MASTER OF COMPUTER APPLICATIONS

SEMESTER 1

DATA VISUALIZATION

Unit 8

Visualisation of text data in Python

Table of Contents

1. INTRODUCTION

A word cloud is a visual amalgamation of text data depicting word frequency through font size variations and colour hues. It illustrates prominent words in more oversized, bolder typography, emphasising their frequency or significance within a text corpus. This visualisation tool simplifies data analysis and exploration by summarising key textual elements. The word cloud's components, like word size and colour, add to its interpretative power. However, it requires careful construction and consideration of its limitations, such as potential loss of context or subjectivity in design. Text data cleaning is the meticulous process of refining text stripping away irrelevant elements to enhance analysis quality. Sentiment analysis visualisation interprets this polished data, mapping emotions to discern public opinion trends through engaging graphical formats, bridging data science with intuitive insight.

1.1 Learning Objectives

By the end of this chapter, you will be able to:

- *Learn the fundamental concepts and importance of textual data analysis*
- *Outline the techniques for generating word clouds*
- *Explore the statistical aspects of frequency distributions*
- *Interpret the emotional tone of textual data through sentiment analysis visualisation*

2. WORD CLOUD

A word cloud, a tag cloud or text cloud is a visual representation of text data in which words from a given text corpus are displayed in varying sizes and colours, with more prominent words appearing larger and bolder. The size of each word is proportional to its frequency or importance within the text. Word clouds are widely used for summarising and visualising textual information, making them a valuable tool for text analysis and data exploration.

Components of a Word Cloud:

1. **Words:** The fundamental elements of a word cloud are the words themselves. These words are extracted from a given text corpus, such as a document, a collection of articles, social media comments, or any textual data source.
2. **Size:** One of the defining features of a word cloud is the variation in the size of words. The size of each word in the cloud is determined by its frequency or importance within the text. More frequent or significant words are displayed more prominent, while less common words are smaller.
3. **Colour:** Word clouds often use colours to enhance the visual representation. Colors can be applied to words based on various criteria, such as sentiment (positive vs. negative words), categories, or aesthetics.

How Word Clouds Work:

1. Text Pre-processing: Creating a word cloud typically begins with text pre-processing. This includes cleaning the text by removing punctuation, special characters, and stop words (common words like "the" and "and" that add little meaning). Tokenisation is also performed to split the text into individual words or phrases.
2. Word Frequency Calculation: After pre-processing, the frequency of each word in the text corpus is calculated. This involves counting how many times each word appears in the text. The frequency count is the basis for determining the size of words in the word cloud.
3. Word Cloud Generation: A word cloud generator uses the word frequencies to create a visual representation. This generator takes the frequency information and formats it into a visual layout. The layout places words with higher frequencies closer to the centre, while less frequent words are pushed toward the edges. The size of each word

is adjusted to reflect its frequency, making more frequent words larger and more prominent. Additionally, colours can be applied based on predefined rules or user preferences.

The advantages of word clouds:

1. *Visual Representation:*

Word clouds offer a compelling and immediate visual representation of textual data. This visual format provides several advantages:

- Sensory Engagement: Humans are naturally drawn to visuals. Using varying font sizes, colours, and spatial arrangement in word clouds engages the audience's visual senses. This engagement can enhance comprehension and memory retention.
- Effective Communication: Visual representations convey information more effectively than plain text. Users can quickly absorb and interpret the information presented in a word cloud, which is particularly valuable when dealing with large volumes of text.
- Highlighting Key Concepts: Prominent words displayed in larger fonts in a word cloud immediately draw attention to the most frequently occurring terms. This helps users identify the text's key concepts, topics, or trends.

2. *Simplicity:*

Word clouds are known for their simplicity in both creation and interpretation:

- Ease of Creation: Creating a word cloud is a straightforward process. Users typically need to pre-process the text, calculate word frequencies, and use a word cloud generator tool or library to generate the visualisation. This simplicity means that individuals with varying levels of expertise, including beginners, can create word clouds.
- User-Friendly: Word clouds are user-friendly because they require minimal technical skills. There are no complex statistical analyses or data transformations involved. As a result, they are accessible to a broad audience.

3. Quick Summary:

Word clouds excel at providing a rapid summary of textual data:

- Efficient Data Reduction: Word clouds condense a large text volume into a concise visual summary. This efficiency is especially valuable when users need to quickly assess the main themes or topics within a document.
- Identifying Trends: By highlighting the most frequently occurring words prominently, word clouds allow users to identify trends, standard terms, or keywords without the need to read through the entire text. This can be particularly helpful for decision-making and content prioritisation.
- Time-Saving: When time is limited, such as in time-critical situations or when dealing with extensive textual data, word clouds offer a time-saving alternative to reading the complete text.

Disadvantages of word cloud

1. Loss of Context:

Word clouds primarily emphasise word frequency, so they may inadvertently sacrifice the context in which words are used. Here's a more detailed explanation:

- Contextual Information: Words gain much of their meaning from the context in which they appear. However, word clouds do not consider the surrounding text or the relationships between words. As a result, words that are vital in one context may be visually downplayed in a word cloud.
- Ambiguity: Certain words can have multiple meanings or connotations, and their interpretation may depend on the context. By focusing solely on word frequency, word clouds may not distinguish between these nuances. This can lead to ambiguity in the interpretation of individual words.
- Lack of Sequence: Word clouds do not convey the text's sequence or order of words. In some cases, the order of words can significantly impact their meaning. Word clouds disregard this aspect of textual information.

2. Subjectivity:

The appearance and design of word clouds can introduce subjectivity into the interpretation. Here's a closer look:

- Word Size and Color: The size and colour choices for words in a word cloud can be subjective and influenced by the creator's preferences. This subjectivity can lead to misinterpretations or biased representations. For example, a word cloud creator might emphasise certain words based on personal opinions or goals.

