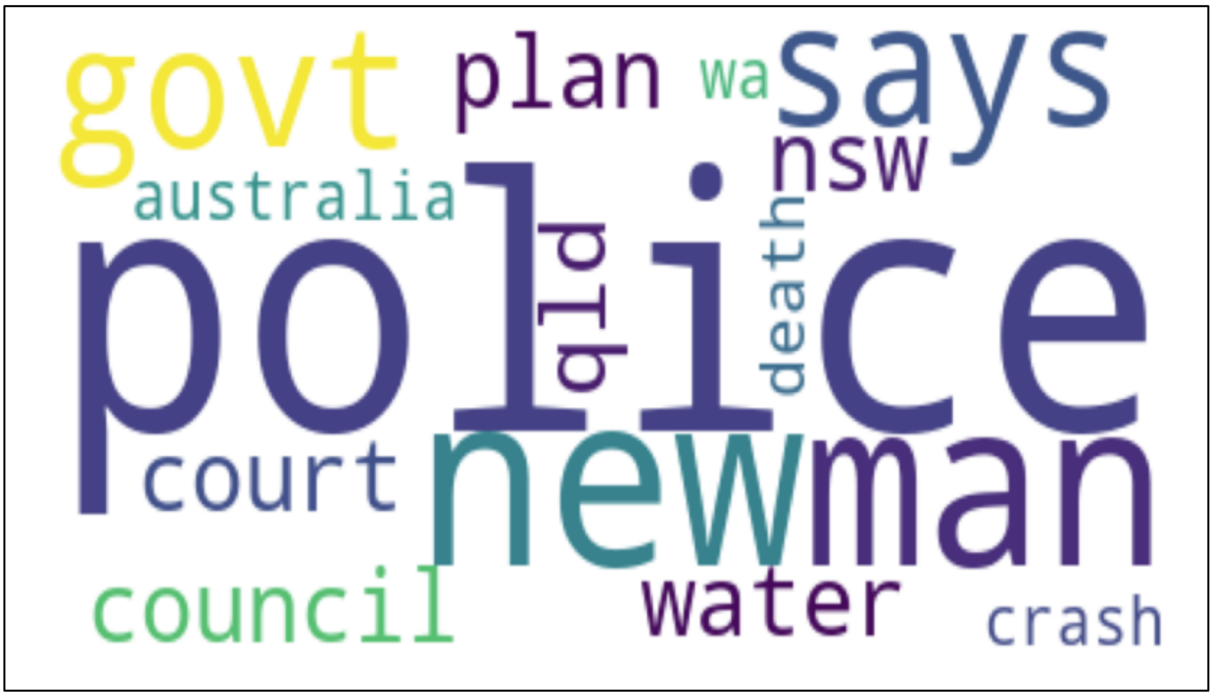


FIGURE 4. Image showing the popular word search

FIGURE 4, shows the most popular word in the nineteen-year news dataset .

# 6 DISCUSSION

The main significance of the findings of popular word search is that it can provide insights into the language used in a particular text corpus. By identifying the most commonly/popular used words or phrases, student can gain a better understanding of the themes and topics that are being discussed in the given dataset corpus. This information can be useful in a variety of contexts, such as in the analysis of social media data, market research, or political discourse. Popular word search can also help to identify key trends or patterns in the data. This information can be used to track changes in public opinion or sentiment over time, or to identify emerging trends or issues. Results show the most popular frequent words that are arise in the news headlines dataset. And the plot and wordcloud shows the frequency distribution of popular words.

# 7 FUTURE SCOPE

Using natural language processing techniques, the popular word search from a nineteen-year news dataset may be analyzed to identify structures and changes in language usage over time. The current research is based on a nineteen-year news dataset, which can be expanded in the future to include a larger data set covering a longer period. This would give a more complete picture of how language usage has changed and developed over the years. For a better comprehension of the differences and structures in language usage across multiple channels, the analysis can be compared with other datasets, such as social media information. The analysis which is happening currently is based on English. For examining the language usage changes across various languages, multiple languages research can be applied in the future. The opportunity for using natural language processing techniques to study and fully understand the changes in language usage through time is vast, and this work's potential future is significant.

# 8 CONCLUSION

In conclusion, the task of building a popular word search from a nineteen-year news dataset using Python and natural language processing (NLP) was a challenge but feasible one. The process involves several steps, including preprocessing the clues and the grid, generating candidate words, scoring candidate words, and displaying the results. Python has a variety of potential NLP libraries, such as NLTK, spaCy, and scikit-learn, which can be used to accomplish the word search analytics. Additionally, the nineteen-year dataset's large amount of news data offers an array of natural language data for building and testing NLP models. Building a popular word search analysis using NLP requires a deep understanding of NLP techniques and the ability to apply them in a creative and flexible way to solve the specific problem at hand. NLP played an important role for analysing the hidden words in a vast and diverse dataset of news articles.

