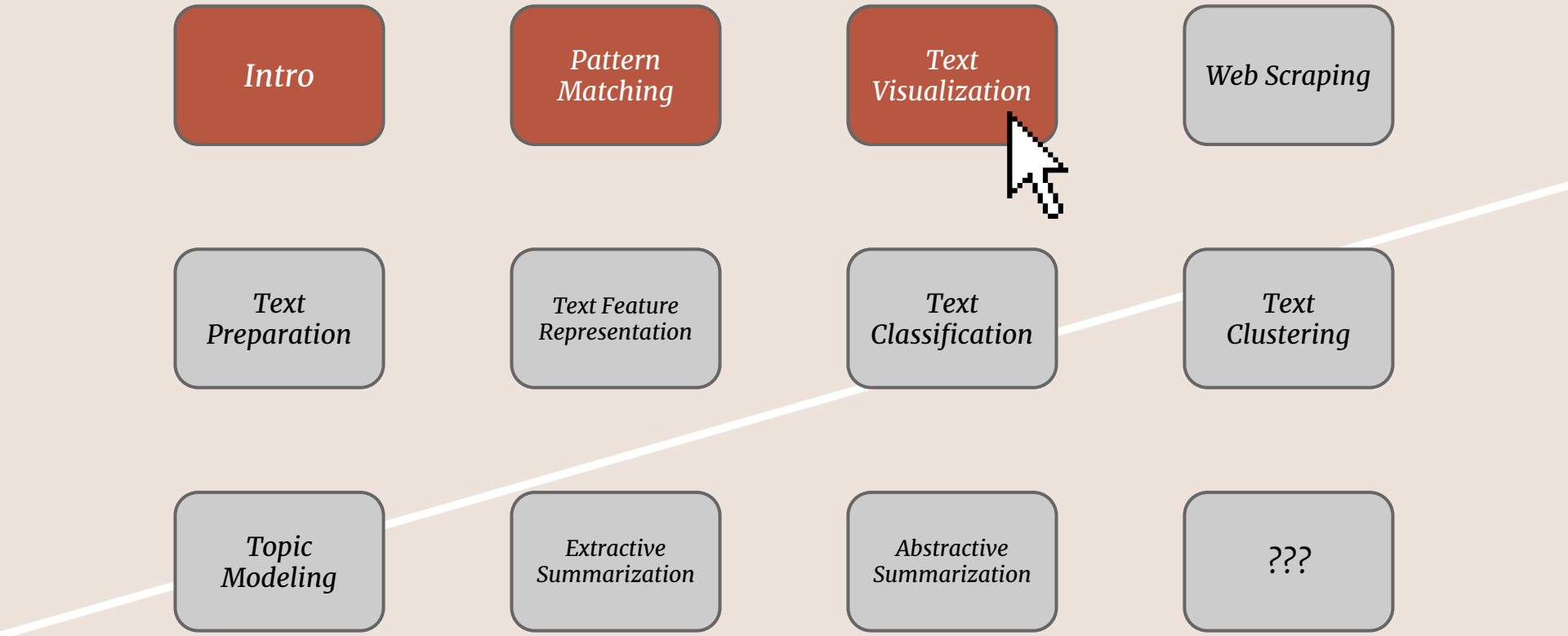
# Textual Data Visualization

## CPE 393: Text Analytics

***Dr. Sansiri Tarnpradab***

*Department of Computer Engineering*
*King Mongkut's University of Technology Thonburi*


Intro
Pattern Matching
Text Visualization
Web Scraping
Text Preparation
Text Feature Representation
Text Classification
Text Clustering
Topic Modeling
Extractive Summarization
Abstractive Summarization
???

# *Outline*

## Text Data Visualization

- Significance & Benefits
- Challenges
- Pre-processing
- Tools
- Examples
  - Word cloud
  - Heatmap
  - Bar chart
  - Bubble chart
  - Network diagram
  - Topic modeling visualization

# *Data Visualization*

**A PICTURE IS WORTH A THOUSAND WORDS**

## What:

Technique to present data in a pictorial/graphical format

## Significance & Benefits:

- Gain insights into an information space by mapping data onto graphical primitives
- Provide qualitative overview of large data sets
- Search for patterns, trends, structure, irregularities, relationships among data
- Help find interesting regions and suitable parameters for quantitative analysis
- Simplification

## Sales

$297K
this month
▲ $16K vs last month

$9.6K
today

$20.6K
yesterday

## Biggest deals this month

| | |
|---|---|
| Alice | $8,600 |
| Jared | $8,500 |
| Heather | $7,540 |
| Shaun | $7,450 |
| Marsha | $6,530 |
| Jared | $4,565 |
| Heather | $4,560 |
| Polly | $4,215 |
| Dalisu | $3,560 |