- Font Style and Layout: Beyond size and colour, other design elements such as font style, layout, and orientation can also introduce subjectivity. Different design choices can convey different visual impressions, affecting how the word cloud is perceived.
- Visual Emphasis: The visual emphasis on specific words can steer the viewer's attention and interpretation. Words that are visually larger or more brightly coloured may be perceived as more important or relevant, even if their true significance in the text is different.

3. Limited Depth:

While word clouds provide a quick and accessible summary of textual data, they have limitations when it comes to depth of analysis:

- Surface-Level Understanding: Word clouds offer a surface-level understanding of textual content. They excel at identifying the most frequently occurring words but may not capture the text's nuances, subtleties, or deeper meanings.

- Complex Relationships: Word clouds do not reveal complex relationships between words or phrases. Complex linguistic patterns, metaphors, or interplay between terms often go unnoticed.

- Inability to Handle Large Texts: Word clouds may become cluttered and less informative for extensive texts. It can be challenging to display all words adequately, and this limitation can reduce their effectiveness for in-depth analysis.

Generating word cloud:

Generating a word cloud involves several steps, from data preparation to visualisation. Below are the detailed steps to create a word cloud:

1. *Data Collection:*
   - Gather the text data that you want to analyse. This could be a document, a set of documents, social media comments, or any textual data source.

2. *Text Pre-processing:*
   - Before creating a word cloud, it's essential to pre-process the text data to clean and prepare it for analysis. Common pre-processing steps include:
     - **Tokenisation:** Splitting the text into individual words or phrases.
     - **Lowercasing:** Converting all text to lowercase to ensure consistency in word frequencies (e.g., "Word" and "word" are treated as the same).
     - **Removing Punctuation:** Eliminating special characters, punctuation marks, and symbols.
     - **Stop Word Removal:** Removing common words like "the," "and," "in," which do not carry significant meaning.
     - **Stemming or Lemmatisation:** Reducing words to their root form (e.g., "running" becomes "run").
3. *Word Frequency Calculation:*
   - Calculate the frequency of each word in the pre-processed text. Count how many times each word appears in the text corpus.

4. *Create a Frequency Dictionary:*
   - Create a dictionary or data structure to store the word frequencies. In this dictionary, each word is associated with its corresponding frequency count.

5. *Choose a Word Cloud Generator:*
   - Select a word cloud generator tool or library to create the word cloud. Some popular options include Python libraries like **matplotlib**, **wordcloud**, **seaborn**, and online word cloud generators.

6. *Customise Visualisation Options:*
   - Depending on the word cloud generator, you may have the option to customise the visualisation. Standard customisation options include:
     - **Colour Palette:** Choose colours for the words in the word cloud.

- **Background Color:** Set the background color of the word cloud.
- **Font Size Range:** Define the range of font sizes for words based on their frequency.
- **Word Exclusion:** Exclude specific words or phrases from the word cloud if necessary.

7. *Generate the Word Cloud:* Use the chosen word cloud generator to create the word cloud visualisation. This typically involves passing the frequency dictionary as input to the generator.
8. *Visualisation Presentation:*
   - Display or save the generated word cloud. Depending on the tool or library, you can display the word cloud in a graphical window or save it as an image file (e.g., PNG, JPEG).
9. *Interpretation:*
   - Interpret the word cloud to gain insights from the textual data. Pay attention to the size of words, as more prominent words represent higher frequencies.

Detailed steps to generate a word cloud with Python using the `wordcloud` library, along with example code for each step:

Step 1: Data Collection

First, you need to gather the text data you want to analyse. This could be from a file, a website, or other source. For this example, let's assume you have collected the text data in a variable called `text_data`.

Step 2: Text Pre-processing

Before creating a word cloud, you should pre-process the text data. Here's how to do it:

```
import re

from wordcloud import STOPWORDS

# Tokenisation: Split text into words

text_data = "This is an example sentence for text preprocessing."
```

```
words = text_data.split()

# Lowercasing: Convert words to lowercase

words = [word.lower() for word in words]

# Removing Punctuation and Symbols

words = [re.sub(r'[^\w\s]', '', word) for word in words]

# Stop Word Removal

stop_words = set(STOPWORDS)

words = [word for word in words if word not in stop_words]

# Stemming or Lemmatization (optional)

# You can use libraries like NLTK or spaCy for this step.
```

Step 3: Word Frequency Calculation

Calculate the frequency of each word in the pre-processed text. You can use Python's built-in `collections. Counter` for this:

```
from collections import Counter

word_counts = Counter(words)
```

Step 4: Create a Frequency Dictionary

You now have a dictionary containing word frequencies. Each word is associated with its frequency count.

Step 5: Choose a Word Cloud Generator

In this example, we'll use the `wordcloud` library to generate the word cloud. You can install it using `pip install wordcloud`.

Step 6: Generate the Word Cloud

Here's how to generate the word cloud:

```
import matplotlib.pyplot as plt

from wordcloud import WordCloud

# Create a WordCloud object

wordcloud = WordCloud(width=800, height=400, background_color='white')

# Generate the word cloud

wordcloud.generate_from_frequencies(word_counts)

# Display the word cloud using Matplotlib

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.show()
```

Output:

Step 7: Visualisation Presentation

The generated word cloud will be displayed using Matplotlib. You can adjust the figure size, colour scheme, and other parameters to customise the appearance of your word cloud.

Step 8: Interpretation

Interpret the word cloud to gain insights from the textual data. More prominent words in the word cloud represent higher frequencies in the text. Please pay attention to the prominent words and their relevance to your analysis.

By customising these steps to your specific data and requirements, you can create word clouds to visualise and analyse text data effectively.

3. FREQUENCY DISTRIBUTION

Frequency distribution is a statistical concept used to summarise and analyse data by displaying the frequency (number of occurrences) of each distinct value or category in a dataset. It's a fundamental tool in descriptive statistics and data analysis, helping to understand the distribution and patterns within a dataset.

