Massive Text Data Visualization

Giovana Voltoline

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Sebastião Augusto Rodrigues Figueiredo Pais

Outubro de 2024

Declaração de Integridade

Eu, Giovana Voltoline, que abaixo assino, estudante com o número de inscrição M11657 de/o Engenharia Informática da Faculdade Departamento Informático, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o Código de Integridades da Universidade da Beira Interior.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 10/10/2024

Dedication

To the three remarkable women who instilled in me the qualities needed to achieve this: my sister, my aunt, and mostly my mom. Thank you for your unwavering support and inspiration.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to José Augusto Martins Nieviadonski, who guided me from the beginning of my journey toward this master's degree. The knowledge you shared with me was crucial in helping me reach this point, and I am truly grateful for the time and effort you dedicated to my success.

I would also like to extend my appreciation to my advisor, Sebastião Pais, for providing the thesis topic and offering guidance during the stages of this research. Your support was instrumental in shaping the direction of this work.

My heartfelt thanks go to Luiz Rossa, for your invaluable insights, practical advice, and for always being available to discuss ideas and challenges. Your expertise and suggestions greatly enriched the quality of this work, and I am certain I would not have been able to complete this journey without your help.

This thesis was made possible through the generous support and funding of NOVA LINCS. I am deeply grateful for their commitment to fostering research and innovation, without which this work would not have been possible. Their contribution has provided the necessary resources and environment for me to carry out this research, and for that, I extend my most sincere thanks.

Lastly, I would like to express my sincere appreciation to Vera Lucia Sabidussi for your continuous encouragement and emotional support. Your positivity and belief in my abilities kept me motivated during the most difficult moments. I am deeply grateful for the balance you brought into my life, reminding me to stay focused and determined.

Without the support and contributions of these individuals, this thesis would not have been possible. Thank you.

Preface

This thesis builds upon the work presented in the paper ”Understanding Massive Text Visualization [1],” co-authored by Giovana Voltoline, Sebastião Pais, Bruno Silva, and João Cordeiro. The paper was published in ASONAM ’23: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2023), and it provided the foundation for the research presented here

Resumo Alargado

Esta tese centra-se no desenvolvimento de uma solução especializada para a visualização de dados textuais. Com o aumento do volume de texto não estruturado em vários campos de investigação, há uma necessidade crescente de ferramentas eficazes que ajudem os utilizadores a analisar e visualizar esses dados. O objetivo principal desta investigação é duplo: em primeiro lugar, investigar as necessidades e os desafios enfrentados pelos investigadores que trabalham com dados de texto e, em segundo lugar, conceber e implementar uma solução prática que satisfaça essas necessidades.

A fase inicial envolve uma extensa revisão da literatura, com o objetivo de compreender o panorama atual da visualização de dados de texto, categorizar os objectivos dos investigadores e identificar os processos que seguem. Esta pesquisa informa a seleção dos tipos de visualização e as bibliotecas Python mais adequadas para os implementar.

O produto final é disponibilizado como um pacote Python e um repositório de código aberto no GitHub, fornecendo uma ferramenta flexível que aborda os principais desafios na visualização de dados de texto. Esta tese não só contribui com uma ferramenta funcional para os investigadores, como também destaca a importância de soluções direcionadas para o tratamento e visualização de dados textuais não estruturados.

Abstract

This thesis focuses on the development of a specialized solution for the visualization of textual data. With the increasing volume of unstructured text in various fields of research, there is a growing need for effective tools that help users analyze and visualize such data. The primary objective of this research is twofold: first, to investigate the needs and challenges faced by researchers working with text data, and second, to design and implement a practical solution that meets those needs.

The initial phase involves an extensive review of the literature, aiming to understand the current landscape of text data visualization, categorize researcher objectives, and identify the processes they follow. This research informs the selection of visualization types and the most suitable Python libraries to implement them.

The final product is made accessible as both a Python package and an open-source repository on GitHub, providing a flexible tool that addresses the key challenges in text data visualization. This thesis not only contributes a functional tool for researchers but also highlights the importance of targeted solutions in handling and visualizing unstructured textual data.

Keywords

Data Visualization, Text Data, Massive, Word Cloud

Contents

List of Figures

List of Tables

NLP	Natural Language Processing
LDA	Latent Dirichlet Allocation
CSV	Comma-Separated Values
TXT	Text File Document

Chapter 1

Introduction

This introductory chapter is organized into three distinct sections, each serving a specific purpose to provide clarity and structure to the thesis.

The first section offers a comprehensive introduction to the theme of the thesis. It explains the topic in detail, outlining the core focus of the research and highlighting its significance. The discussion aims to clarify why the chosen topic is important and why it merits in-depth study. By doing so, it sets the foundation for understanding the research problem and justifies the need for the proposed solution.

The second section focuses on the objectives of the thesis. These objectives are divided into two key parts: one centered on the research phase and the other on the development phase. The research phase involves understanding the needs and current approaches to textual data visualization, while the development phase aims to create a practical solution based on the insights gathered. This section outlines the goals of the thesis, ensuring that the reader understands the dual focus of both exploration and implementation.

Finally, the third section provides an overview of the thesis structure. It explains how the subsequent chapters are organized, guiding the reader through the flow of the research and development processes. This structural outline serves to prepare the reader for what will be covered in each chapter, offering a roadmap to help navigate through the work as a whole.

Together, these sections form a cohesive introduction that not only introduces the subject and objectives but also ensures that the reader has a clear understanding of how the thesis is structured and what to expect in the upcoming chapters.

1.1 Problem Statement

The Digital 2023 Global Overview Report, produced by DataReportal in partnership with Meltwater and We Are Social, revealed that 64.4% of the worldwide population uses the internet, and that the average amount of time spent daily on it is 6 hours and 37 minutes.

The same report also shows that the two main reasons people use the internet are finding information and staying in touch with friends and family. Regarding the latter, roughly 60% of the world's population uses social media. Moreover, Twitter, for example, has a 280-character limit for most of its users, and every minute about 350,000 tweets are sent.

This is just a small fraction of the amount of textual data constantly being produced.

So, in the digital age, an overwhelming amount of textual data is generated daily from various sources, such as social media, news articles, scientific publications, emails, and online documents. This explosion of unstructured text poses a unique challenge and opportunity for researchers, analysts, and organisations seeking to glean valuable insights from this vast repository of information. Massive text visualisation emerges as a powerful approach to navigating through the immense sea of textual data, transforming it into visual representations humans can comprehend, interpret, and leverage for informed decision-making.

Massive text visualisation is a multidisciplinary field that combines elements of natural language processing (NLP), data visualisation, and machine learning to extract meaningful patterns, trends, and relationships from unstructured text. Converting complex textual data into interactive and visually appealing representations empowers users to explore, analyse, and understand the underlying information at a glance. This newfound ability to distil key insights from massive text datasets accelerates the pace of research, facilitates data-driven decision-making, and opens up new avenues for innovation across various domains.

In conclusion, massive text visualization is essential for harnessing the power of textual data. These visualization techniques allow us to extract valuable insights, identify patterns, and better understand large text collections' underlying structure and content. Researchers, analysts, and practitioners can uncover hidden knowledge and make informed decisions based on today's vast textual information by combining computational methods with intuitive visual representations.

Considering all of this, we can begin to understand why Massive Textual Data deserves to be explored. The amount of textual data produced and analyzed is significantly disproportional to each other. In addition, the amount of textual data represented graphically is even smaller.

1.2 Objectives

This thesis pursues two primary objectives. The first is to develop an intuitive solution for data visualization that specifically targets textual data. The goal is to allow users to easily input their text-based datasets and receive tailored visualization options that align with their specific interests and needs. The second objective focuses on understanding the requirements and preferences of researchers when dealing with text data visualization. This involves investigating why certain aspects of the data are of interest for visualization, and analyzing the processes and decision-making involved in these situations. A comprehensive understanding of these factors is crucial for selecting and designing the optimal features for the final visualization program.

1.2.1 Understanding Massive Text Visualization

This step focuses on understanding the existing work done by researchers in the field of text data visualization, identifying their specific needs, and exploring the tools that can help us achieve our own objectives. To accomplish this, we will undertake the following tasks:

- Review the relevant literature: This involves studying academic papers, articles, and reports related to text data visualization to gain insights into current methodologies and tools.
- Identify the general needs of researchers: Determine the common challenges, goals, and requirements researchers face when working with textual data visualization.
- Categorize the objectives of researchers: Break down the various aims and purposes researchers have when visualizing textual data, such as pattern recognition, trend analysis, or data interpretation.
- Categorize the processes used by researchers: Analyze the typical processes and methodologies employed by researchers for preparing and visualizing textual data.
- Track specific types of visualizations used: Document the types of visualizations (e.g., word clouds, sentiment graphs, topic maps) that are commonly applied to text data.
- Identify relevant Python libraries: Explore and compile a list of Python libraries that provide specific visualization functionalities suitable for textual data (e.g., Matplotlib, Seaborn, Plotly, WordCloud).

1.2.2 Text Data Visualization Solution Development

The primary objective of this thesis is to develop a solution specifically designed for the visualization of textual data. This is why an in-depth research phase was conducted, as outlined in the previous section. The goal is to create a solution that focuses exclusively on text data visualization. After thoroughly understanding the users' needs, the next steps in this process are as follows:

- Select the most effective visualization types: Based on the research findings, choose the visualization methods that are most commonly used and best suited for textual data.
- Identify suitable Python libraries: Choose Python libraries that best match the selected visualization types, ensuring they meet the requirements for rendering high-quality visualizations.

- Consider performance and interactivity: Select libraries that offer a balance of execution speed, efficiency, and user interactivity to enhance the overall experience.
- Implement the optimal solution: Develop and integrate the chosen visualizations in a way that allows users to effectively utilize each visualization type for their specific needs.
- Ensure accessibility: Make the final solution accessible by publishing the code both on GitHub and as a Python package, allowing easy integration and use by the wider community.

1.3 Thesis Structure

This thesis is organized into six chapters, each contributing to the understanding and development of the proposed solution for text data visualization. The first chapter introduces the research topic, outlining the study's purpose and objectives, establishing the context for addressing the growing challenges of large-scale textual data visualization.

The second chapter presents a comprehensive literature review, which examines key academic works related to Massive Text Visualization. Through this review, various approaches and tools in the field are analyzed, categorized, and evaluated to better understand the needs and objectives of future users. Additionally, this chapter explores the main data visualization libraries, comparing their capabilities to determine the most suitable options for processing and visualizing text data.

The third chapter describes the methodology adopted for the development of the solution. It explains the detailed process of how the tool was built, including the technologies and libraries chosen for implementation. This chapter also outlines the testing framework used to evaluate the performance of the tool with different datasets, defining the criteria used to measure its effectiveness in addressing the identified needs.

The fourth chapter presents the results of the testing phase. This section provides a detailed analysis of the outcomes from applying the developed solution to various datasets. The performance of the tool is evaluated in terms of its ability to handle large-scale text data and its effectiveness in producing meaningful visualizations. Each type of visualization produced during the tests is assessed based on its clarity, accuracy, and usability.

The fifth chapter is dedicated to discussion. This section critically interprets the results presented in the previous chapter, highlighting the strengths and limitations of the tool. It explores how well the solution meets the initial objectives of the thesis and addresses the user needs identified in the literature review. The chapter also delves into the implications of the findings and provides insights into possible improvements or modifications that could enhance the tool's performance.

The final chapter, the conclusion, summarizes the key achievements of the thesis. It reviews the objectives outlined at the beginning of the research and assesses how well they were met. Additionally, this chapter discusses future directions for expanding the work, suggesting areas where further development or research could improve the tool's capabilities or extend its application to other areas of text data analysis.

Chapter 2

Background and Related Work

This chapter comprises two major sections. The first section provides essential background information to enhance the reader's understanding of the themes explored in this thesis. The second section presents related work and available tools for text data visualization, offering insights into current methodologies and technologies in the field.

2.1 Overview

Massive Text

Refers to a large collection of text-based data, such as emails, social media posts, online reviews, news articles, and more. This type of data, as mentioned before, is becoming increasingly prevalent in today's digital age. Massive text is an extensive volume of textual data characterized by size and complexity. It encompasses a vast array of written information, such as books, articles, reports, social media posts, emails, and other forms of textual content. Massive text datasets often contain millions or even billions of words, making them challenging to process, analyze, and derive insights from traditional methods.

The advent of digital technologies and the widespread availability of textual information have led to massive text collections in various domains. These collections present unique challenges due to their size and unstructured nature. Extracting valuable knowledge from massive text requires advanced natural language processing, text mining, and machine learning techniques.

Massive text datasets offer valuable opportunities for various applications. They can be used for sentiment analysis, topic modeling, opinion mining, information retrieval, trend analysis, and many other tasks. Researchers, analysts, and organizations can harness the power of massive text to gain insights, discover patterns, identify trends, and make informed decisions.

However, working with massive text comes with its challenges. Processing and analyzing such large volumes of data require specialized computational resources, efficient algorithms, and scalable infrastructure. Managing and storing massive text collections can be complex, requiring advanced data management techniques and storage solutions.

Furthermore, ensuring data quality and dealing with noise, ambiguity, and linguistic vari-

ations present additional obstacles when working with massive text. Preprocessing steps, such as text cleaning, normalization, and entity recognition, are crucial for improving the accuracy and reliability of subsequent analyses.

Another challenge lies in extracting meaningful information and insights from massive text collections. With the sheer volume of data, it becomes essential to employ techniques for feature extraction, text summarization, and visualization to distill key patterns and knowledge.