## Social followers

19.5K
LinkedIn
▲ 11 v yday

10.5K
Twitter
▲ 22 v yday

## Website (past 7 days)

27.2K
Users
▲ 1.6K vs last week

126
Enquiries
▼ 28 vs last week

## Active users

## NPS (past 30 days)


61

## Recent feedback

OK
14 days ago

Very Helpful!!
2 months ago

very good "thumbs up"
2 months ago

Daily pulse dashboard

13:33

# *Data Visualization*

# *for* ***Text***

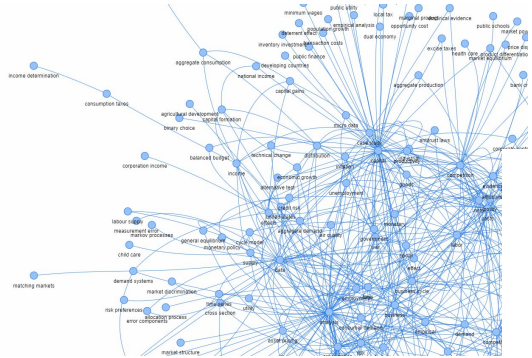
**What:**

- Textual information in a visual form
- Easier to understand and analyze

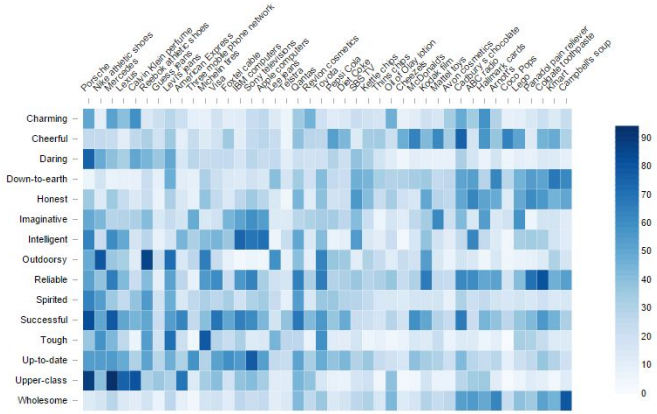
**Examples:**

- Word cloud
- Heatmap
- Bar chart
- Bubble chart
- Network diagram
- Topic modeling visualization

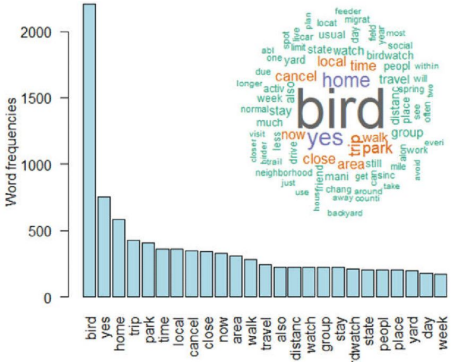

Word Clouds



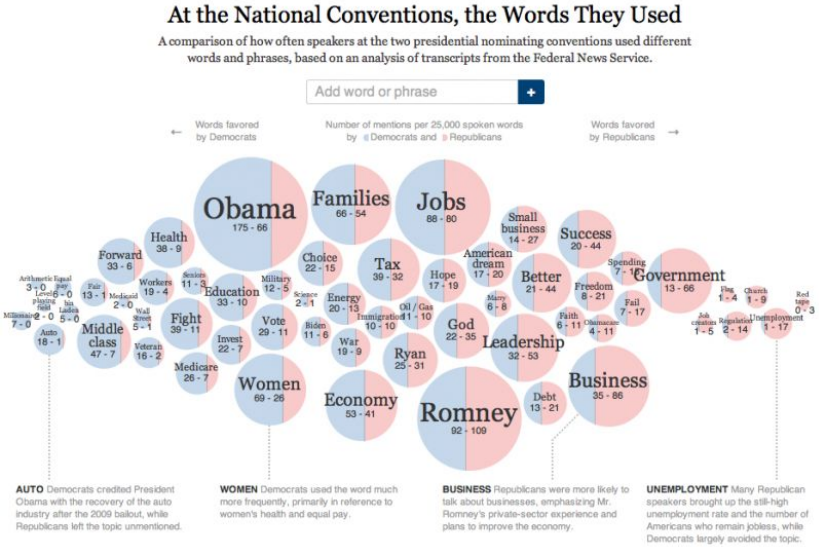
Network Diagram



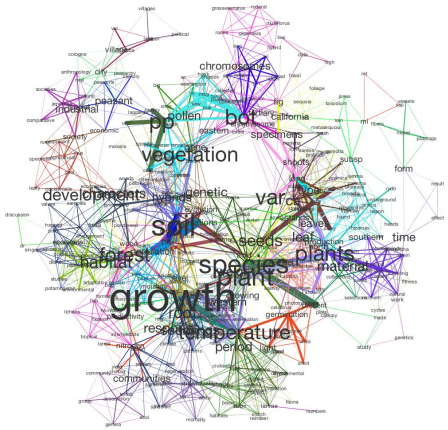

Heatmap




Bar Chart




Bubble Chart




Topic Modeling

Refs:

[Word Clouds]
[Heatmap]
[Bar Chart]
[Bubble Chart]
[Network Diagram]
[Topic Modeling]

- S. Tarnpradab -

# *Challenges*

**Sources:**

- Fixed format
  - E.g. Surveys
  - What else?

- Semi-/Unstructured format
  - E.g. Social media posts
  - What else?

**Nature of Data:**

- High dimensionality
- Ambiguity
- Variability in language

# *Pre-processing*

**Common Techniques:**

- Tokenization
- Stopword removal
- Lemmatization (or stemming)
- Special characters

# *Tools*

## *for Text Data Visualization*

- `wordcloud`
- `matplotlib`
- `seaborn`
- `nltk`
- `networkx`
- `plotly`

And more...

# *Word Clouds*


Word clouds display words from a text document with font size proportional to their frequency.

## Keys:

- Frequency $\propto$ Size
- Display of each word, as-is

## Python Libraries:

- `wordcloud`
- `matplotlib`

# Text

The amber droplet hung from the branch, reaching fullness and ready to drop. It waited. While many of the other droplets were satisfied to form as big as they could and release, this droplet had other plans. It wanted to be part of history. It wanted to be remembered long after all the other droplets had dissolved into history. So it waited for the perfect specimen to fly by to trap and capture that it hoped would eventually be discovered hundreds of years in the future.

*Note: A paragraph generated by https://randomword.com/paragraph*

```python
from wordcloud import WordCloud

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

# Display the generated word cloud using matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

| | Word | Frequency |
|---|---|---|
| 12 | to | 6 |
| 14 | . | 6 |
| 5 | the | 5 |
| 10 | and | 3 |
| 20 | other | 3 |
| 15 | It | 3 |
| 19 | of | 3 |
| 34 | be | 3 |
| 7 | , | 2 |
| 33 | wanted | 2 |


Ref: https://giphy.com/search/incredulous-disbelief

# Word Clouds

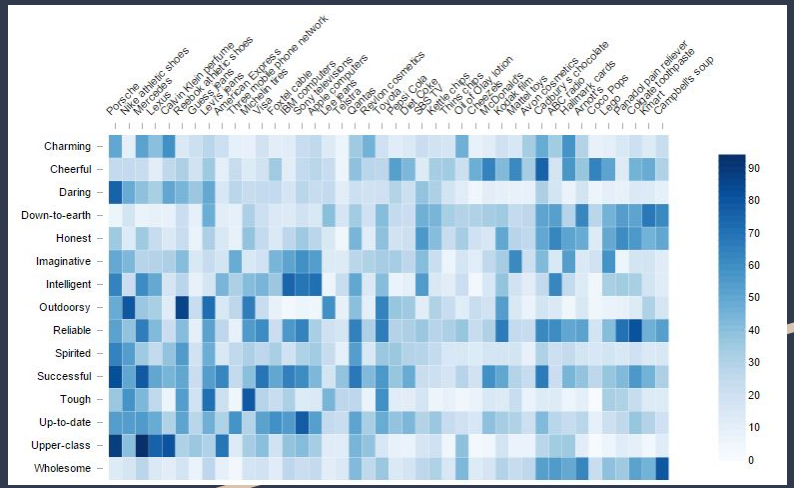
## Steps

1. Provide input text
2. Tokenizing input → words
3. Remove stopwords
4. Determine word frequency
5. Assign size
6. Display

| | Word | Frequency |
|---|---|---|
| 17 | wanted | 2 |
| 1 | droplet | 2 |
| 8 | waited | 2 |
| 19 | history | 2 |
| 10 | droplets | 2 |
| 26 | trap | 1 |
| 21 | long | 1 |
| 22 | dissolved | 1 |
| 23 | perfect | 1 |
| 24 | specimen | 1 |


Now this makes better sense 👍

# *Heatmap*




Heatmap can show the frequency of specific terms in a document or across a collection of documents.

## Keys:

- Frequency ∝ Color intensity
- Each word is represented by a color

## Python Libraries:

- `seaborn`
- `matplotlib`

# *CountVectorizer*

**Steps:**

1. Tokenization
2. Build a vocabulary of unique words
3. Construct a DTM
4. Sparse representation (non-zero entries are stored)
5. Output matrix (Voila!)

**What:**

- From `scikit-learn` library
- Is a feature extraction technique

**Purpose:**

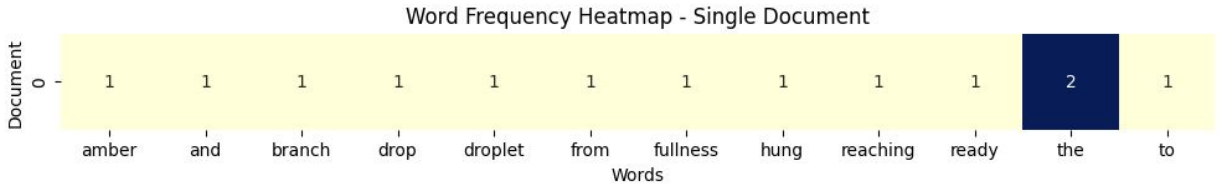
- To convert the single document into a document-term matrix (DTM).
- Matrix of token counts

**Output:**

- Vector (for each document)
- Each element of the vector = count of each word in the document

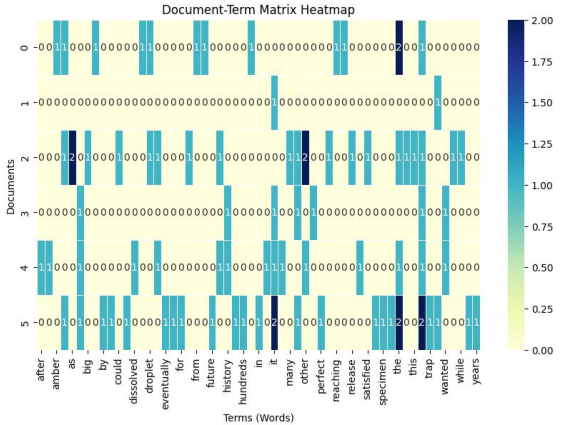
# Heatmap: One Document