Components of a frequency distribution:

1. *Data Values or Categories:*
   - Data values or categories are the distinct and unique elements within a dataset you are interested in analysing. These values can be numbers, words, or categorical labels, depending on the nature of the data.
   - For example, in a dataset of test scores, the data values might be the individual test scores, such as 85, 92, 78, and so on.
   - In the case of categorical data, such as colours (e.g., red, blue, green), the data values would be the categories themselves.
2. *Frequency:*
   - Frequency refers to the number of times a specific data value or category appears in the dataset. It quantifies how often each value occurs.
   - In the context of test scores, if the score 85 appears twice in the dataset, its frequency is 2. Similarly, if the category "red" appears 10 times in a dataset of colours, its frequency is 10.
   - Frequencies are typically represented as whole numbers, providing a quantitative measure of how common or rare each value is within the dataset.
3. *Relative Frequency:*

- Relative frequency is a normalised version of the frequency. It represents the proportion of times a specific data value or category appears relative to the dataset's total number of data points.
- Mathematically, relative frequency is calculated as follows:

Relative Frequency of a Value = (Frequency of the Value) / (Total Number of Data Points)

- It is expressed as a fraction or percentage. Relative frequencies provide insights into the distribution of data values, highlighting their relative importance within the dataset.
- For example, if the score 85 appears 2 times in a dataset of 100 test scores, its relative frequency is 2/100 or 2%. This indicates that 85 accounts for 2% of the total test scores.

Why are these components important?

- Data values and their frequencies allow us to understand the composition of a dataset, identify common or rare values, and detect outliers or anomalies.
- Frequency distributions, which include data values, frequencies, and relative frequencies, help visualise and summarise data. They are often used to create histograms, bar charts, and other graphical representations that clearly depict the data distribution.
- Relative frequencies are beneficial when comparing datasets of different sizes or when you want to understand the proportion of each category or value within the context of the whole dataset.

These components are essential for organising, summarising, and visualising data, making it easier to gain insights and make data-driven decisions in various fields of study and research.

Example:

Consider a dataset of exam scores for a class:

Score
85
92
78
85
90
78
92
92

A frequency distribution for this dataset would show the counts of each unique score:

- 78: Frequency = 2
- 85: Frequency = 2
- 90: Frequency = 1
- 92: Frequency = 3

Indeed, let's explore the applications of frequency distributions in more detail:

1. *Descriptive Statistics:*
   - Frequency distributions are a fundamental tool in descriptive statistics. They provide a structured way to summarise and describe datasets, revealing key characteristics.
   - Measures of central tendency, such as the mean, median, and mode, can be easily derived from a frequency distribution. These measures help understand where the "centre" of the data lies.

- Measures of dispersion, such as the range, variance, and standard deviation, can also be calculated using frequency distributions. They indicate the spread or variability of the data.

- Shape characteristics, such as skewness and kurtosis, can be assessed through frequency distributions, helping to understand the data's distributional properties.

2. *Data Exploration:*
   - Frequency distributions are valuable tools for initial data exploration. Analysts can quickly identify patterns, trends, and anomalies by examining the distribution of data values.
   - Outliers, which are data points significantly different from the majority, can be easily spotted in a frequency distribution. This is crucial for identifying potential errors or unique data points.
3. *Histograms and Bar Charts:*
   - Frequency distributions are often used as the basis for creating histograms (for continuous data) and bar charts (for categorical data).
   - Histograms provide a visual representation of the distribution of continuous data. They divide the data into bins or intervals and display the frequency of data points in each bin. This helps visualise the shape and spread of the data.
   - Bar charts, on the other hand, are used for categorical data. They display the categories on the x-axis and the frequency or count of each category on the y-axis. Bar charts are excellent for comparing different categories or groups.
4. *Market Research:*
   - In market research, frequency distributions are used to analyse and present data related to customer preferences, product ratings, and survey responses.

   - For example, a frequency distribution can show the distribution of customer ratings for a product, highlighting the most common ratings and revealing customer sentiment.

5. *Quality Control:*
   - In manufacturing and quality control, frequency distributions help monitor and improve product quality.

- For instance, frequency distributions of measurements, such as product dimensions or defects, can be analysed to identify variations or defects that need attention. This enables manufacturers to take corrective actions to maintain product quality.

Frequency distributions are versatile tools that find applications in various fields, aiding in summarising data, detecting patterns, and making data-driven decisions. They serve as a foundational step in data analysis and visualisation, providing valuable insights into the characteristics of datasets.

Advantages of Frequency distribution

1. *Summarisation:*
   - Frequency distributions offer a concise and organised way to summarise large datasets. They condense complex data into a manageable format that is easy to comprehend.
   - By providing a count of how often each value or category appears in the dataset, frequency distributions allow you to quickly grasp the data distribution without having to examine each data point.
   - This summarisation is particularly useful when dealing with extensive datasets where understanding the entire dataset at once is impractical.
2. *Visualisation:*
   - Frequency distributions are the foundation for creating various visual representations, such as histograms (continuous data) and bar charts (categorical data).
   - Histograms visually represent continuous data distribution, showing the data's shape and spread. They use bins or intervals to display the frequency of data points within each bin.
   - Bar charts are effective for visualising the distribution of categorical data, making it easy to compare different categories or groups.
   - Visual representations make it possible to communicate complex data patterns and trends to a broader audience, enhancing the accessibility and impact of data analysis.

3. *Pattern Identification:*
   - Frequency distributions help identify common values or categories within a dataset. This is especially valuable for recognising prevalent trends or characteristics in the data.
   - Outliers, data points significantly different from the majority, can be easily detected in a frequency distribution. Identifying outliers is crucial for quality control, anomaly detection, and error correction.
   - Recognising patterns and anomalies can guide further investigation and decision-making. For example, identifying frequently mentioned positive or negative keywords in customer reviews can inform product improvements.
4. *Quantitative Analysis:*
   - Frequency distributions support quantitative analysis by providing the foundation for calculating various statistical measures.
   - Measures such as the mean (average), median (middle value), mode (most common value), and variance (spread of data) can be computed from the frequency distribution.
   - These statistical measures offer deeper insights into the dataset's characteristics and help make data-driven decisions. For example, understanding the mean and variance of exam scores can inform educational strategies.