Despite the challenges, the massive text offers immense opportunities for understanding human behavior, social dynamics, and cultural trends. It provides valuable research, business intelligence, market analysis, and policy-making resources. Advanced computational techniques and innovative approaches are continuously being developed to address the challenges and leverage the potential of massive text in various domains.

In summary, massive text refers to vast collections of textual data that pose challenges in processing, analyzing, and extracting insights. While managing and making sense of massive text requires specialized techniques and infrastructure, it offers rich opportunities for knowledge discovery and understanding in diverse fields. Leveraging the power of massive text can unlock valuable information and drive innovation in numerous domains.

Data Visualization

As previously mentioned, the primary reason people use the internet is to find information, which indicates a concern for easy access to knowledge. Even with well-organized and low-volume data sets, it can still be challenging to identify relevant information by simply examining it. Therefore, the purpose of data visualization is to enhance accessibility to data patterns, outliers, trends, and insights for the viewer through the use of charts, graphs, word clouds, and other visual aids.

Data visualization is a powerful tool for transforming raw data into meaningful insights and actionable information. By visually representing data using charts, graphs, maps, and other visual elements, it enhances our ability to understand complex datasets and identify patterns, trends, and relationships.

One of the key advantages of data visualization is its ability to simplify and condense large volumes of data into concise and digestible visual representations. By presenting data visually, it becomes easier to perceive patterns, spot outliers, and make comparisons. For example, a line chart can effectively illustrate the trend of sales over time, while a bar chart can compare the performance of different products or regions.

Data visualization also enables the exploration of multidimensional datasets. By mapping variables to different visual attributes like color, size, and shape, we can visualize multiple dimensions simultaneously. This allows us to uncover hidden insights and correlations

that might be difficult to identify through traditional tabular data.

Furthermore, data visualization promotes effective communication and storytelling. Visual representations have the power to engage and captivate audiences, making complex information more accessible and memorable. By presenting data in a visually appealing and intuitive manner, data visualization helps convey findings and recommendations to stakeholders and decision-makers in a compelling way.

Data visualization is not limited to a single domain or industry. It is utilized in fields such as business, finance, healthcare, education, social sciences, and more. From interactive dashboards that allow users to explore data in real-time to infographics that simplify complex information for public consumption, data visualization plays a crucial role in enabling data-driven decision-making and facilitating data literacy.

However, data visualization also comes with challenges. Choosing the appropriate visualization techniques, ensuring accurate representation, handling missing or inconsistent data, and maintaining visual clarity are among the considerations. Design principles such as selecting appropriate chart types, labeling, and using color effectively are crucial for creating effective and meaningful visualizations.

In conclusion, data visualization is a powerful tool for transforming data into insights. By leveraging visual representations, it enhances understanding, enables exploration, facilitates communication, and empowers decision-making. With the increasing availability of data in today's world, data visualization is more important than ever in helping us extract value and make sense of the vast amount of information at our disposal.

2.1.1 Massive Text Visualization

Massive text visualization is a dynamic and evolving field that focuses on representing and exploring large-scale textual data. With the ever-growing volume of digital content, such as social media posts, news articles, scientific publications, and online documents, the need to derive meaningful insights from this vast sea of unstructured information has become increasingly critical [17].

At its core, massive text visualization aims to transform complex textual data into visual representations easily interpretable by humans. By leveraging a combination of Natural Language Processing (NLP), data visualization techniques, and machine learning algorithms, massive text visualization enables researchers, analysts, and organizations to discover patterns, trends, and correlations hidden within the vast amount of text. One of the primary challenges in massive text visualization is the sheer volume of data. Traditional text analysis methods are ill-equipped to handle such massive datasets efficiently. Therefore, scalable and parallel processing algorithms are crucial to enable real-time exploration and interaction with the data [18].

The unstructured nature of text also presents a significant obstacle. Unlike structured data, which fits neatly into rows and columns, text lacks a standardized format. It contains multiple languages, idioms, slang, and context-specific jargon, making it challenging to organize and represent coherently. NLP techniques, including tokenization, part-of-speech tagging, and named entity recognition, are essential for transforming raw text into structured data that can be visualized effectively.

Another critical aspect of massive text visualization is abstraction and summarization. As the volume of data grows, presenting all the details becomes impractical and overwhelming. Therefore, advanced machine learning algorithms, such as topic modelling and text clustering, help identify key themes and topics within the text, creating concise and informative visual summaries.

Interactive visualization is vital in empowering users to explore massive text datasets effectively. Interactive interfaces allow users to filter, drill down, and zoom in on specific aspects of the data, revealing deeper insights and supporting data-driven decision-making.

Moreover, combining text with other data modalities, such as images, time series data, or geographical information, enriches the visualization and provides a more comprehensive understanding of the underlying patterns and connections.

Massive text visualization has various applications across various industries and domains. In social media analytics, it aids in sentiment analysis, tracking trends, and understanding public opinion. It helps journalists and news organizations extract essential information from vast amounts of news articles quickly. In scientific research, it supports literature reviews, trend analysis, and hypothesis generation. Additionally, business intelligence enables organizations to monitor customer feedback, market trends, and competitor activities [19].

Despite the numerous advancements and promising applications, challenges remain. Ensuring data privacy and ethical considerations when dealing with sensitive textual information is paramount. As text data often contains personally identifiable information, researchers and practitioners must adopt privacy-preserving techniques to safeguard users' identities.

In conclusion, massive text visualization is a powerful and indispensable tool for making sense of the vast amounts of textual data generated daily. By integrating cutting-edge technologies, such as NLP, machine learning, and interactive visualization, this field enables us to derive valuable insights, discover trends, and gain a deeper understanding of the complex world of text. As technology continues to evolve, so will the capabilities of massive text visualization, revolutionizing how we analyze and interact with textual information.

2.2 Related Work

This section is organized into three parts. The first part offers a concise introduction to key papers within the field of Massive Text Visualization. The following subsection categorizes these papers into three main themes: objectives, processing methods, and data visualization. To enhance comprehension, select visualizations from these papers are included in this section. The final part features libraries that are commonly used in data visualization, even though none were specifically designed for text data visualization.

2.2.1 Literature Overview

This section serves as an introduction to the eight most relevant papers concerning this research. These papers are also positioned as the current state-of-the-art within the thesis, each contributing to the field. The intent of this section is to provide a comprehensive overview of the key literature, and understand the needs of visualization of researchers.

Tweetviz - Twitter Data Visualization:

Published in 2014 by the authors Dario Stojanovski, Ivica Dimitrovski, and Gjorgji Madjarov from the Faculty of Computer Science and Engineering at Ss. Cyril and Methodius University, this paper [2] aims to introduce a web tool called Tweetviz.

The purpose of Tweetviz is to enable users to visualize Twitter data effectively. Tweetviz offers a range of interactive visualizations, allowing users to gain insights into various aspects such as Twitter users, keywords, and hashtags. By utilizing the Tweetviz interface, users can input their desired data and explore different visualization techniques. The tool's visualizations intend to provide valuable information and aid in understanding patterns and trends within Twitter data.

The authors acknowledge the growing popularity of social media platforms and highlight the increasing prevalence of microblogging services. Specifically focusing on the Twitter community, they note its wide range of uses in daily life, including sharing thoughts, discussing popular topics, disseminating news, and even marketing.

Given the vast volume of daily tweets generated on Twitter, the authors are motivated to extract valuable knowledge regarding user interests and behavior. Their goal is to detect trends and patterns in information dissemination within this source of data. Recognizing the immense potential for insights hidden within this dataset, the authors aim to uncover meaningful information that can contribute to a deeper understanding of user behaviors and preferences.

To develop Tweetviz, several components were required: frontend design, visualization techniques, and topic distribution analysis. The authors utilized a Python framework for

implementing the Latent Dirichlet Allocation (LDA) topic distribution. In terms of visualization, they leveraged tools such as Google Charts and d3.js. For the frontend interface, the authors opted for a combination of HTML, CSS, and JavaScript.

Acquiring the data is the initial step in data processing, the Twitter REST API was utilized for this purpose. Although the authors acknowledged that the API does not index all tweets, it still provided a substantial number of tweets that were sufficient for the project.

In conclusion, while this paper may not extensively delve into the processing steps, it offers a comprehensive showcase of five different visualizations. These visualizations include distributions, word cloud, and streamgraph for the LDA topic distribution. Through these visual representations, the paper successfully presents the outcomes of the data processing and analysis, shedding light on the underlying patterns and insights derived from the dataset.

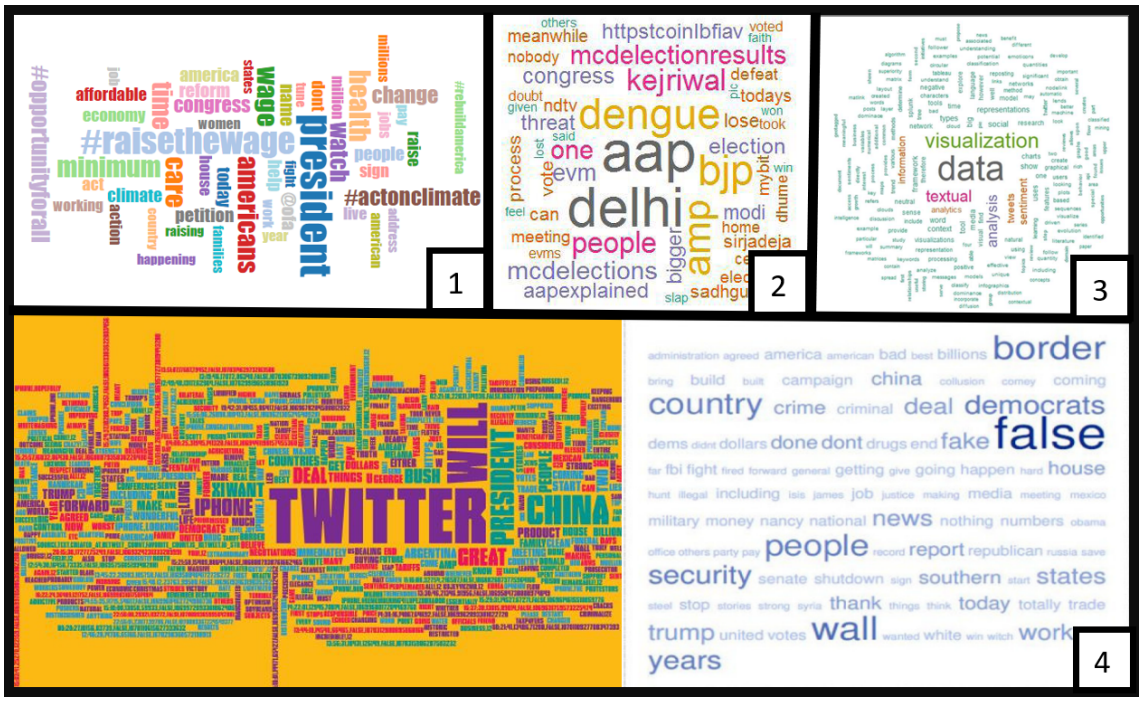

Figure 2.1: Word clouds found in the literature. (1) Frequent keywords from paper [2], (2) Word frequency from [3], (3) Word cloud from paper [4], (4) Image to the left is word cloud before preprocessing and on the right after preprocessing from the paper [5]

.

Twitter Data Clustering and Visualization:

This paper [8], authored by Andrei Sechelea, Tien Do Huu, Evangelos Zimos, and Nikos Deligiannis, affiliated with the Department of Electronics and Informatics at Vrije Universiteit Brussel, was published in 2016. The objective is to introduce a comprehensive system that collects geotagged tweets, performs data processing and analysis, and finally visualizes the outcomes. The incorporation of geotagged tweets specifically serves the

purpose of facilitating the creation of a three-dimensional (3D) environment for the visualization of identified trends.

In the introductory chapter, the authors discuss the evolution of Twitter from a platform designed for personal connections to a global phenomenon generating vast amounts of daily data. They emphasize that since 2009, Twitter has enabled geotagging of tweets, enabling users to capture real-time information from specific regions. This capability presents an invaluable opportunity to verify trends and patterns within distinct geographical locations.

This system implementation involved the utilization of several materials. For data collection, the authors utilized the Twitter REST API in conjunction with a Python script to filter and gather geotagged tweets. The analysis phase employed Python, MapReduce, and Hadoop. The visualization aspect leveraged Gephi to depict topic proximity and OpenStreetMaps for mapping.

As previously mentioned, one of the first steps is always the data acquisition. For this paper the data was collected in the period of one month and encompassed two locations: Brussels and London. Considering only the geotagged tweets, the dataset comprised 250,000 tweets from London and 15,000 tweets from Brussels.

To identify dominant themes within the dataset, the authors employed a clustering method that combined the k-means algorithm with the Density-Based Spatial Clustering of Applications with Noise (DSCAN) algorithm.

In terms of visualization, the paper presents seven distinct visualizations, including 2D/3D maps and clusters. The authors conclude by emphasizing that this system serves as a prototype and lays the foundation for future work across various domains.

Twitter Data Visualization and Sentiment Analysis of Article 370:

Published in 2019, this paper [6] was authored by Rupali Patil, Nishant Gada, and Krisha Gala from the Department of Electronics and Telecommunication at K.J.Somaiya College of Engineering. The objective of the paper is to leverage Twitter as a platform to verify people's opinions regarding the revocation of Article 370 of the Indian Constitution. Additionally, the authors explore its impact on trade in Pakistan, terrorism in both Pakistan and India, as well as the opinions on Kashmir while related to the Article 370. To achieve this, they collected tweets related to these themes and categorized them as positive, neutral, or negative, providing insights into public sentiment.