```python
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer

# Sample text data
document = "The amber droplet hung from the branch reaching fullness and ready to drop."

# Tokenize the document into words
vectorizer = CountVectorizer()
X = vectorizer.fit_transform([document])
words = vectorizer.get_feature_names_out()

# print(words) # all words
# print(X.toarray()) # frequency of each word

# Create a heatmap
plt.figure(figsize=(12, 1))
sns.heatmap(X.toarray(), cmap="YlGnBu", annot=True, fmt="d", xticklabels=words, cbar=False)
plt.title('Word Frequency Heatmap - Single Document')
plt.xlabel('Words')
plt.ylabel('Document')
plt.show()
```

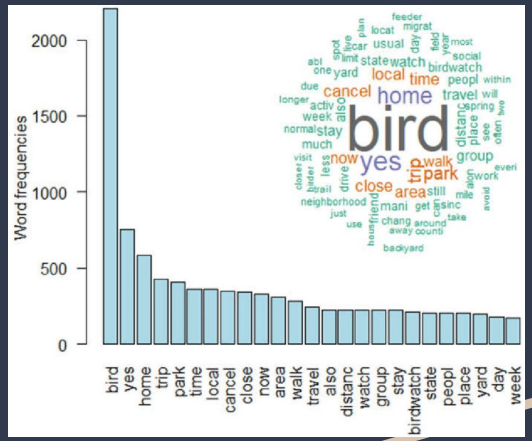


# Heatmap: Multiple Documents

```python
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Sample text data where each sentence is considered a document
text = [
    "The amber droplet hung from the branch, reaching fullness and ready to drop.",
    "It waited.",
    "While many of the other droplets were satisfied to form as big as they could and release, this droplet had other plans.",
    "It wanted to be part of history.",
    "It wanted to be remembered long after all the other droplets had dissolved into history.",
    "So it waited for the perfect specimen to fly by to trap and \
    capture that it hoped would eventually be discovered hundreds of years in the future."
]

# Create a document-term matrix using CountVectorizer
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(text)
dtm_df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())