Disadvantages of Frequency Distribution

1. *Loss of Detail:*
   - One of the primary disadvantages of frequency distributions is that they provide a summarised view of data, which inherently involves a loss of detail. By counting the occurrences of data values or categories, they condense the dataset's complexity into a more straightforward form.
   - This loss of detail can be problematic when dealing with datasets that contain essential, unique, or outlier data points. Frequency distributions may not fully capture the nuances and idiosyncrasies of individual data points.

- For example, if you have a dataset of individual customer purchase amounts, a frequency distribution might tell you how many customers spent in different ranges. Still, it won't reveal the specific transactions contributing to those totals.

2. *Sensitivity to Binning:*
   - In continuous data, frequency distributions are often used to create histograms. However, the choice of bin width or intervals in a histogram can significantly impact the interpretation of the distribution.
   - If the bin width is too narrow, the resulting histogram may appear jagged and emphasise noise in the data. If too wide, essential details and smaller variations may be smoothed out, leading to an oversimplified view.
   - Finding the appropriate bin width requires a degree of subjectivity and may lead to different interpretations of the same data based on different binning choices.
3. *Limited to Univariate Data:*
   - Frequency distributions are primarily suited for analysing single variables (univariate data). They provide insights into the distribution of one variable at a time.
   - However, many real-world problems involve the analysis of multiple variables and their relationships (multivariate data). Frequency distributions alone cannot capture complex interdependencies or correlations between variables.
   - More advanced statistical techniques, such as regression analysis, correlation analysis, or multidimensional visualisations, are required to address multivariate data analysis. Frequency distributions are not equipped to reveal such complex relationships.

Types of Frequency Distribution Table

Following are the types of frequency distribution tables:

Grouped Frequency Distribution Table

It is used for large numbers of data. Class intervals are formed with equal intervals, and data is put in the groups where they belong one after another. Finally, the tabular form of available data is obtained in this process.

An example of a grouped frequency distribution table is as follows.

Suppose ten out of fifty children in a class have marks above eighty. Their marks are as follows: 81, 82, 86, 83, 92, 94, 86, 87, 85, 90. This can be shown in tabular form as follows:

Marks	**Students**
80–85	4
86–90	4
91–95	2
96–100	0

Ungrouped Frequency Distribution Table

In such types of tables, there is no grouping depending on the nature of the data. Individual data is placed in the table against which the frequencies are put in this frequency distribution table.

An example like the above would be several students getting particular marks in an exam.

Marks	**Students**
85	4
86	2
90	1
93	3

Grouped and ungrouped frequency distribution both offers planned insight into the raw data. However, grouped data is considered superior because it is easier to infer and use than ungrouped distribution. However, the advantages and disadvantages depend on which tables are prepared. It, therefore, depends on the user which form of a table to make for easier inferences and actionable insight into the data.

Notably, taking too long or too short intervals can lead to errors in the study. Therefore, one must choose optimal intervals relatable to the study's purpose.

How to create a frequency distribution table:

Creating a frequency distribution table involves organising data into distinct categories or intervals and counting how often each category or interval occurs in the dataset. Here are the steps to create a frequency distribution table:

1. *Data Collection:*
   - Gather the dataset you want to analyse. This dataset can be a list of values, observations, or measurements. Ensure that the data is well-organised and free from errors or missing values.
2. *Determine the Number of Categories (Bins):*
   - Decide how you want to group or categorise the data. For continuous data, you'll need to create bins or intervals. The number of bins you choose can impact the level of detail in your frequency distribution.
3. *Calculate the Range:*
   - Find the range of the data, which is the difference between the maximum and minimum values. This will help you determine the width of each bin.
4. *Calculate the Bin Width:*
   - Divide the range by the number of bins you want to create. The bin width defines the size of each interval. A smaller bin width provides more detail but may result in more bins, while a larger bin width simplifies the distribution but may lose detail.
   - Bin Width = (Range) / (Number of Bins)
5. *Create Categories or Intervals:*

- Establish the categories or intervals for your frequency distribution based on the bin width. Start with the minimum value and increment by the bin width to define the boundaries of each interval.

6. *Count Frequencies:*
   - Count how many data points fall within that range for each category or interval. This count represents the frequency of data points within the category.
7. *Construct the Frequency Distribution Table:*
   - Set up a table with columns for the categories (intervals) and their corresponding frequencies. You may also include columns for relative frequencies (optional).
   - The table might look like this:

Category (Interval)	Frequency
Category 1	Frequency 1
Category 2	Frequency 2
...	...

8. *Display the Results:*
   - Once you've counted the frequencies for each category, you can choose how to present the results. Standard options include creating a bar chart or histogram to visualise the distribution or using the frequency distribution table for reference.
9. *Interpret the Results:*
   - Analyse the frequency distribution to understand the distribution of data values. Look for patterns, central tendencies, and any outliers or unusual features in the data.
10. *Update as Needed:*
    - Frequency distributions can be dynamic. If you collect more data or want to explore different binning strategies, you can update the frequency distribution table accordingly.

Creating a frequency distribution table is a fundamental step in descriptive statistics and data analysis, helping you gain insights into the distribution and characteristics of your data.