## REFERENCES

Abadi (2016). *TensorFlow: learning functions at scale.* Available: https://dl.acm.org/doi/abs/10.1145/2951913.2976746 . Accessed 11.04.2023

Chaudhary M (2020). *TF-IDF Vectorizer scikit-learn.* Available: https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a . Accessed 27.03.2023

Corbo (2022). *What is Python.* Available: https://builtin.com/software-engineering-perspectives/python . Accessed 14.03.2023

Cardoso, Leitao & Teixeria (2019). *Using the Jupyter Notebook as a Tool to Support the Teaching and Learning Processes in Engineering Courses* . Available: https://link.springer.com/chapter/10.1007/978-3-030-11935-5_22 . Accessed 18.03.2023

Donoho (2017, 745–766). *'50 Years of Data Science'* Journal of Computational and Graphical Statistics.Doi:https://doi.org/10.1080/10618600.2017.1384734.
Available: https://www.tandfonline.com/doi/epdf/10.1080/10618600.2017.1384734?needAccess=true&role=button . Accessed 12.03.2023

D. Manning, Raghavan, & Schütze (2008). *Introduction to information retrieval. Cambridge University Press.* Available : https://nlp.stanford.edu/IR-book/information-retrieval-book.html .
Accessed 02.04.2023

Fan & Qin Y. (2018 May, 501-503). *Research on text classification based on improved tf-idf algorithm.*
Available:
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjaqcn46fvAhXWmosKHQ6SApsQFnoECA8QAQ&url=https%3A%2F%2Fwww.atlantis-press.com%2Farticle%2F25896557.pdf&usg=AOvVaw3DiDqNYx8xkCe4hBDhHHdw . Accessed 28.03.2023

Hardeniya, Perkins, Chopra (2016, 3) *Natural Language Processing: Python and NLTK learning path.*
Available:
https://books.google.fi/books?hl=en&lr=&id=0J_cDgAAQBAJ&oi=fnd&pg=PP1&dq=scikit+using+nlp&ots=lfwqA3jwYY&sig=Ze3XJvG59e9VDVwjZ7T-6e11CcA&redir_esc=y#v=onepage&q&f=false . Accessed 18.04.2023

Heimerl Florian , Lohmann Steffen, Lange Simon, Erti Thomas (10th March 2014). *Word Cloud Explorer: Text AnalyticsBasedon Word Clouds*. Available: https://ieeexplore.ieee.org/abstract/document/6758829/authors#authors . Accessed 17.05.2023

McKinney (2013). *Python for Data Analysis*. Available:
https://books.google.fi/books?hl=en&lr=&id=v3n4_AK8vu0C&oi=fnd&pg=PR3&dq=pandas+python+&ots=rhLH6hzumv&sig=pyHqV8hFkkWsrmXByg8oSePNbcg&redir_esc=y#v=onepage&q=pandas%20python&f=false . Accessed 09.04.2023

NumPy online (2023) . *NumPy:The fundamental package for scientific computing with Python*. Available : https://numpy.org/ . Accessed 23.04.2023

Niyazi Ari (29 December 2014), *Matplotlib in python*.
Available: https://ieeexplore.ieee.org/document/6997585/authors#authors . Accessed 10.05.2023

PyTorch online (2023*). PyTorch 2.0 - Faster, more pythonic and dynamic as ever*.
Available: https://pytorch.org/ . Accessed 19.04.2023

Petrovic, Osborne, & Lavrenko (2013, 181-192). *Streaming first story detection with application to Twitter. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.* Available: https://aclanthology.org/N10-1021.pdf . Accessed 29.03.2023

Pedregosa, Varoquaux, Gramfort, Bertrand Thirion (2011, 2825-2830) *Scikit-learn: Machine Learning in Python.* Available: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https:/ Accessed 26.03.2023

Qiaoyan Kuang, Xiaoming Xu (2010). *Improvement and Application of TF•IDF Method Based on Text Classification.* Available: https://ieeexplore.ieee.org/abstract/document/5566113 . Accessed 22.04.2023

R. Méndez, E. L. Iglesias, F. Fdez-Riverola, F. Díaz & J. M. Corchado (2006). *Tokenizing, Stemming and Stop word Removal on Anti-spam Filtering Domain J.* Available: https://link.springer.com/chapter/10.1007/11881216_47 . Accessed 05.05.2023

WordNet online (2023) . *WordNet: A Lexical Database for English*. Available: https://wordnet.princeton.edu/ . Accessed 22.04.2023