# Create a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(dtm_df, cmap="YlGnBu", annot=True, fmt="d", linewidths=.5)
plt.title('Document-Term Matrix Heatmap')
plt.xlabel('Terms (Words)')
plt.ylabel('Documents')
plt.show()
```



# Bar Chart




Bar chart can represent the frequency of words or phrases in a text.

## Keys:

- Frequency $\propto$ Bar height
- Each token is represented by a bar

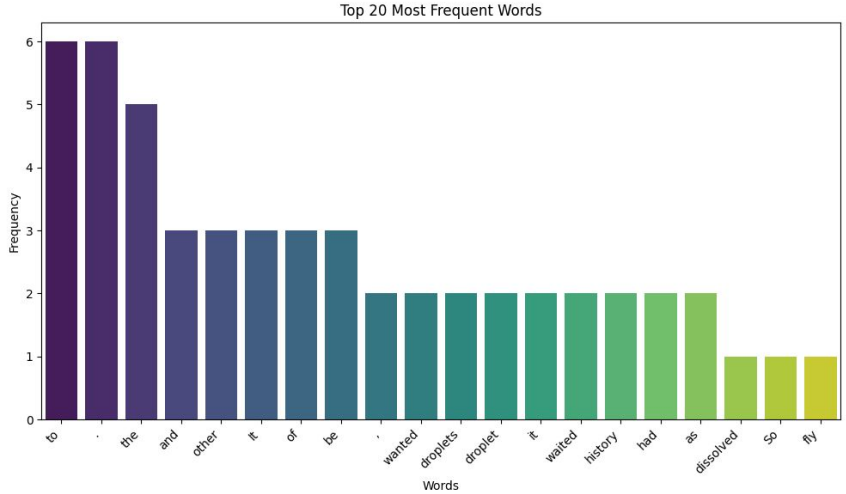
## Python Libraries:

- `matplotlib`
- `nltk`
- `seaborn`

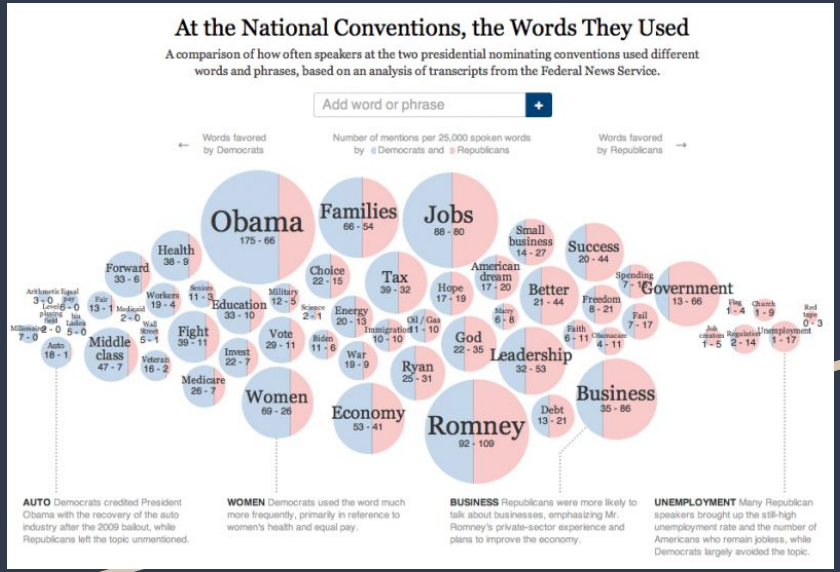
```
import seaborn as sns
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
import nltk

# Download NLTK resources (if not already downloaded)
nltk.download('punkt')

# Sample text data
text = "The amber droplet hung from the branch, reaching fullness and ready to drop.\
    It waited. While many of the other droplets were satisfied to form as big as they could and release, \
    this droplet had other plans. It wanted to be part of history. \
    It wanted to be remembered long after all the other droplets had dissolved into history. \
    So it waited for the perfect specimen to fly by to trap and \
    capture that it hoped would eventually be discovered hundreds of years in the future."
```

Ref: https://giphy.com/search/incredulous-disbelief

```
# Tokenize the text into words
words = word_tokenize(text)

# Calculate word frequencies
word_freq = FreqDist(words)

# Convert word frequencies to a DataFrame for seaborn
data = {'Word': list(word_freq.keys()), 'Frequency': list(word_freq.values())}
df_word_freq = pd.DataFrame(data)

# Sort DataFrame by frequency in descending order
df_word_freq = df_word_freq.sort_values(by='Frequency', ascending=False)