At the time of writing this paper, Twitter had already amassed 100 million users and was generating a remarkable 500 million tweets daily. Given the vast amount of data available and the techniques of opinion mining, the authors recognized this as an opportune moment to investigate and comprehend the impact of the revocation of Article 370 on both

India and Pakistan. By harnessing the power of Twitter data analysis, the authors aimed to gain valuable insights into the public sentiment and reactions surrounding this event.

For data collection, the authors utilized Tweepy, a Python client, to gather tweets within the period of August 5th to August 30th. Sentiment analysis was conducted using TextBlob, a Python library. In terms of visualization, the authors employed matplotlib and pandas, both Python libraries, to effectively present their findings.

This paper offers a comprehensive analysis through six distinct visualization techniques, including bar graphs, line graphs, histograms, and a pie chart. The authors' analysis uncovers meaningful patterns and insights, enabling them to draw informed conclusions regarding the investigated theme. Based on their analysis, the authors highlight that Pakistani users displayed greater concern regarding the impact of Article 370's revocation on trade, whereas Indian users expressed heightened apprehension about the escalation of terrorism. However, it is noteworthy that despite these concerns, the authors found that Indian users generally maintained a positive perspective on the revocation.

Supervised Sentiment Analysis of Twitter Handle of President Trump with Data Visualization Technique:

In their 2020 publication [5], Kalyan Sahu, Yu Bai, and Yoonsuk Choi from California State University's Computer Engineering Program conducted a study to explore the potential influence of then-president Donald Trump's social media usage on the public approval of his administration. The researchers analyzed tweets posted by the president over a span of approximately three years and compared them with approval ratings obtained from two different websites. By examining this data, the authors aimed to ascertain whether there existed a correlation between the president's social media activity and the public's approval.

The authors highlight that this may be the first time in history a US president had so much access to social media, namely Twitter, and actively used it. This, combined with the availability of approval rate data and the potential for sentiment analysis were essential factors for the authors to undertake this project.

The Twitter data utilized in this project was collected from January 2016 to January 2019 through an archive website that organizes postings based on date, time, and device. The approval rate data was obtained from The Real Clear Politics and subsequently cross-checked with other reliable sources. To process the sentiment analysis, the authors utilized a Python kernel in Jupyter notebook, along with the Pandas and TextBlob libraries.

This paper presents seven diverse visualizations, which enable the authors to draw several conclusions. Firstly, they observe that the use of Twitter does not have an immediate impact on approval ratings; instead, a lag of approximately five days is observed when comparing datasets. Furthermore, the authors suggest that while this approach may be

less effective for assessing mass political opinion compared to consumer products, it could yield more detailed insights when combined with additional data sources such as newspapers and visual media reports.

Tweetviz - Visualization Tweets for Business Intelligence:

Published in 2016 by Bas Sijtsma from the University of Amsterdam, Pernilla Qvarfordt, and Francine Chen from FX Palo Alto Laboratory, this paper [10] introduces Tweetviz, an interactive tool designed to extract valuable business information. Tweetviz offers visualizations of sentiment analysis on tweets related to business locations, identifies other businesses frequented by customers, and estimates customer profiles. The authors aim to provide businesses with a comprehensive tool for gaining insights into customer sentiment and behavior.

In the introductory chapter, the authors highlight the abundance of customer opinions on products and services shared on social media platforms such as Facebook and Twitter. They emphasize the need for automated tools that can efficiently filter, organize, mine, and visualize this vast quantity and variety of data. Thus, the objective of Tweetviz for businesses is to facilitate the process of obtaining customer opinions, which traditionally is time-consuming.

The authors collected a vast amount of data from multiple datasets, comprising 24 million geo-tagged tweets from the San Francisco Bay Area. This data spanned from June 2013 to March 2015 and was stored in a MySQL server. To estimate customer addresses, the authors leveraged Zillow and Flickr. The frontend of the system was developed using the Angular JavaScript framework, while the back-end was supported by Node.js.

This paper includes a screen capture showcasing the Tweetviz interface. The interface allows users to visualize a map displaying various business venues, accompanied by a sentiment scale that ranges from negative to positive.

Analysis and Visualization of Twitter Data Using k-means Clustering:

This paper [3], authored by Neha Garg and Rinkle Rani in 2017, was published by the Department of Computer Science and Engineering at Thapar University. Its main focus was to explore the behavior and structure of a social network by utilizing geotagged tweets. The study involved extracting, refining, analyzing, and visualizing these tweets in a geospatial representation The purpose of incorporating geotagged tweets was to enable the visualization of tweet clusters in specific geographical locations, providing insights based on contextual information.

In the introductory section, the authors emphasize the significant growth of social networks and their role as platforms for users to express, share, and discuss their ideas. They

highlight that social networks have played a pivotal role in the widespread adoption of the internet, serving as a primary reason for many individuals becoming avid internet users. Furthermore, the authors note that social networks can also cater to professional-oriented purposes.

The same section also provides an explanation of microblogging websites, which are services that enable users to post their ideas and opinions within a specified word limit. Specifically, Twitter is highlighted as the microblogging platform used in this study. At the time of the research, Twitter allowed users to compose tweets with a maximum of 140 characters.

In this study, OAuth was employed to extract the required data, which was initially obtained in JSON format and subsequently converted into a data frame format. The analysis, clustering, and visualization tasks were performed using sixteen distinct R packages.

Upon completing the processing, six distinct clusters were identified across India, extending slightly beyond its borders. The visualizations offered in the study comprised three map-based visualizations and a word cloud.

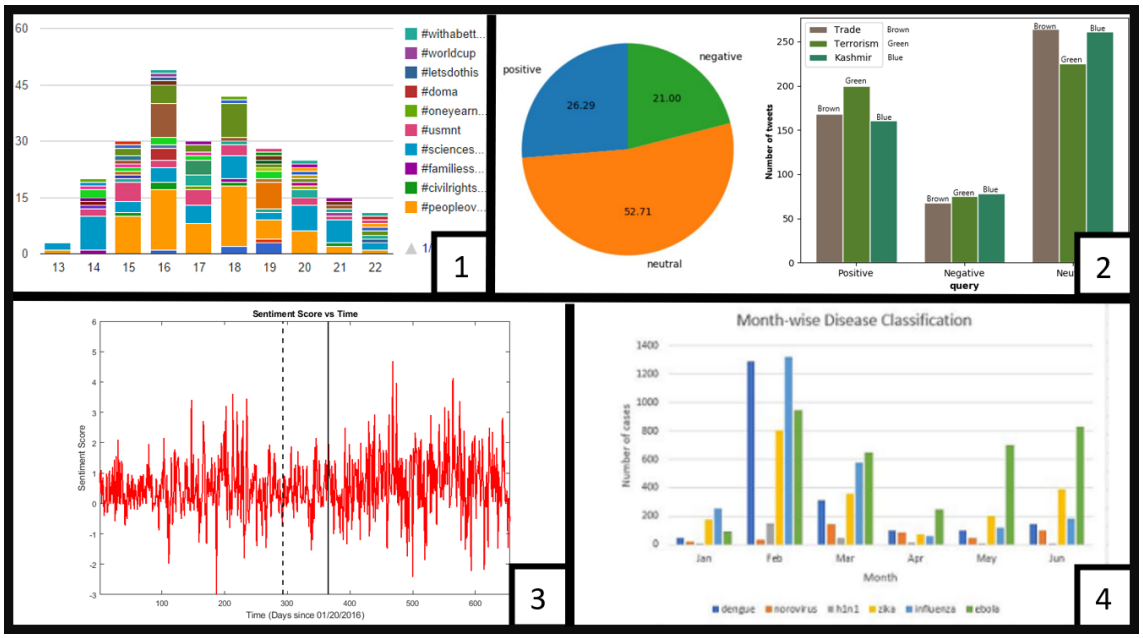

Figure 2.2: Graphs found in the literature. (1) User-hashtag distribution from different times of the day [2], (2) Sentiment Analysis pie chart and Sentiment Analysis on topics from [6], (3) Sentiment score vs time [5], (4) Montwise disease classification and count [7].

Factors Driving the Popularity and Virality of COVID-19 Vaccine Discourse on Twitter - Text Mining and Data Visualization Study:

Published in 2021, this paper [9] was authored by Jueman Zhang from the Polk School of Communications at Long Island University, Yi Wang and Molu Shi from the Department of Communication at the University of Louisville, and Xiuli Wang from the School of New Media at Peking University. The objective of this study was two-fold: firstly, to investigate the popularity and virality of tweets related to the COVID-19 vaccine using text-mining techniques, and secondly, to examine the topic communities of the most liked and retweeted tweets through network analysis and visualization.

The authors emphasize that following the declaration of the COVID-19 outbreak as a pandemic by the World Health Organization in March 2020, the topic generated extensive discussions. The approval of the initial two vaccines further fueled debates, including concerns regarding vaccine safety and effectiveness. Moreover, the authors noted the impact of political polarization in the United States, which reinforced the arguments. Given the abundance of data available on social media platforms, particularly Twitter, the authors were motivated to delve into the topic using this rich source of information.

The data collection for this study focused on tweets from US-based users and specifically included English-language content related to the COVID-19 vaccination. The dataset consisted of a total of 501,531 tweets, which were collected between January 1, 2020, and April 30, 2021.

For this study, the authors utilized topic modeling to identify latent topics within the tweets. They also conducted sentiment analysis to classify the general sentiment expressed in the tweets. In addition, for the 2500 most liked tweets and 2500 most retweeted tweets, the authors employed network analysis and visualization techniques to detect the topics and explore the relationships between them. The study provided two different networks visualizations.

A Machine Learning Approach for Disease Surveillance and Visulization using Twitter Data:

Published in 2019, this paper [17] by Ashwin Ashok, Guruprasad M, Prakash C O, and Shylaja S S from the Department of Computer Science and Engineering at PES University aims to explore the potential of Twitter as a tool for detecting real-time events, with a particular focus on disease surveillance. Their study emphasized the significance of such surveillance in empowering the public to adopt preventive measures, aiding pharmaceutical manufacturers in enhancing sales of disease-specific medications, and ensuring their timely availability. Additionally, health organizations can leverage the findings to implement necessary measures for controlling outbreaks.

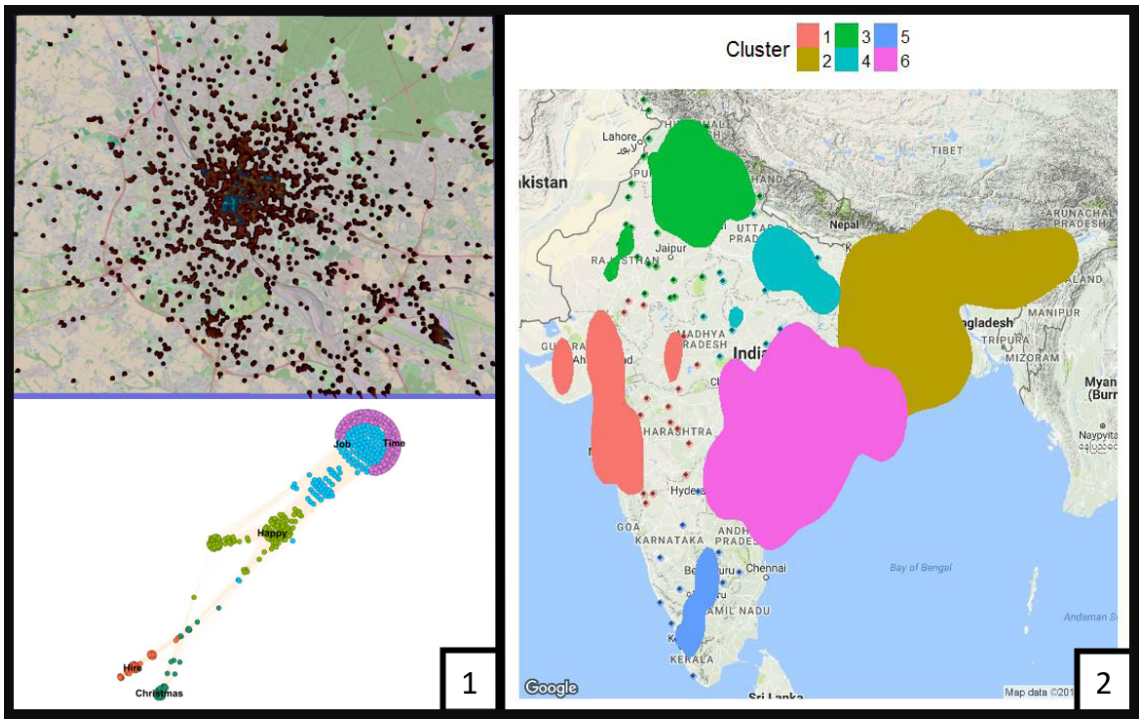

Figure 2.3: Clusters visualization found in the literature. (1) Tweet Density for the city of Brussels and keywords given by clutering algorithm [8], (2) Overlay visualization of Clusters from the paper [6].

The authors highlight that Twitter offers an unprecedented volume of research material, which enables the study of disease outbreaks characterized by a rapid increase in positive cases within a short timeframe. Consequently, the authors argue that Twitter-based methods may offer a faster alternative compared to conventional disease detection approaches.