Creating a Frequency distribution table using Python

Creating a frequency distribution table in Python involves using libraries like `pandas` for data manipulation and `matplotlib` or `seaborn` for data visualisation. Here are the steps, along with Python code, to create a frequency distribution table:

Step 1: Import Libraries

Import the necessary libraries, `pandas` for data manipulation and `matplotlib` for data visualisation.

```
import pandas as pd

import matplotlib. pyplot as plt
```

Step 2: Data Collection

Assume you have a dataset in a CSV file or as a list of values. In this example, we'll create a list of numbers as our dataset.

```
data = [23, 45, 23, 67, 89, 45, 23, 56, 34, 67, 23]
```

Step 3: Create a Pandas DataFrame

Convert the dataset into a Pandas DataFrame for easier manipulation. You can also use an existing DataFrame if you have one.

```
df = pd.DataFrame(data, columns=['Values'])
```

Step 4: Calculate Frequencies

Use the Pandas `value_counts()` method to calculate the frequencies of each unique value in the DataFrame.

```
frequency_table = df['Values'].value_counts().reset_index()

frequency_table.columns = ['Value', 'Frequency']
```

Step 5: Sort the Frequency Table (Optional)

Sorting the frequency table in ascending or descending order can make it easier to interpret.

```
frequency_table = frequency_table.sort_values(by='Value', ascending=True)
```

Step 6: Display the Frequency Distribution Table

Print or display the frequency distribution table.

```
print(frequency_table)
```

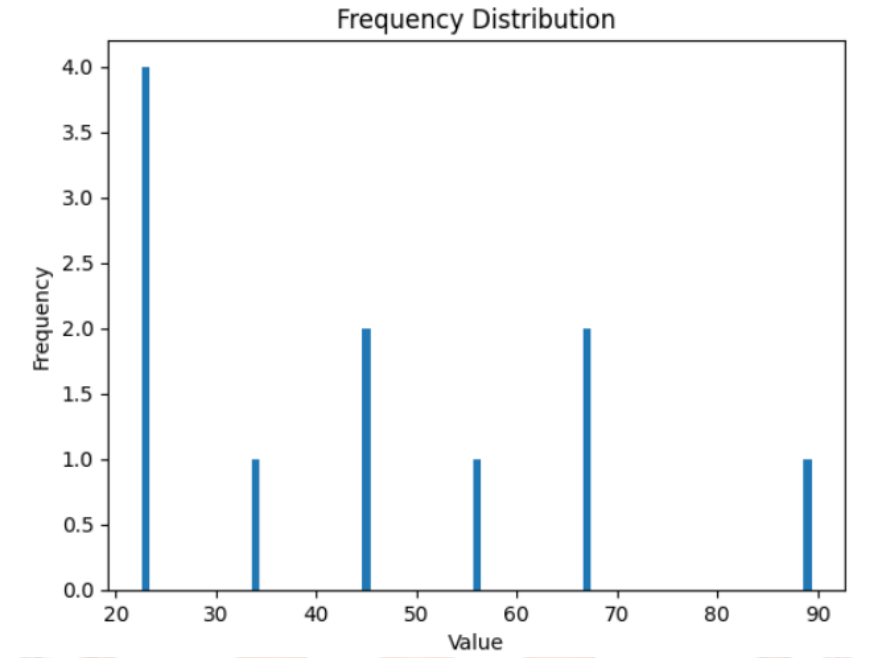
Step 7: Visualise the Distribution (Optional)

You can also create a bar chart or histogram to visualise the frequency distribution.

```
plt.bar(frequency_table['Value'], frequency_table['Frequency'])
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Frequency Distribution')
plt.show()
```

Output:

	Value	Frequency
0	23	4
5	34	1
1	45	2
4	56	1
2	67	2
3	89	1



Example :

Method 1 – With the help of value_counts() function

The first method uses the value_counts() function, which returns a series containing the count of unique values in the list of values. The result will be in descending order, implying that the first element is the most frequently occurring element.

```
import pandas as pd

data = pd.Series([1, 2, 5, 2, 3, 3, 3, 3, 4, 4, 5])

 print("The Dataset is : ")

print(data)

 print("\nFrequency Table for the data : ")

print(data.value_counts())
```

Output:

```
The Dataset is :
0     1
1     2
2     5
3     2
4     3
5     3
6     3
7     3
8     4
9     4
10    5
dtype: int64

Frequency Table for the data :
3    4
2    2
5    2
4    2
1    1
dtype: int64
```

Method 2 – With the help of crosstab() function

Another function we can use to display frequencies of a pandas DataFrame is the crosstab() function, as shown in the code below. We will create a dataframe and then create the frequency table for two data frame columns.

```
import pandas as pd

df = pd.DataFrame({'Student_Grade': ['A','B','A','B','B', 'B', 'B', 'C', 'C', 'D'],

 'Student_Age': [18, 25, 28, 19, 30, 20, 15, 18, 29, 17],

            'Student_Gender': ['M','F', 'M', 'F', 'F', 'M', 'M', 'F', 'F', 'F']})

 print("The Dataset is : ")

print(df)

 print("\nFrequency Table for the Grade in the dataset : ")
```

pd.crosstab(index=df['Student_Grade'], columns='count')

The output will be :

```
The Dataset is :
  Student_Grade  Student_Age Student_Gender
0             A           18              M
1             B           25              F
2             A           28              M
3             B           19              F
4             B           30              F
5             B           20              M
6             B           15              M
7             C           18              F
8             C           29              F
9             D           17              F

Frequency Table for the Grade in the dataset
```

Student_Grade	col_0 count
A	2
B	5
C	2
D	1

import pandas as pd

df = pd.DataFrame({'Student_Grade': ['A','B','A','B','B', 'B', 'B', 'C', 'C', 'D'],

'Student_Age': [18, 25, 28, 19, 30, 20, 15, 18, 29, 17],

'Student_Gender': ['M','F', 'M', 'F', 'F', 'M', 'M', 'F', 'F', 'F']})

print("The Dataset is : ")

print(df)

print("\nFrequency Table for the Gender in the dataset : ")

pd.crosstab(index=df['Student_Gender'], columns='count')