# Plot a bar chart using seaborn
plt.figure(figsize=(12, 6))
sns.barplot(x='Word', y='Frequency', data=df_word_freq.head(20), palette='viridis')
plt.title('Top 20 Most Frequent Words')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.show()
```



# *Bubble Chart*




The frequency of words used at the National Conventions.
The colour within circles reflects the political party.

Bubble chart involves visualizing not only the word frequency but also an additional dimension.

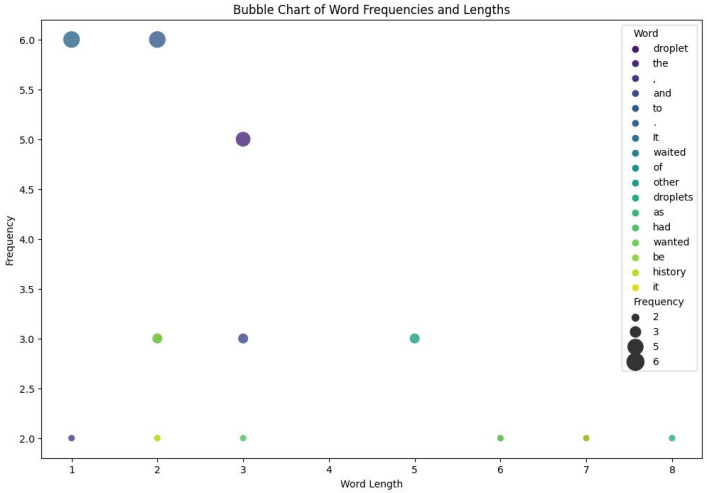
## Keys:

- Frequency $\propto$ Bubble size
- 2-Dimensional representation
- Can represent other context

## Python Libraries:

- `matplotlib`
- `nltk`
- `seaborn`
- `plotly`

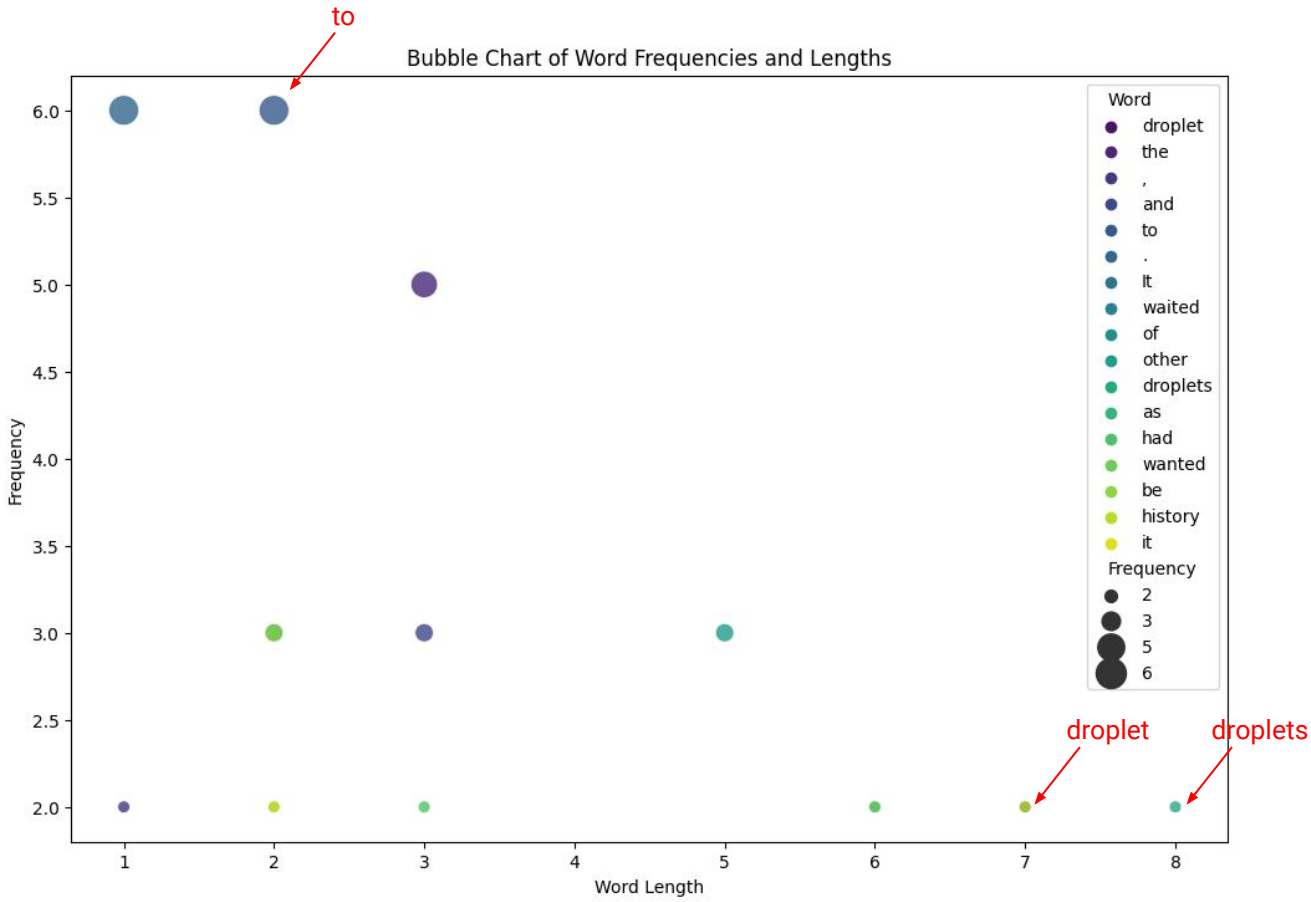
```
import seaborn as sns
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
import nltk