For this study, a comprehensive collection of tweets was gathered between January 2018 and June 2018, utilizing the Twitter REST API, focusing on dengue, norovirus, H1N1, Zika, influenza, and Ebola. Subsequently, the acquired data was subjected to cleaning procedures using StreamListener and stored in a JSON file. To facilitate further analysis, preprocessing techniques were applied to the data. Finally, clustering techniques were employed to process the tweets.

In addition, this paper presented two visualizations to enhance understanding. Firstly, a distribution map was created to illustrate the geographical spread of the various diseases. Secondly, a heat map was generated, depicting the intensity of these diseases worldwide, providing a visual representation of their prevalence.

2.2.2 Literature Observations

This section provides an overview of the findings from the literature review. In addition to the eight papers discussed in the previous section, numerous other sources were incorporated to enrich the analysis. A major goal of this research was to determine which types of visualizations users are most concerned with and what specific data they aim to visualize. Equally important was gaining an understanding of the processes users typically follow before creating their visualizations. As a result, this section is divided into three key areas: objectives, data processing, and data visualization.

2.2.2.1 Objectives

Given the amount of papers analyzed, it is natural to expect some divergences in the researchers' objectives for their respective datasets. Thus, the purpose of this section is to delineate the main objectives observed in the literature and provide an example of each objective using a selected paper. Despite significant variations in the objectives among the papers, it was still possible to identify common themes and group them into distinct categories, including:

- **Trend Finding** A simple technique used to identify the most common words or phrases in a piece of textual data. It involves analyzing the frequency of specific words or phrases within the text, and then ranking them in order of importance. This technique is useful for identifying the main ideas or themes in a text and gaining a basic understanding of the content. This paper [2] presents an analysis of Twitter data including keywords and hashtags, as well as user-specific information such as posting frequency, most commonly used hashtags, and frequently used words.
- **Topic Modelling** This is one of the most basic concerns while working with textual data. It's a more advance technique than trend finding, that aims to identify not just the most common words or phrases, but also the underlying topics or themes that emerge from the text. It involves using statistical algorithms to group words together based on their co-occurrence patterns in the text, and then clustering these groups into meaningful topics. In this paper [9] a Latent Dirichlet Allocation model was used to identify the leading topics in a data set of 501,531 tweets related to the COVID-19 vaccine. The analysis resulted in the identification of 12 major topics.
- **Sentiment Analysis** There has long been a desire to equip computers with the capability to comprehend human speech. Sentiment Analysis is the objective, where researchers utilize computers to categorize text as either positive, negative, or neutral. This technique can be employed, for instance, to gain an understanding of customers feedback without the need to manually read every individual comment. The objective of [5] is to investigate whether there is a correlation between President

Donald Trump's tweets and his approval rating over a period of three years. To achieve this, the authors classified each of the president's tweets as negative or positive on a scale ranging from -1.0 to 1.0.

- **Geo Location** With the ability to track device location on social media platforms, many papers had the objective of visualizing what were the trending topic in certain regions. This is an example of objective that is always used combined with other. In this paper [7] , the authors aimed to investigate the feasibility of tracking diseases using Twitter by analyzing tweets that mentioned diseases over a period of five months. In addition, they utilized geotagged tweets to determine the location of the diseases.

- **Business Intelligence** Many business owners are interested in understanding their customers' opinions on the products they are selling, as well as their overall customer profile. Textual data, whether it comes from social media, emails, or product feedback, can provide valuable insights to help uncover this information. This paper [10] aims to provide business owners with insights related to their business, such as the sentiment towards certain venues' locations and their customers' profiles.

2.2.2.2 Data Processing

During our research, we came across papers that did not have a practical component or already had their data preprocessed and ready for visualization. However, we chose to focus on practical papers that included textual data and aimed to extract new insights from it.

In order to find new information, researchers often had to preprocess the data, a step that is sometimes referred to as "cleaning the data". This preprocessing step can provide valuable insights into the dataset, but its primary purpose is to make subsequent processing easier.

This section is organized into three parts. Firstly, we will introduce what preprocessing may entail. Secondly, we will describe the different process approaches that researchers may take. Finally, we will present two tables, one for preprocessing and one for the process approach, which indicate which steps were used in the papers we selected.

A. Preprocessing Methods

Texts that are meant to be read and understood by people may include many elements, such as common words like "and", "the", "for", and so on, that are not crucial in a computational context. In order to simplify processing and generate useful results, it's important

to remove or reduce certain text components. This can be achieved through several preprocessing steps, which may include the following:

- **Case Normalization:** A basic preprocessing step involves converting all uppercase characters in the text to their lowercase counterparts.
- **Irrelevant Term Removal:** As mentioned earlier, during preprocessing, some components of the text may be considered less relevant and can be removed. While some papers remove almost all possible components, making the text much simpler, others may find it unnecessary or even detrimental to the process. Some examples of components that can be removed include:
  - Punctuation;
  - Stop words, such as “and”, “the”, “of”, and others;
  - Rare words;
  - Hashtags;
  - URLs;
  - Emoticons;
  - Doubled information.
- **Spelling Correction:** Simply correcting misspelled words that are written wrongly can make the next step clearer.
- **Tokenization:** This step involves dividing the text into smaller units, such as individual words, symbols, and other elements.
- **Stemming:** This refers to the process of shortening words by removing the suffix and returning them to their base form. For example, the words ”change”, ”changing”, ”changer”, and ”changes” can be reduced to their base form ”chang” by removing the suffix ”-ing”, ”-er”, and ”-es”.
- **Lemmatization:** Similar to stemming, lemmatization is used to reduce words, but in a more sophisticated manner. It takes into account vocabulary and morphological analysis to return words to their base form, or ”lemma”. For example, the word ”is” would become ”be” after lemmatization. In the same instance as before “changing”, “changer” and “changes” would simply become “change.”

B. Processing Approach

In the majority of papers, the purpose of preprocessing the data was to prepare it for further processing. While the objectives of the papers varied greatly, the processing techniques mainly focused on one of three processes. This section will introduce these processes, which include:

- **Latent Dirichlet Allocation:** The term 'latent' is defined by Oxford Languages as "existing but not yet developed or manifest; hidden or concealed". In the context of topic modeling, 'latent' refers to the topics that exist within a text but are not yet apparent. Latent Dirichlet Allocation (LDA) is a technique commonly used in topic modeling, which involves dividing the words in a text into a list of topics.
- **Natural Language Processing:** This subfield of Artificial Intelligence (AI) deals with the interaction between computers and human language, whether it is spoken or written. In the papers selected for this study, Natural Language Processing (NLP) techniques were used, specifically Sentiment Analysis which involves identifying and categorizing opinions expressed in text as positive, neutral, or negative.
- **Clustering:** Is an unsupervised machine learning technique that involves grouping data with similar characteristics or features into clusters. Several clustering methods have been used in the literature, including:
  - K-means: This method fixes a specific number (K) of clusters and randomly assigning one centroid to each cluster. The algorithm then calculates the distance between each data point and the centroids and assigns each data point to the closest centroid. The algorithm iteratively recalculates the centroid until the data converges to a stable solution.
  - Density-Based Spatial Clustering of Applications with Noise (DBSCAN): The DBSCAN algorithm starts with an arbitrary point and expands the cluster by adding neighboring points that satisfy the density criteria. When a cluster cannot be expanded any further, the algorithm starts with a new unvisited point and repeats the process until all data points have been visited.
  - Agglomeration: This technique involves starting with each data point as its own cluster, and then calculating the distance between each pair of clusters. The two most similar clusters are then merged into one cluster, and this process continues until the desired number of clusters is reached.

C. Tables

In the literature, we came across a variety of techniques, with some papers utilizing only one technique while others employing multiple techniques, particularly in the preprocessing stage. The tables below summarize the papers based on the Preprocessing Techniques (Table 2.1) and Processing Approach (Table 2.2) they used.

2.2.2.3 Data Visualization

As mentioned earlier, data visualization is a graphical representation of information. While not all of the papers we selected included this component, many did, and for some, it was

Table 2.1: Preprocessing. Each row displays the papers that utilized the preprocessing technique.

Preprocessing Technique	Papers
Case Normalization	[5, 7, 8, 6, 3, 11, 12]
Irrelevant Term Removal	[9, 5, 7, 8, 6, 3, 11, 12]
Spelling Corrections	[5]
Tokenization	[9, 5, 7, 8, 6, 3, 12]
Stemming	[5, 8]
Lemmatization	[9, 5, 12]

Table 2.2: Processing Approach. Each row displays the papers that utilized the processing approaches.

Processing Approach	Papers
Latent Dirichlet Allocation (LDA)	[2, 9, 11]
Natural Language Processing (NLP)	[9, 5, 10, 6, 13]
Clustering	[7, 8, 3]

the primary focus. The visualization techniques used in the literature were generally similar. In this section, we will explain some of these techniques and showcase examples of data visualizations provided by the papers.

- **Word Cloud:** This method (Figure 2.1) visualizes word frequency by displaying words in varying sizes. The smaller words are less frequent, while the larger ones are used more frequently.

- **Graphs:** This method was the most used in the literature (Figure 2.2) as it represent various types of information such as word frequency, topic distribution over time, and comparisons. It was commonly presented in the form of bar graphs, line graphs, or pie chart.

- **Clusters:** Various methods were used in the papers (Figure 2.3) to graphically represent clusters, with the most frequent ones being bubble charts or visualizations within maps.

- **Others:** Other visualization techniques were used less frequently in the papers (Figure 2.4), such as network diagrams, score plots, and distribution maps. And this visualizations overall, were not clear on the information it tried to express.

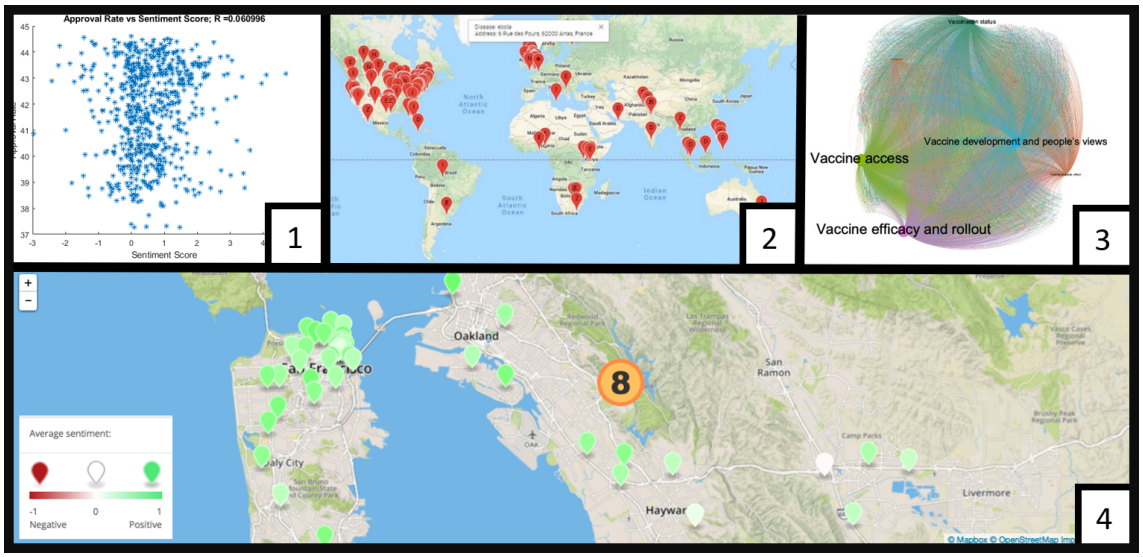

Figure 2.4: Various types of data visualization found in the literature. (1) Approval Rate vs Sentiment Score from [5], (2) Map depicting distribution of various diseases [7], (3) Topic communities of 2500 most retweeted tweets related to the COVID-19 vaccination [9], (4) Color corresponding to the location average sentiment [10].

2.2.3 Python Libraries

Given our intention to utilize Python in this project, it is crucial to explore the existing resources available for data visualization. This section introduces several libraries with significant potential for implementation.

Figure 2.5: Python Libraries. Matplotlib, Seaborn, Bokeh, Plotly, Altair, Plotnine and Folium.

Matplotlib

Matplotlib serves as a versatile library, providing users with the capability to generate static, animated, and interactive visualizations. Among the diverse range of visualizations it supports are line plots, histograms, scatter plots, 3D plots, image plots, and various others, making it a powerful tool for data representation and analysis.

Seaborn

Seaborn is based on Matplotlib, taking a step forward by simplifying the creation of attractive statistical graphics through a high-level interface. While Seaborn primarily focuses on static visualizations, it can be combined with other libraries for interactive components.

Bokeh

Bokeh is a library specifically designed for interactive visualizations. It emphasizes flexibility and simplicity, enabling users to easily create common plots while also providing the capability to design custom visualizations.

Plotly

Plotly is a library dedicated to interactive visualizations, offering support for a wide range of plot types. Beyond its interactive capabilities, Plotly aids in simplifying data comprehension, featuring tools like a hover tool for identifying outliers and anomalies. Known also for its visual, Plotly allows extensive customizations to enhance the aesthetics of visualizations.

Altair

The Vega-Altair or simply, Altair is a library built on top of Vega-Lite, which utilizes a JSON-specific syntax. Altair promises to deliver effective and aesthetically pleasing data visualizations with minimal coding effort.

Pygal

Pygal is a library designed for creating interactive plots and charts. One of its notable characteristics is its strong capability for producing Scalable Vector Graphics (SVG) visualizations.