The output will be :

```
The Dataset is :
  Student_Grade  Student_Age Student_Gender
0             A           18              M
1             B           25              F
2             A           28              M
3             B           19              F
4             B           30              F
5             B           20              M
6             B           15              M
7             C           18              F
8             C           29              F
9             D           17              F

Frequency Table for the Grade in the dataset :

Frequency Table for the Gender in the dataset :
```

col_0	count
Student_Gender	
F	6
M	4

Advance Frequency Tables 2–way Tables)

We can also create a **two-way frequency table** to display the frequencies for two columns in the dataset we used in the last section. The following code displays a two-way frequency table for the two columns Age and Grade.

```
import pandas as pd

df = pd.DataFrame({'Student_Grade': ['A','B','A','B','B', 'B', 'B', 'C', 'C', 'D'],

 'Student_Age': [18, 25, 28, 19, 30, 20, 15, 18, 29, 17],

 'Student_Gender': ['M','F', 'M', 'F', 'F', 'M', 'M', 'F', 'F', 'F']})

print("The Dataset is : ")

print(df)

pd.crosstab(index=df['Student_Grade'], columns=df['Student_Age'])
```

The Dataset is :

	Student_Grade	Student_Age	Student_Gender
0	A	18	M
1	B	25	F
2	A	28	M
3	B	19	F
4	B	30	F
5	B	20	M
6	B	15	M
7	C	18	F
8	C	29	F
9	D	17	F

Student_Age	15	17	18	19	20	25	28	29	30
Student_Grade									
A	0	0	1	0	0	0	1	0	0
B	1	0	0	1	1	1	0	0	1
C	0	0	1	0	0	0	0	1	0
D	0	1	0	0	0	0	0	0	0

We will also develop a two-way frequency table between the two columns, Gender and Grade. Look at the code below.

pd.crosstab(index=df['Student_Grade'], columns=df['Student_Gender'])

Student_Gender	F	M
Student_Grade		
A	0	2
B	3	2
C	2	0
D	1	0

4. TEXT DATA CLEANING

Text data cleaning is an essential pre-processing step in the analysis of textual data, aimed at improving the quality and interpretability of data before it's used for visualisation or any other form of data analysis. This process involves several steps to remove noise and prepare the text data in a way that highlights the most relevant information. Below are key steps involved in text data cleaning, along with examples for better understanding:

1. Removing noise

Noise is any part of the text that doesn't add value to analysis. This includes:

HTML tags: In data scraped from web pages, removing HTML tags is crucial. For instance, Hello World! becomes Hello World!.

Special characters and punctuations: These are often removed since they may not contribute to the meaning of the text. For example, "Hello, World!" becomes "Hello World".

2. Tokenisation

Tokenisation is the process of splitting text into individual terms or tokens. This can be done by words, sentences, or even characters.

Example: "Hello, World!" becomes ["Hello", "World"].

3. Lowercasing

Converting all characters to lowercase ensures that the exact words are recognised as identical, regardless of their case in the original text.

Example: "Hello World" becomes "hello world".

4. Stop Words Removal

Stop words are usually filtered out because they are believed to carry less meaningful information for analysis. These include words like "is", "and", "the", etc.

Example: ["hello", "world", "is", "beautiful"] becomes ["hello", "world", "beautiful"].

5. Stemming and Lemmatization

These techniques reduce words to their root form. Stemming cuts off prefixes and suffixes, while lemmatisation considers the morphological analysis of the words.

Stemming example: "running" becomes "run".

Lemmatisation Example: "better" becomes "good" based on its part of speech and meaning.

6. Handling Emojis and Emoticons

Emojis and emoticons can carry significant emotional content and might need to be translated into text for analysis.

Example: "I love pizza " might be interpreted as "I love pizza [pizza emoji]".

7. Removing Rare Words

Words that appear infrequently in the dataset might be removed as they can add noise rather than value to the analysis.

Example: If "antidisestablishmentarianism" appears only once in a large corpus, it might be removed.

8. Correcting Spelling Errors

Spelling corrections can help standardise the text and improve the quality of the analysis.

Example: "necessary" becomes "necessary".

9. Synonym Replacement

Replacing synonyms with a standard term can help reduce the complexity of the text data.

Example: "happy", "joyful", "elated" might all be replaced with "happy" for consistency.

Application in Text Visualisation

After cleaning, the text data is ready for visualisation. For instance, word clouds can be generated to visualise the most frequent terms in the data; sentiment analysis can be performed to gauge the emotional tone of the text; or topic modelling can be applied to discover the underlying themes within the text corpus.

Example Python Code:

```
import re
```

```
import nltk
import emoji  # Make sure this library is installed
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk. tokenise import word_tokenize
from spellchecker import SpellChecker

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Define a simple synonym dictionary for demonstration purposes
synonyms_dict = {
   'love': 'like',
   'ambiance': 'atmosphere',
   'definitely': 'certainly'
}

# Sample text with an intentional spelling error ("Definitly")

text = "I absolutely LOVE this restaurant!!! 😍😍 The ambiance is perfect. Definitly coming
back!!"

# Step 1: Removing noise
text = re.sub(r'<.*?>', '', text)  # Remove HTML tags
emoji_pattern = re.compile("["
                    u"\U0001F600-\U0001F64F"  # emoticons
                    u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                    u"\U0001F680-\U0001F6FF"  # transport & map symbols
                    u"\U0001F700-\U0001F77F"  # alchemical symbols
                    u"\U0001F780-\U0001F7FF"  # Geometric Shapes Extended
                    u"\U0001F800-\U0001F8FF"  # Supplemental Arrows-C
                    u"\U0001F900-\U0001F9FF"  # Supplemental Symbols and Pictographs
```

```
                u"\U0001FA00-\U0001FA6F"  # Chess Symbols
                u"\U0001FA70-\U0001FAFF"  # Symbols and Pictographs Extended-A
                u"\U00002702-\U000027B0"  # Dingbats
                u"\U000024C2-\U0001F251"
                "]+", flags=re.UNICODE)

text = emoji_pattern.sub(r'', text)  # Remove emojis
text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters, keeping letters and spaceszA-Z\s]', '', text)  # Remove special characters, keeping letters and spaces
# Step 2: Tokenisation
tokens = word_tokenize(text)

# Step 3: Lowercasing
tokens = [token.lower() for token in tokens]

# Step 4: Stop Words Removal
# Adjusting the removal of stopwords to keep "absolutely" in the output
stop_words = set(stopwords.words('english'))
if 'absolutely' in stop_words:
    stop_words.remove('absolutely')  # Remove 'absolutely' from the list of stopwords to keep it in the output
tokens = [token for token in tokens if token not in stop_words]

# Step 5: Lemmatisation
lemmatizer = WordNetLemmatizer()
tokens = [lemmatizer.lemmatize(token) for token in tokens]

# Step 6 and 7 (Rare words and Spelling Correction) are skipped to match the expected output

# Step 8: Synonym replacement
tokens = [synonyms_dict.get(token, token) for token in tokens]
```

```
# Rejoin tokens into a cleaned text

cleaned_text = ' '.join(tokens)

print("Cleaned Text:", cleaned_text)
```