# Download NLTK resources (if not already downloaded)
nltk.download('punkt')

# Sample text data
text = "The amber droplet hung from the branch, reaching fullness and ready to drop.\
    It waited. While many of the other droplets were satisfied to form as big as they could and release, \
    this droplet had other plans. It wanted to be part of history. \
    It wanted to be remembered long after all the other droplets had dissolved into history. \
    So it waited for the perfect specimen to fly by to trap and \
    capture that it hoped would eventually be discovered hundreds of years in the future."
```



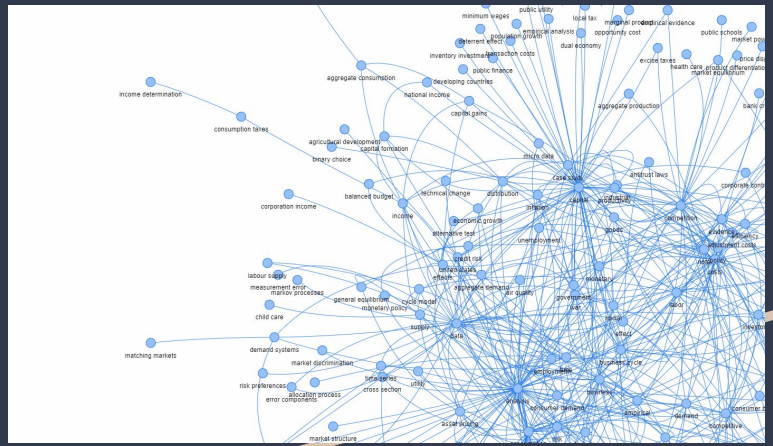

```
# Tokenize the text into words
words = word_tokenize(text)

# Calculate word frequencies
word_freq = FreqDist(words)

# Create a DataFrame with word frequencies and lengths
data = {'Word': list(word_freq.keys()), 'Frequency': list(word_freq.values()), 'Length': [len(word) for word in word_freq.keys()]}
df_word_data = pd.DataFrame(data)

# Filter out words with frequency less than 2 for better visualization
df_word_data = df_word_data[df_word_data['Frequency'] >= 2]