Geoplotlib

Geoplotlib specializes in map rendering, offering a straightforward API for visualizing data on OpenStreetMap tiles. The library includes implementations for dot maps and kernel density visualizations, enhancing its versatility and effectiveness in geographical data representation.

Plotnine

Based on the R package Ggplot2, this library allows users to create complex and customized plots with simplicity, enabling efficient exploration and communication of data insights through clear and concise visual representations.

Folium

Folium is a library that simplifies the creation of interactive leaflet maps. Leveraging the Leaflet.js library, Folium allows users to generate maps embedded with various markers, pop-ups, and other features, enhancing the visualization of geospatial data in Python applications.

Missingno

Missingno is a library for visualizing missing data in datasets. It provides informative visualizations, such as bar charts and heatmaps, to highlight missing or null values in tabular data. This allows users to quickly identify patterns and assess the completeness of their datasets.

Chapter 3

Methodology

This chapter is structured into three main sections. The first section presents the visualization types selected for progression into the development stage, along with a brief explanation of the rationale behind the choice. This selection process considered the specific needs of users and the objectives of the project seen in the chapter before.

The second section serves as the core of the chapter, detailing the development process itself. Here, we outline the materials used, including both hardware and software specifications, and the libraries that were initially chosen for implementation. Additionally, this section discusses the overall steps planned for future development. We also explain the metrics we deemed necessary for guiding the development process and for evaluating the tool's effectiveness in subsequent testing.

Finally, the last section introduces the final testing phase, where the developed tool is applied to a larger dataset. This segment describes the planned approach for evaluating how the tool performs under the conditions of processing extensive, unprocessed data. Overall, this chapter provides a comprehensive framework for the methodologies employed throughout the development of the text data visualization tool.

3.1 Visualization Types

As mentioned previously, this project is centered on visualizations specifically tailored for text data. After thoroughly reviewing a substantial number of research papers, we observed that while researchers employed a variety of libraries, they consistently leaned towards similar types of visualizations. Our analysis of these libraries indicated that some provide a broader range of features than others, varying in important aspects such as interactivity, execution time, and the depth of information displayed.

In light of these observations, we decided to focus our efforts on the two most prevalent visualization types: Word Clouds and Graphs. To facilitate this focus, we compiled a comprehensive list of the libraries that are most commonly used for each visualization type. But first, we identified ten different visualization types to include in our study, which are

1 - Word Cloud

2 - Bar Plot

3 - Pie Chart

4 - Bubble Chart

5 - Line Chart

6 - Multiple Lines Chart

7 - Grouped Bar Plot

8 - Stacked Bar Plot

9 - Scatter Plot

10 - Data Quality

These visualization types are commonly featured in numerous research papers and can be applied to a variety of objectives. The only exception is the Data Quality visualization, which we chose to include to assist users in identifying where their data may be lacking. For our analysis, we utilized eight different libraries: Bokeh, Matplotlib, Plotly, Plotnine, Pygal, Seaborn, Vega-Altair, and Missingo, with the latter being exclusively dedicated to the Data Quality visualization type. Additionally, we plan to employ the Word Cloud tool from Python for the Word Cloud visualization, as none of the other libraries offer this specific type of visualization.

Another aspect we aimed to determine was whether the program should offer the user only one library per data visualization type or include multiple libraries, allowing users to choose based on their specific needs. This decision would depend on our observations of performance differences among the libraries. If we found significant variations in performance, we would opt to provide users with a selection of libraries for each visualization type, enabling them to make informed choices that align with their requirements.

3.2 Data Visualization Solution Development

After reviewing the literature and determining the best visualization types for this project, the next step was to develop the solution for textual data visualization. This section outlines the steps taken to achieve this goal.

First, it presents the materials and tools used during the development and testing stages. Then, it details which libraries were chosen for each visualization type. Next, it describes how the development process was planned, including the key steps and decisions. It also outlines the metrics used to assess the effectiveness of the solution. Finally, it explains the final test that was planned, which involved applying the solution to a larger dataset to evaluate its performance.

3.2.1 Materials

Since this project was developed and utilized entirely on a software level, the physical materials required were minimal. The development and testing were carried out on a Dell Inspiron 7560 laptop, equipped with an Intel(R) Core(TM) i7-7500U CPU running at 2.70GHz, 16 GB of RAM, and the Windows 10 Home operating system. This hardware setup provided sufficient computational power to develop and test the solution efficiently.

On the software side, the development environment used was Visual Studio Code, version 1.93.1, chosen for its flexibility in managing Python-based projects. Python version 3.12.2 was employed as the main programming language due to its extensive library support, particularly in the areas of data analysis and visualization.

Several key Python libraries and dependencies played a crucial role in the development process, including:

- Pandas and NumPy for efficient data manipulation and numerical operations.
- TimeIt for measuring code execution time to ensure optimal performance.
- Bokeh, Matplotlib, Plotly, Seaborn, Altair, Missingno and Wordcloud for the visualizations.

To develop the Python package, setuptools wheel, and twine were used. The package was then made available on PyPI for easy installation via pip. For version control, GitHub was employed throughout the development process, and later, the complete code was made publicly accessible on GitHub for broader use and collaboration.

3.2.2 Libraries Choice

As previously mentioned, we selected ten different visualization types, many of which could be generated using various libraries. To ensure we chose the best tools for each visualization, we decided to test a range of libraries to compare their performance, features, and overall suitability. While we initially selected eight libraries for testing, we made the decision to assess their effectiveness first before committing to all of them.

Given that some libraries offered overlapping functionality, we opted to exclude the final two—Plotnine and Pygal—from the evaluation if the other libraries met our requirements. Both of these libraries provided similar visualization types already covered by at least one other option, allowing us to streamline the development process and focus on the most efficient tools. This approach ensured that we concentrated our efforts on the libraries that delivered the best results for the project.

The final list of libraries available for testing for each data visualization type (Table 3.1) is as follows:

Data Visualization Types	
Data Visualization Type	Libraries/Tools Options
Word Cloud	Wordcloud
Bar Plot	Plotly, Matplotlib, Bokeh, Vega-Altair, Seaborn, Pygal, Plotnine
Pie Chart	Plotly, Matplotlib, Bokeh, Vega-Altair, Pygal
Bubble Chart	Plotly
Line Chart	Plotly, Matplotlib, Bokeh, Vega-Altair, Seaborn, Pygal, Plotnine
Multiple Line Chart	Plotly, Bokeh
Grouped Bar Plot	Vega-Altair, Seaborn
Stacked Bar Plot	Plotly, Vega-Altair
Scatter Plot	Plotly, Matplotlib, Bokeh, Vega-Altair, Seaborn, Plotnine
Data Quality	Missingno

Table 3.1: Data Visualization Types and Libraries/Tools Options

3.2.3 Development

For the development stage, we utilized the materials listed above, along with two simple, pre-processed datasets in CSV format and a text file. These datasets were interrelated but structured differently to ensure that we could test all the visualization types provided by the tool. Given that the visualization types were designed for different needs, this diversity in data formats was necessary.

The text file used was the book Harry Potter and the Prisoner of Azkaban by J.K. Rowling. This file had already undergone pre-processing, including the removal of stopwords, irrelevant terms, case normalization, tokenization, and lemmatization. This ensured that the data was clean and ready for analysis.

The two CSV files related to the entire Harry Potter series, focusing on word frequency analysis. The first CSV file contained the 22 most frequent words across the whole series. It included data points such as the total frequency of each word, its occurrence in each individual book, the chapter and book with the highest frequency, and a categorization of the words.

The second CSV file listed the five most frequent words but with a different structure, featuring only one column for the books and a corresponding column for the frequency of each word, resulting in a file with 35 rows, five for each book. These variations in structure allowed us to test the flexibility of the visualization tool across different types of datasets.

A key aspect of the project was determining whether the program should offer users a single library per visualization type or multiple library options, allowing users to choose based on their specific requirements. This decision hinged on our evaluation of the performance differences among the libraries. If we observed significant variations in performance, such as speed, interactivity, or aesthetic quality, we would provide users with

multiple library options for each visualization type, enabling them to make an informed choice based on their needs (e.g., faster rendering versus greater interactivity).

With this in mind, we first tested the libraries and visualization types individually, comparing their functionality and performance. Based on the results, we then decided between two options: (1) retaining only one library per visualization type for simplicity and ease of use, or (2) offering multiple libraries for some visualization types to accommodate different user preferences and requirements.

3.2.4 Metrics

To determine whether we should design the program to include all libraries or select a single library for each visualization type, we established some criteria, some types had more than others, which are the following:

- **Execution Time:**

  This metric is straightforward: it measures the time taken by each library to execute a specific visualization type. By applying this approach, we identified which libraries performed faster and evaluated the differences in their execution times. We utilized the Python tool TimeIt to conduct our tests, timing each library's performance for each visualization type across 500 iterations and calculating the average time for more accurate results.

- **Execution Time with Image Generation:**

  This metric is similar to the previous one, but it also considers the time required to fully generate the image. Given the differing results from this test compared to the previous one, we decided to prioritize this approach. However, this test cannot be repeated as many times due to the higher computational load involved in image generation, which is why we used both metrics.

- **Design Ranking:**

  This metric evaluates the amount of information provided by each library, as some libraries offer more comprehensive details than others. We chose to rank this aspect since some libraries may present the same information. While we also considered the overall appearance of the visualizations, we prioritized clarity and accessibility for the user above aesthetic factors.

- **Interactive:**

  This metric is the most straightforward, as it addresses a simple yes or no question regarding the library's visualization capabilities. Specifically, it determines whether the visualization type offered by the library is interactive or not. Interactivity can significantly enhance the user experience by allowing users to engage with the data

in real time, explore different aspects of the visualization, and gain deeper insights. By evaluating this metric, we can assess the extent to which each library supports interactive features, helping users make informed choices about which tools best meet their needs for dynamic and engaging visualizations.

- **Limitations:**

  This metric serves to identify any limitations that a particular library may have compared to others. By assessing these limitations, we can better understand the constraints of each library and determine how they might impact functionality and user experience. Recognizing these differences allows us to make informed decisions about which libraries are best suited for specific visualization needs.

3.3 Test Large Data

As previously mentioned, we also found it important to assess how the program would handle a larger, unprocessed dataset to better understand its performance and scalability. For this purpose, we selected the 960K Spotify Songs with Lyrics dataset, available on Kaggle. This dataset provided a substantial amount of raw data, allowing us to test the program's efficiency and robustness in processing and visualizing large-scale datasets.

The dataset was divided into two main parts: the first, titled "Songs with Attributes and Lyrics," was 1.54GB in size and included song metadata, such as attributes and corresponding lyrics. The second dataset, "Songs with Lyrics and Timestamps," was even larger, at 2.42GB, and included both lyrics and timestamps for each song. These two datasets offered a comprehensive test case, covering various aspects of data complexity—from textual information (lyrics) to structured metadata (attributes and timestamps).

By using this large and unprocessed dataset, we aimed to evaluate how well the tool could manage more demanding, real-world data and whether it could effectively scale its visualizations without compromising performance. This test helped highlight potential areas for improvement in both data handling and visualization efficiency, providing valuable insights for further refinement of the tool.

Chapter 4

Results

This chapter is divided into three main sections. The first, Visualization Types Results, presents the outcomes for each individual visualization type. Some visualization types are explored through multiple libraries, while others utilize only a single library. The second section, Final Program, provides a comprehensive overview of how the final version of the program was structured and developed, showcasing its key features and functionality. Lastly, the third section, discusses the outcomes obtained from testing the final program with a larger, unprocessed dataset, examining how it performed and any challenges encountered.

4.1 Visualization Types Results

In this section, we present the results for each individual visualization type. We provide detailed examples and outline the criteria we deemed essential for evaluating each type. To ensure clarity and thoroughness, this section is organized into ten distinct subsections, each dedicated to a specific visualization type.

4.1.1 Word Cloud

Among the selected tools, only one was specifically designed for generating Word Clouds: the Word Cloud tool itself. This tool offers extensive customization options, which we explored in two primary configurations:

1. **Default:** In this mode, the words are displayed on a standard rectangular image. We tested various color palettes to assess their impact on visual clarity and aesthetic appeal.
2. **Image Contour:** This mode allows words to be shaped according to a specific contour derived from an image. For this test, we used a simple Harry Potter logo as the contour image, given that our dataset was related to the Harry Potter book series. We also applied different color palettes to this configuration to evaluate how well the words integrated with the logo and how color choices influenced the overall design.

Examples of Word Clouds generated with different configurations are presented below. Each version demonstrates the impact of varying color palettes and the effectiveness of

contour shaping inspired by the Harry Potter logo. This comparison showcases the tool's flexibility and the visual results achievable through different customizations.

Figure 4.1: Harry Potter and the Prisioner of Askaban Default Contour Word Cloud. Color palette: Accent, Blues r, BrBG r, BuGn, BuPu r, Pastel1, PuRd r, twilight shifted

Figure 4.2: Harry Potter and the Prisioner of Askaban Image Contour Word Cloud. Color palette: Accent, Blues r, PuRd r, twilight shifted

4.1.2 Bar Plot

To evaluate and select the most suitable library for this type of visualization, bar plot, we tested five prominent libraries: Bokeh(Figure 4.5), Matplotlib(Figure 4.3), Plotly(Figure 4.4), Seaborn(Figure 4.6), and Vega-Altair(Figure 4.7). Our selection for this visualization type was guided by the following criteria:

- Execution Time
- Execution Time while Generating Image
- Overall Design
- Interactivity

Below it's displayed the visualizations produced by each library, accompanied by a comprehensive table (Table 4.1) summarizing their performance against the criteria mentioned above. This comparison aims to provide a clear view of each library's strengths and weaknesses.