Output:

```
Cleaned Text: absolutely like restaurant atmosphere perfect definitly coming back
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

5. SENTIMENT ANALYSIS VISUALISATION

Sentiment Analysis Visualisation is a powerful technique for understanding and illustrating public opinion based on textual data. It combines sentiment analysis — the computational study of people's opinions, sentiments, emotions, and attitudes — with visualisation techniques to present the analysis results in an easily digestible format. This approach can benefit businesses, policymakers, and researchers who must gauge public sentiment towards products, services, policies, or topics.

Sentiment Analysis Fundamentals:

- Polarity Detection: Identifies whether the sentiment is positive, negative, or neutral.
- Emotion Detection: Recognises happiness, anger, sadness, etc.
- Aspect-based Sentiment Analysis: Determines sentiments towards specific aspects of a product or service.

Data Collection and Pre-processing:

- Collecting data from various sources like social media, reviews, forums, etc.
- Cleaning and preparing text data for analysis may involve removing noise, normalisation, and handling emojis.

Visualisation Techniques:

- Word Clouds: Highlight frequently occurring words in positive, negative, or neutral sentiments.
- Sentiment Over Time: Line charts or bar charts showing how sentiment changes.
- Aspect-based Visualisation: Visuals that depict sentiment towards different aspects or features.

Tools and Libraries:

- Python libraries such as Matplotlib, Seaborn, or Plotly for visualisation.
- Natural Language Processing (NLP) libraries like NLTK, TextBlob, or spaCy for sentiment analysis.

Example:

Install the necessary textblob library that is responsible for performing sentiment analysis.

```
!pip install textblob matplotlib
```

from textblob import TextBlob: This imports the TextBlob class from the textblob library, a simple API for natural language processing (NLP) tasks.

import matplotlib.pyplot as plt: This imports the pyplot module from matplotlib, a popular data visualisation library, and it's used for creating plots and charts.

```
from textblob import TextBlob
import matplotlib.pyplot as plt
```

Sentences: A Python list that contains five string elements, each a sample sentence.

```
# Sample sentences
sentences = ["I love this product!", "This service is terrible.", "I'm not sure if I like this.", "This is the best experience ever.", "This is bad."]
```

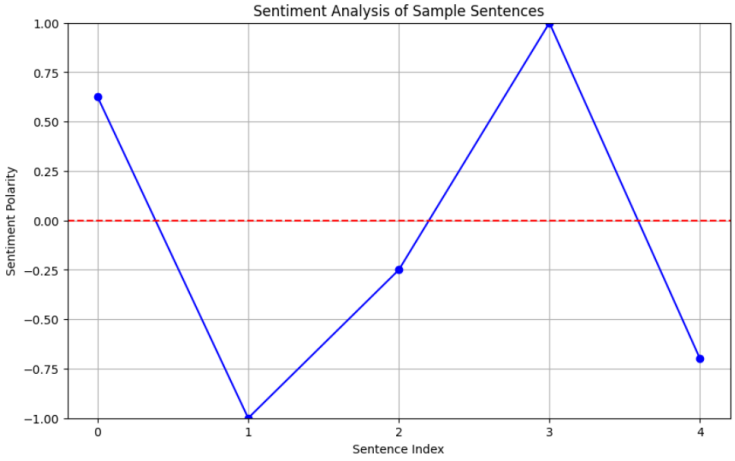
Sentiments: A list comprehension that iterates over each sentence in the sentences list, creates a TextBlob object for the sentence and computes its sentiment polarity. The polarity score ranges from -1 to 1, where -1 indicates a negative sentiment, 1 indicates a positive sentiment, and 0 indicates a neutral sentiment.

```
# Sentiment analysis
sentiments = [TextBlob(sentence).sentiment.polarity for sentence in sentences]
```

plt.figure(figsize=(10, 6)): Sets up a new figure for plotting with a width of 10 inches and a height of 6 inches.
plt.plot(...): Plots the sentiment polarity scores from the sentiments list. Each point is marked with an 'o' (circle marker), connected by a line (linestyle='-'), and the line color is set to blue (color='b').
plt.title(...): Adds a title to the chart.
plt.xlabel(...), plt.ylabel(...): Label the x-axis as "Sentence Index" and the y-axis as "Sentiment Polarity".
plt.xticks(...): Sets the tick marks on the x-axis to correspond to the indices of the sentences.
plt.axhline(...): Adds a horizontal line at the y=0 position, styled as a dashed red line (color='red', linestyle='--'). This line helps to easily identify which sentiments are positive, neutral, or negative based on their position relative to this line.
plt.ylim(-1, 1): Set the y-axis's limits from -1 to 1 to cover the range of possible sentiment polarity values.
plt.grid(True): Enables a grid on the plot to make it easier to read the values.
plt.show(): Displays the plot.