# Plot a bubble chart using seaborn
plt.figure(figsize=(12, 8))
sns.scatterplot(x='Length', y='Frequency', size='Frequency', data=df_word_data, hue='Word', sizes=(50, 300), palette='viridis', alpha=0.8)
plt.title('Bubble Chart of Word Frequencies and Lengths')
plt.xlabel('Word Length')
plt.ylabel('Frequency')
plt.show()
```

*- S. Tarnpradab -*

| | Word | Frequency | Length |
|---|---|---|---|
| 2 | droplet | 2 | 7 |
| 5 | the | 5 | 3 |
| 7 | , | 2 | 1 |
| 10 | and | 3 | 3 |
| 12 | to | 6 | 2 |
| 14 | . | 6 | 1 |
| 15 | It | 3 | 2 |
| 16 | waited | 2 | 6 |
| 19 | of | 3 | 2 |
| 20 | other | 3 | 5 |
| 21 | droplets | 2 | 8 |
| 25 | as | 2 | 2 |
| 31 | had | 2 | 3 |
| 33 | wanted | 2 | 6 |
| 34 | be | 3 | 2 |
| 36 | history | 2 | 7 |
| 44 | it | 2 | 2 |



# Network Diagram




Network diagram can represent relationships between entities in a text. Nodes can be entities, and edges represent relationships.

## Keys:

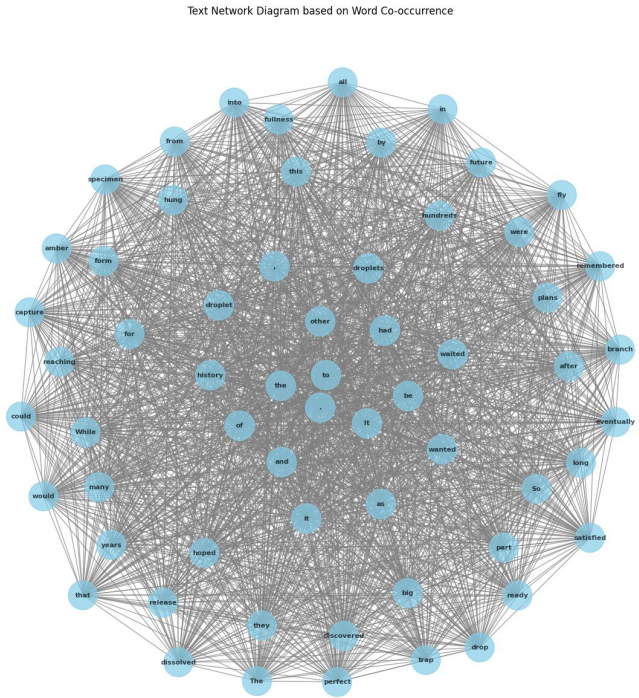
- Graph-based concept
- Depict relationships

## Python Libraries:

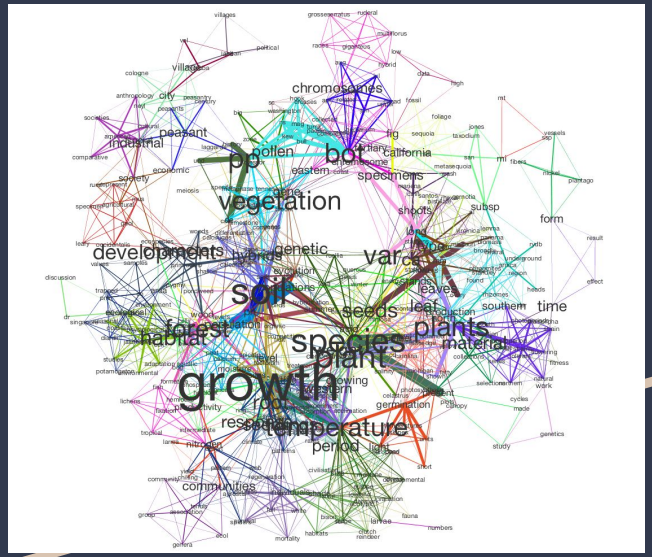
- `matplotlib`
- `nltk`
- `networkx`

```
import seaborn as sns
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
import nltk

# Download NLTK resources (if not already downloaded)
nltk.download('punkt')

# Sample text data
text = "The amber droplet hung from the branch, reaching fullness and ready to drop.\
    It waited. While many of the other droplets were satisfied to form as big as they could and release, \
    this droplet had other plans. It wanted to be part of history. \
    It wanted to be remembered long after all the other droplets had dissolved into history. \
    So it waited for the perfect specimen to fly by to trap and \
    capture that it hoped would eventually be discovered hundreds of years in the future."
```

```
# Tokenize the text into words
words = word_tokenize(text)

# Create a graph using networkx
G = nx.Graph()

# Create edges between co-occurring words
for word1, word2 in combinations(words, 2):
    if G.has_edge(word1, word2):
        G[word1][word2]['weight'] += 1
    else:
        G.add_edge(word1, word2, weight=1)

# Set node size based on degree (number of connections)
node_size = [deg * 20 for deg in dict(G.degree()).values()]

# Draw the network diagram
plt.figure(figsize=(12, 12))
pos = nx.spring_layout(G, seed=42)
nx.draw(G, pos, with_labels=True, font_size=8, node_size=node_size, font_color='black',
        edge_color='gray', font_weight='bold', alpha=0.7, node_color='skyblue')
plt.title('Text Network Diagram based on Word Co-occurrence')
plt.show()
```



# *Topic Modeling*




An NLP technique to automatically identify topics present in a document without prior knowledge of the topics.

## Keys:

- Topics in a document
- Thematic structure
- Automatic process

## Python Libraries:

- `nltk`
- `gensim`
- `pyLDAvis`

# EDA

A process that involves visually and statistically summarizing, interpreting, and understanding the main characteristics of a dataset.

## Purposes:

For example...

- Better understand textual content
- Visualize word frequencies
- Visualize textual relationships
- Identify keywords
- Identify themes (or topics)
- Detect anomalies/outliers
- Explore document similarity

# *Conclusion*

## Text Data Visualization

- Significance & Benefits
- Challenges
- Pre-processing
- Tools
- Word cloud
- Heatmap
- Bar chart
- Bubble chart
- Network diagram
- Topic modeling visualization

# Q & A