Bar Plot Results				
Library	Execution Time	Execution Time with Image Generation	Design Ranking	Interactive
Bokeh	0.0181055s	0.3161025s	4	
Matplotlib	0.1960951s	0.5870333s	3	
Plotly	1.0900968s	1.670982s	1	Yes
Seaborn	0.19153519s	0.482209s	3	
Vega-Altair	0.0066557s	0.428929s	2	

Table 4.1: Bar Plot Results

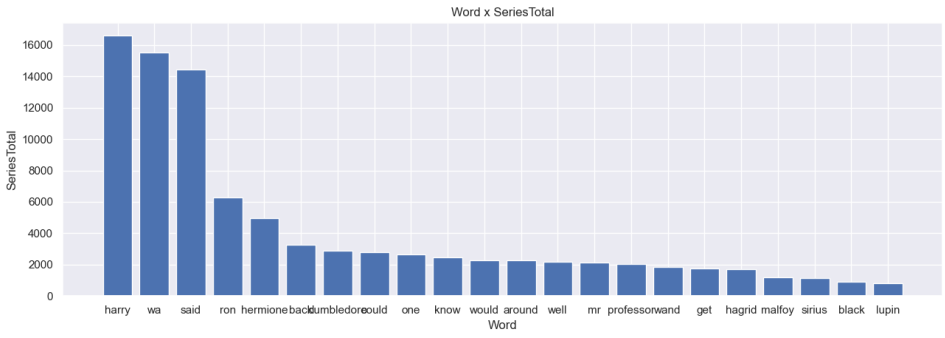

Figure 4.3: Matplotlib Bar Plot, x-axis: most frequent words, y - axis: word frequency on the series

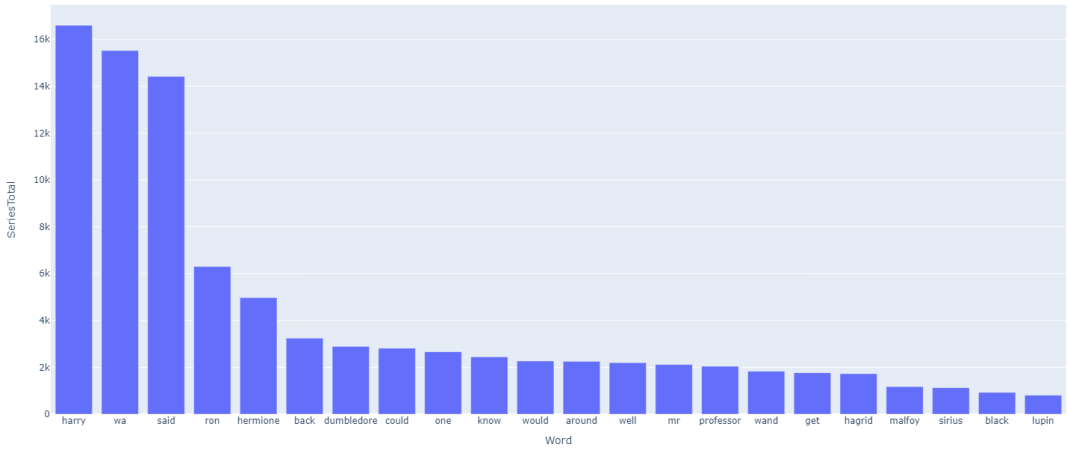

Figure 4.4: Plotly Bar Plot, x-axis: most frequent words, y - axis: word frequency on the series

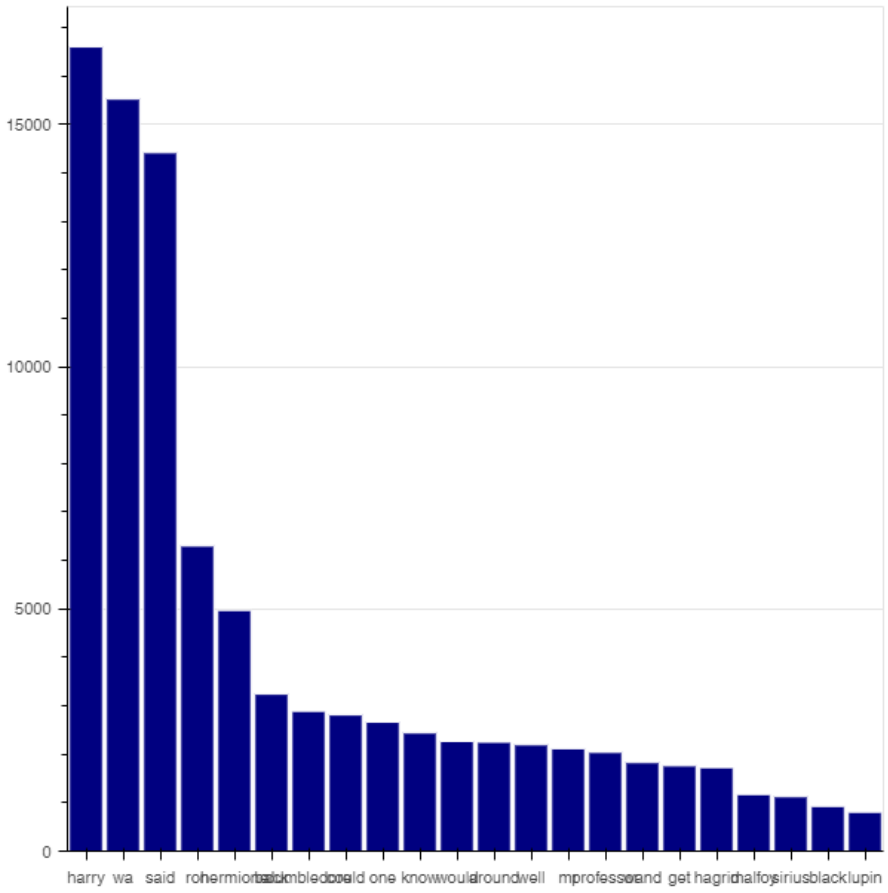

Figure 4.5: Bokeh Bar Plot, x-axis: most frequent words, y - axis: word frequency on the series

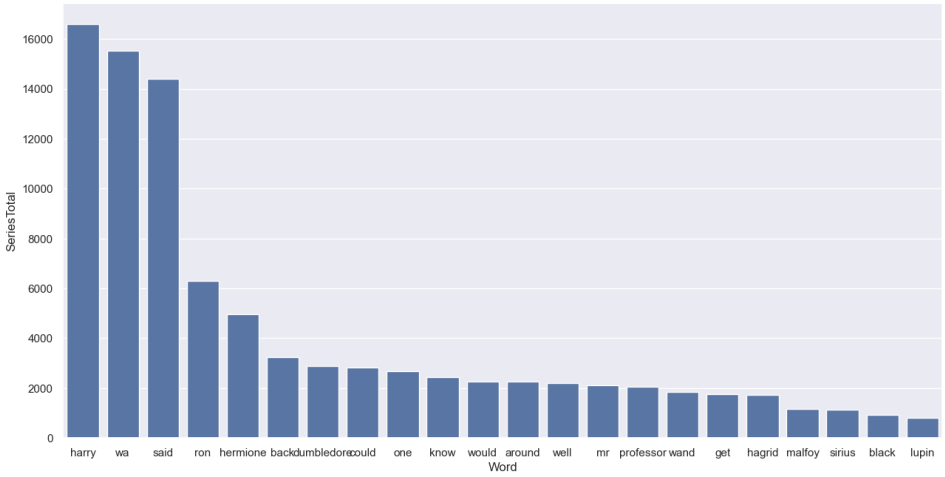

Figure 4.6: Seaborn Bar Plot, x-axis: most frequent words, y - axis: word frequency on the series

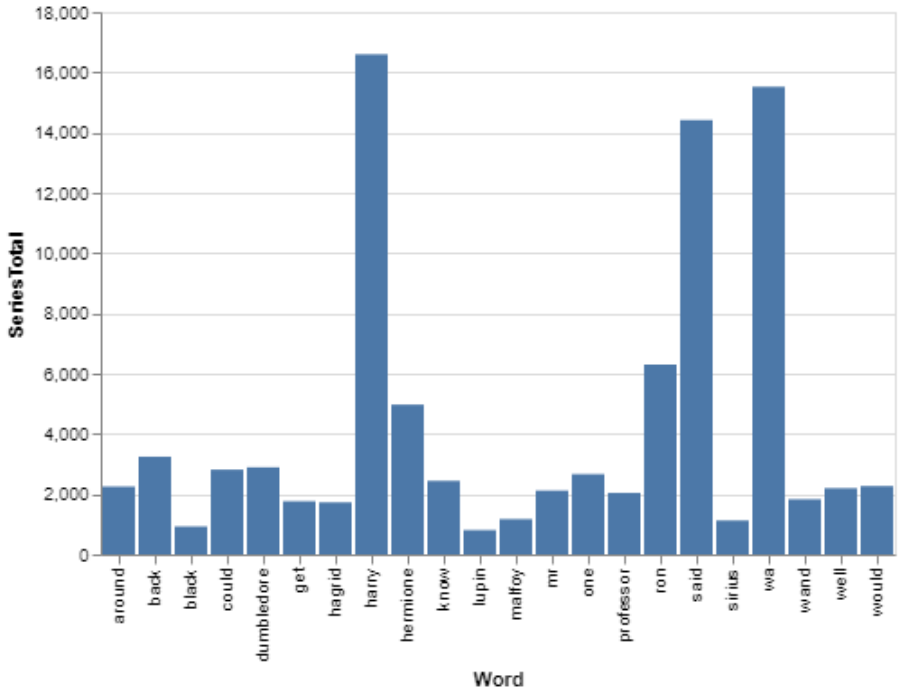

Figure 4.7: Vega-Altair Bar Plot, x-axis: most frequent words, y - axis: word frequency on the series

4.1.3 Pie Chart

For the pie chart visualization, we evaluated four different libraries: Bokeh (Figure 4.10), Matplotlib(Figure 4.8), Plotly(Figure 4.9), and Vega-Altair(Figure 4.11). Our evaluation process was based on several key criteria:

- Execution Time
- Execution Time while Generating Image
- Overall Design
- Interactivity
- Library Limitations

The evaluation results are outlined below, featuring the pie charts produced by each library along with a comprehensive table (Table 4.2) summarizing their performance based on the established criteria. This analysis provides insight into the strengths and limitations of each library regarding pie chart visualizations.

Pie Chart Results					
Library	Execution Time	Execution Time with Image Generation	Design Ranking	Interactive	Limitations
Bokeh	0.0186442s	0.3543881s	4		Max. 20 rows
Matplotlib	0.2327091s	0.7125657s	3		
Plotly	1.0966642s	2.1857749s	1	Yes	
Vega-Altair	0.0004343s	0.3012513s	2		

Table 4.2: Pie Chart Results

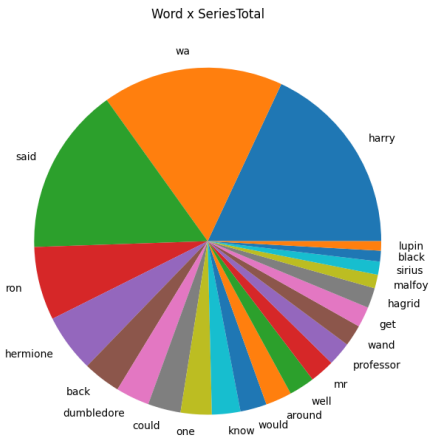

Figure 4.8: Matplotlib Pie Chart, labels: most frequent words, values: series total amount

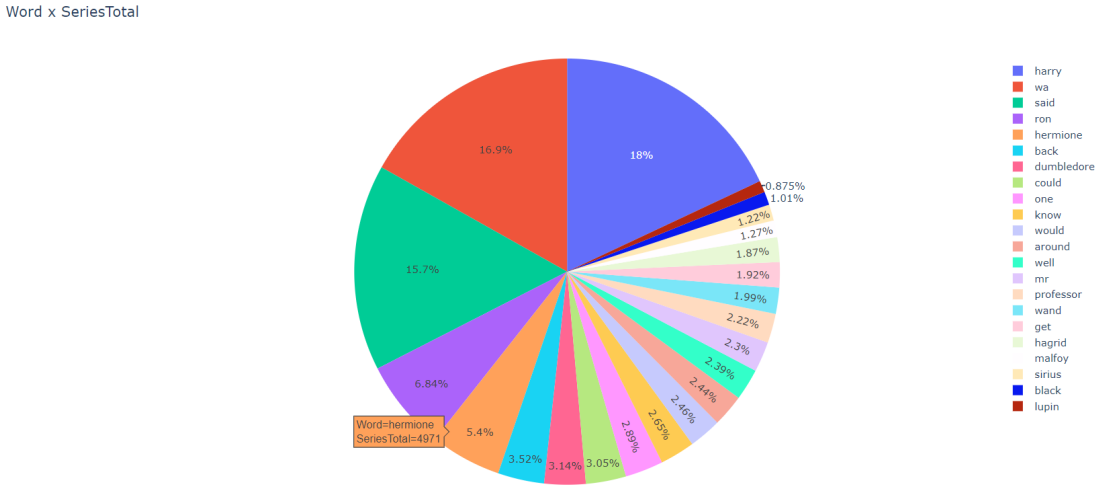

Figure 4.9: Plotly Pie Chart, labels: most frequent words, values: series total amount