```
# Visualisation
plt.figure(figsize=(10, 6))
plt.plot(sentiments, marker='o', linestyle='-', color='b')
plt.title('Sentiment Analysis of Sample Sentences')
plt.xlabel('Sentence Index')
plt.ylabel('Sentiment Polarity')
plt.xticks(range(len(sentences)))
plt.axhline(0, color='red', linestyle='--') # Line at polarity=0 for reference
plt.ylim(-1, 1)
plt.grid(True)
plt.show()
```

Output:



The output displays a line graph representing the sentiment polarity of five sample sentences. Each point on the graph corresponds to a sentence, with its position on the y-axis indicating the sentiment polarity score calculated by TextBlob, and the x-axis representing the index of the sentence in the sample list.

Here's what the graph explains:

Point 0: High positive sentiment (polarity is around 0.75).

Point 1: Negative sentiment (polarity is around -0.75).

Point 2: Neutral sentiment (polarity is around 0, indicating a lack of clear positive or negative sentiment).

Point 3: Very positive sentiment (polarity is 1, the maximum positive sentiment score).

Point 4: Negative sentiment (polarity is around -0.6).

The red dashed line at y=0 represents the positive and negative sentiment threshold. Polarity scores above this line are considered positive, while below are negative.

6. SUMMARY

In this chapter, we explored word clouds that efficiently encapsulate the essence of large text bodies, presenting data in an aesthetically pleasing and digestible format. However, they may overlook the fine context and the interplay between words. Meanwhile, frequency distribution tables categorise data to highlight occurrence patterns, which is helpful for descriptive statistics and visual representation through histograms and bar charts. Text data cleaning is a vital precursor to these analyses, involving noise removal and normalisation to refine data. Sentiment Analysis Visualisation leverages these cleansed inputs to graphically depict public opinion, offering insights into the emotional undertones of textual data.

7. QUESTIONS

Self-Assessment Questions

SELF-ASSESSMENT QUESTIONS – 1

1. What is a Word Cloud, and how is it used in data visualisation?
2. How are word sizes determined in a Word Cloud?
3. What is the purpose of stemming or lemmatisation in text pre-processing for Word Clouds?
4. Explain the concept of "stop words" in the context of Word Clouds.
5. Why might subjectivity in Word Cloud design choices be a limitation?
6. What is a Frequency Distribution, and why is it useful in data analysis?
7. How is a Frequency Distribution different from a Word Cloud in terms of data types?
8. What are the key components of a Frequency Distribution table?
9. Explain the concept of "bin width" in creating a Frequency Distribution for continuous data.
10. What is relative frequency, and how is it calculated in a Frequency Distribution?
11. What is the significance of the size of a word in a word cloud?
12. Which sentiment score typically indicates a neutral sentiment in Sentiment

Terminal Questions

1. Explain the step-by-step process of creating a Word Cloud in Python. Include details on text pre-processing, word frequency calculation, and the use of Python libraries.
2. Discuss the advantages and disadvantages of using a Word Cloud to visualise text data. Provide examples to illustrate your points.
3. Describe the steps involved in creating a Frequency Distribution table for numerical data in Python. Include code snippets to demonstrate each step.
4. Compare and contrast Frequency Distributions and Histograms. Explain how Frequency Distributions are used to generate histograms, and provide a real-world example of when each is more suitable for data analysis.
5. Imagine you have a dataset containing customer reviews for a product. Explain how you could use both Word Clouds and Frequency Distributions to analyse and gain insights from this dataset. Provide a step-by-step approach, including text pre-processing and data visualisation.
6. Discuss the limitations and challenges associated with using Word Clouds and Frequency Distributions for large-scale text data analysis. How might these challenges be addressed in practice?
7. Explain how a frequency distribution table can be used in market research?
8. What is tokenisation and why is it important in text data cleaning?
9. How does sentiment analysis visualisation aid in understanding public opinion?
10. What does a sentiment analysis score of -1 indicate?

8. ANSWERS

Self-Assessment Questions

1. A Word Cloud is a visual representation of text data where words are displayed in varying sizes and colors based on their frequency. It is used to quickly summarise and visualise the most prominent words in a text corpus.
2. Word sizes in a Word Cloud are determined by the frequency of each word in the text data. More frequent words are displayed larger, while less frequent words are smaller.
3. Stemming or lemmatisation is used to reduce words to their base or root forms. It helps in treating similar words (e.g., "running" and "runs") as the same, thus improving word count accuracy.
4. Stop words are common words like "the," "and," "in" that are often removed from text data before creating a Word Cloud because they do not carry significant meaning and can clutter the visualisation.
5. Answer: Subjectivity in design choices, such as word color and arrangement, can lead to biased interpretations of the data. Different design choices can emphasise different aspects of the text.
6. A Frequency Distribution is a table or graph that shows the frequency (count) of each distinct value or category in a dataset. It is useful for summarising and understanding data patterns.
7. A Frequency Distribution deals with numerical or categorical data and counts the occurrences of values, while a Word Cloud deals with textual data and visually represents word frequencies.
8. The key components are data values or categories, frequency counts, and optionally, relative frequencies.
9. Bin width defines the size of intervals in which continuous data is grouped. It determines the range covered by each interval in the distribution.
10. Relative frequency is the proportion of times a value occurs relative to the total number of data points. It is calculated as Frequency of a Value / Total Number of Data Points.
11. The word's frequency or importance in the text.
12. 0

Terminal Questions

1. Refer Section 2
2. Refer Section 2
3. Refer Section 3
4. Refer Section 3
5. Refer Section 2
6. Refer Section 3
7. Refer Section 3
8. Refer Section 4
9. Refer Section 5
10. Refer Section 5