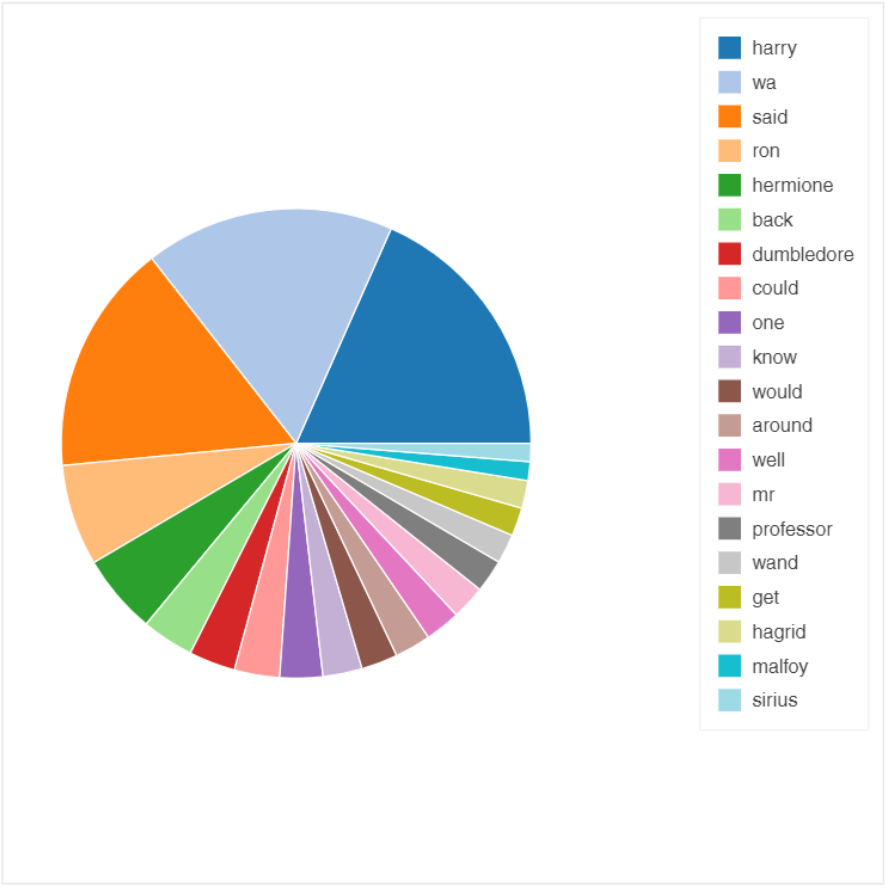

Figure 4.10: Bokeh Pie Chart, labels: most frequent words, values: series total amount

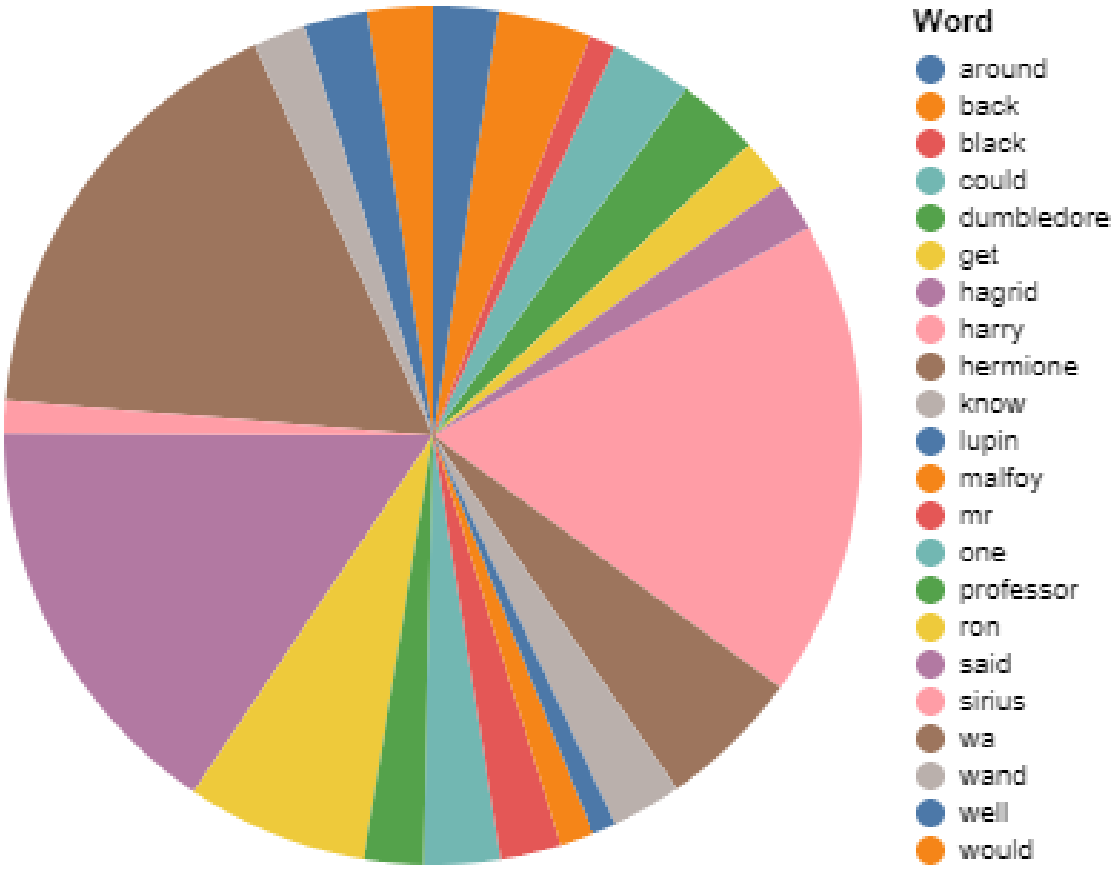

Figure 4.11: Vega-Altair Pie Chart, labels: most frequent words, values: series total amount

4.1.4 Bubble Chart

The Bubble Chart another type of visualization that was supported by only one of the libraries we evaluated: Plotly. Recognizing that Plotly was the sole option for creating Bubble Charts, our focus was on ensuring that the resulting visualizations met our quality standards.

We tested Plotly's capabilities for generating Bubble Charts and found the output to be highly satisfactory. The chart's clarity, functionality, and visual appeal were all evaluated to ensure it met our expectations. Given the positive results from these tests, we decided to adopt Plotly for this visualization type in the final version of our program.

The Bubble Chart produced using Plotly is presented below (Figure 4.12), exemplifying the library's effectiveness in creating engaging visualizations. The inclusion of this chart type in the final program was influenced by its demonstrated capabilities.

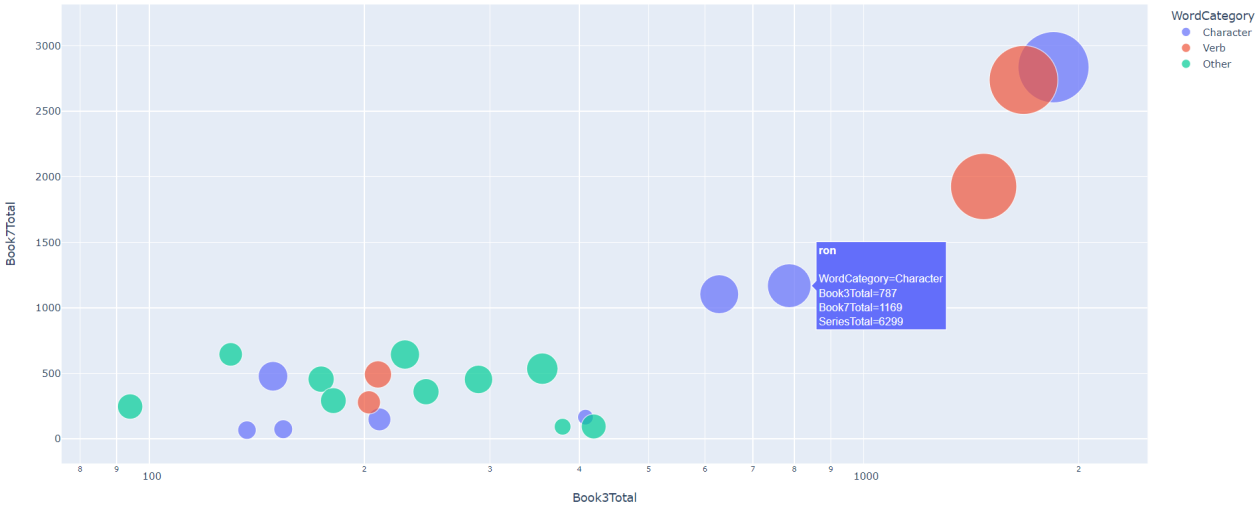

Figure 4.12: Plotly Bubble Chart, x-axis: Word amount on Book 3, y-axis: Word amount on Book 7, bubble size: Word amount on whole series, color: word category

4.1.5 Line Chart

In our analysis of line chart visualizations, we evaluated five different libraries: Bokeh (Figure 4.15), Matplotlib(Figure 4.13), Plotly(Figure 4.14), Seaborn(Figure 4.16), and Vega-Altair(Figure 4.17). Each of these libraries has its unique strengths and weaknesses, so we carefully considered multiple aspects to determine their suitability for this particular visualization type. Specifically, we assessed each library based on four key criteria:

- Execution Time
- Execution Time while Generating Image
- Overall Design
- Interactivity
- Library Limitation

Below, we provide a detailed table (Table 4.3) summarizing the results and observations from our tests, along with the corresponding images generated by each library for the line chart visualization. This comparison should offer a clearer understanding of how each library performs across the chosen metrics.

Line Chart Results					
Library	Execution Time	Execution Time with Image Generation	Design Ranking	Interactive	Limitation
Bokeh	0.0352845s	0.5132982s	2		Only numeric input
Matplotlib	0.2798482s	0.8207036s	2		
Plotly	1.3963225s	2.3599831s	1	Yes	
Seaborn	0.1786253s	0.4121637s	1		
Vega-Altair	0.0087531s	0.7292989s	2		

Table 4.3: Line Chart Results

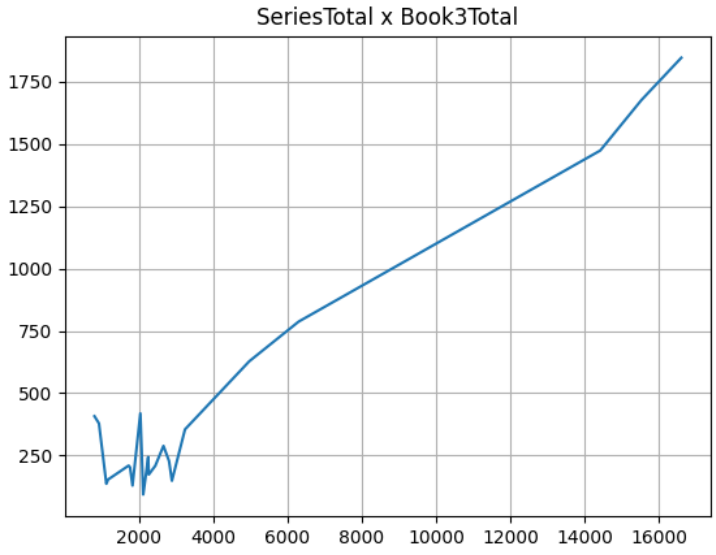

Figure 4.13: Matplotlib Line Chart, x-axis: series total amount of most frequent words, y-axis: book 3 amount of series most frequent words.

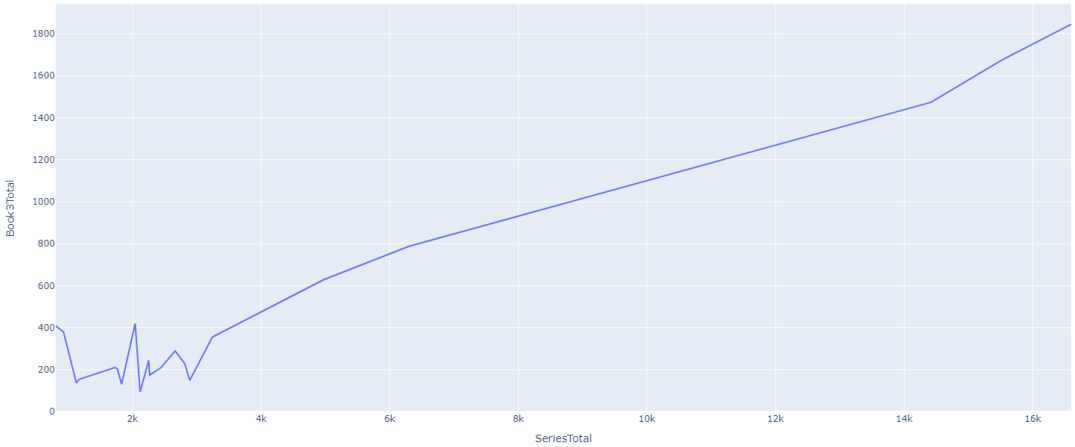

Figure 4.14: Plotly Line Chart, x-axis: series total amount of most frequent words, y-axis: book 3 amount of series most frequent words.

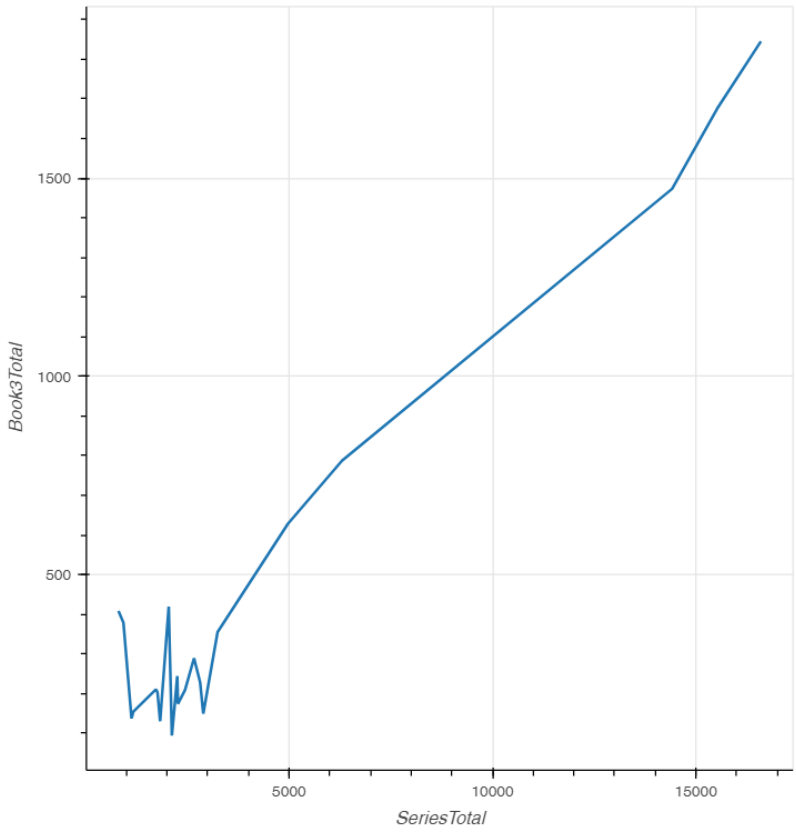

Figure 4.15: Bokeh Line Chart, x-axis: series total amount of most frequent words, y-axis: book 3 amount of series most frequent words.

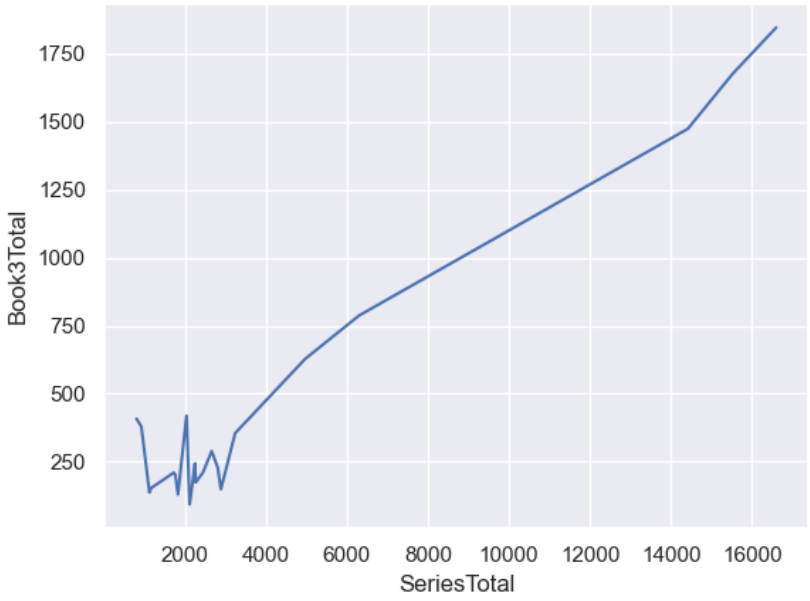

Figure 4.16: Seaborn Line Chart, x-axis: series total amount of most frequent words, y-axis: book 3 amount of series most frequent words.

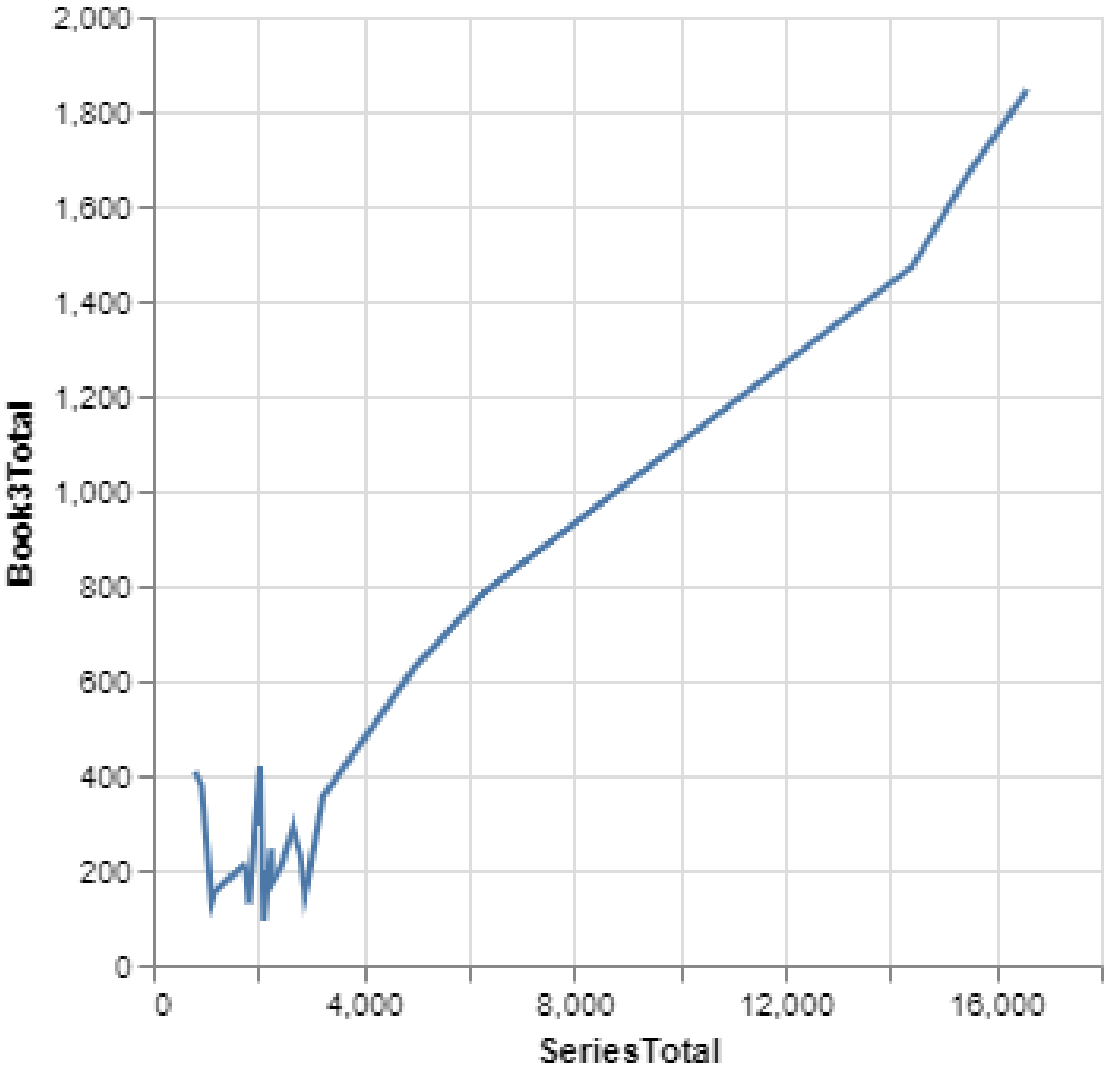

Figure 4.17: Vega-Altair Line Chart, x-axis: series total amount of most frequent words, y-axis: book 3 amount of series most frequent words.

4.1.6 Multiple Line Chart

For the multiple line chart visualization, only two libraries were suitable: Bokeh (Figure 4.18) and Plotly(Figure 4.19). However, Bokeh supports only numeric inputs, which posed a limitation since this project focuses primarily on textual data. As a result, Plotly emerged as the more effective option for our needs. Nonetheless, we've included visualizations from both libraries for comparison.

Upon review, it's clear that Plotly not only handled the data better but also provided a more polished, interactive experience, which proved especially useful for this type of visualization.

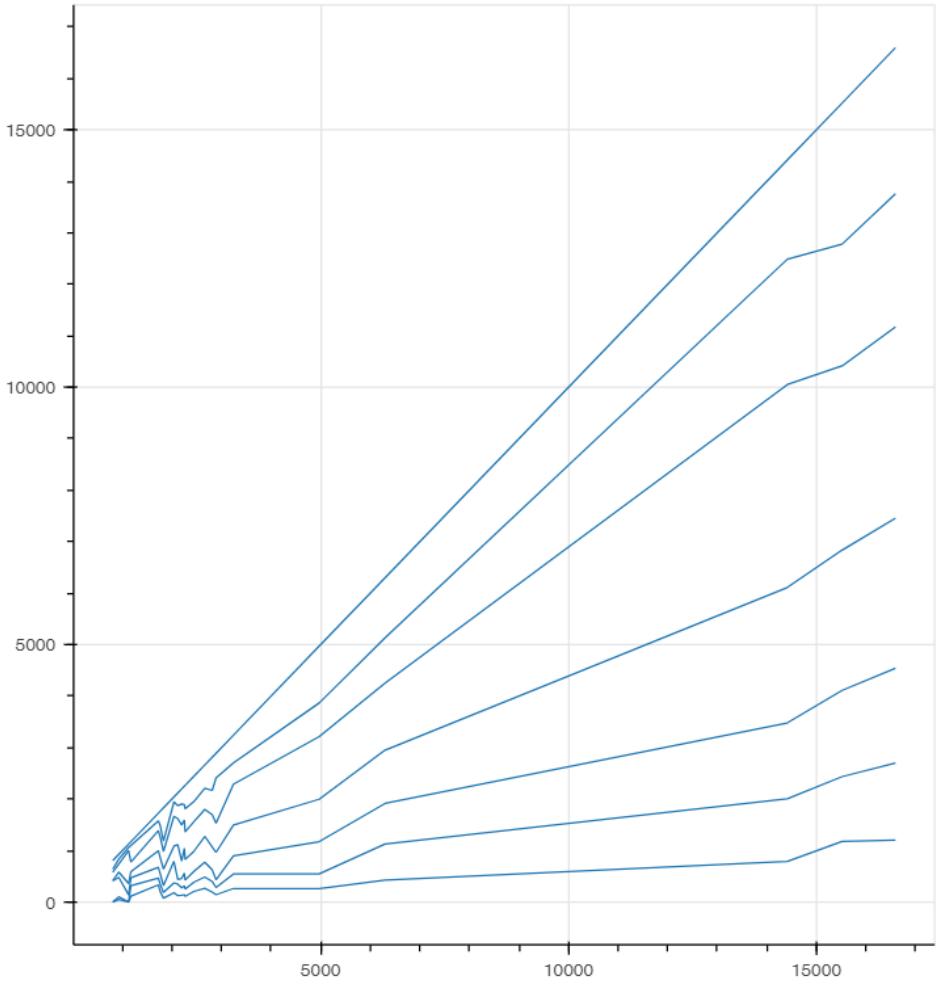

Figure 4.18: Bokeh Multiple Line Chart, x-axis: series amount of most frequent used words, y-axis: amount of word by each individual book.

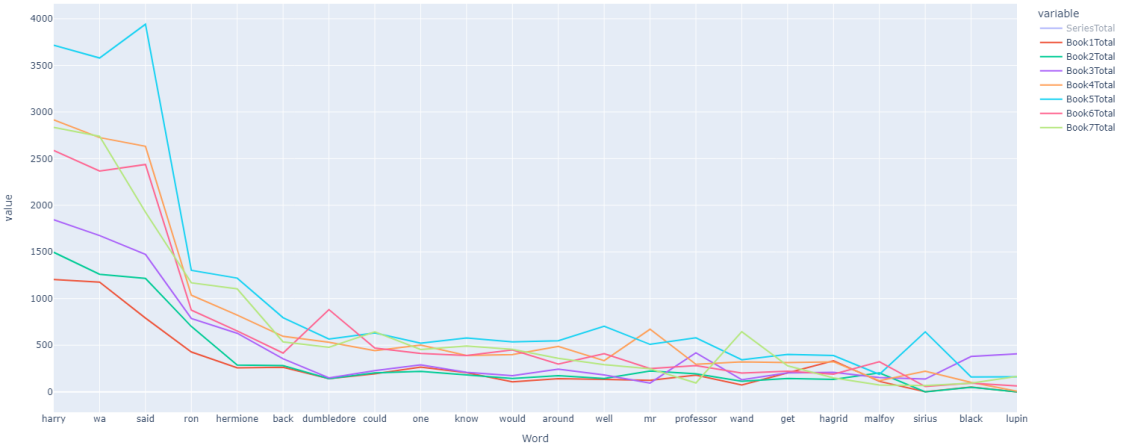

Figure 4.19: Plotly Multiple Line Chart, x-axis: most frequent words, y-axis: amount by book (selected), amount by series(not selected).

4.1.7 Grouped Bar Plot

For the grouped bar plot visualization, we utilized two libraries: Seaborn (Figure 4.20) and Vega-Altair (Figure 4.21). In our evaluation, we considered key factors such as execution time and the execution time with image generation.

Below is a table (Table 4.4) summarizing the timing results, along with the images generated by each library for comparison.

Grouped Bar Plot Results		
Library	Execution Time	Execution Time with Image Generation
Seaborn	0.1786253s	0.4121637s
Vega-Altair	0.0087531s	0.7292989s

Table 4.4: Grouped Bar Plot Results

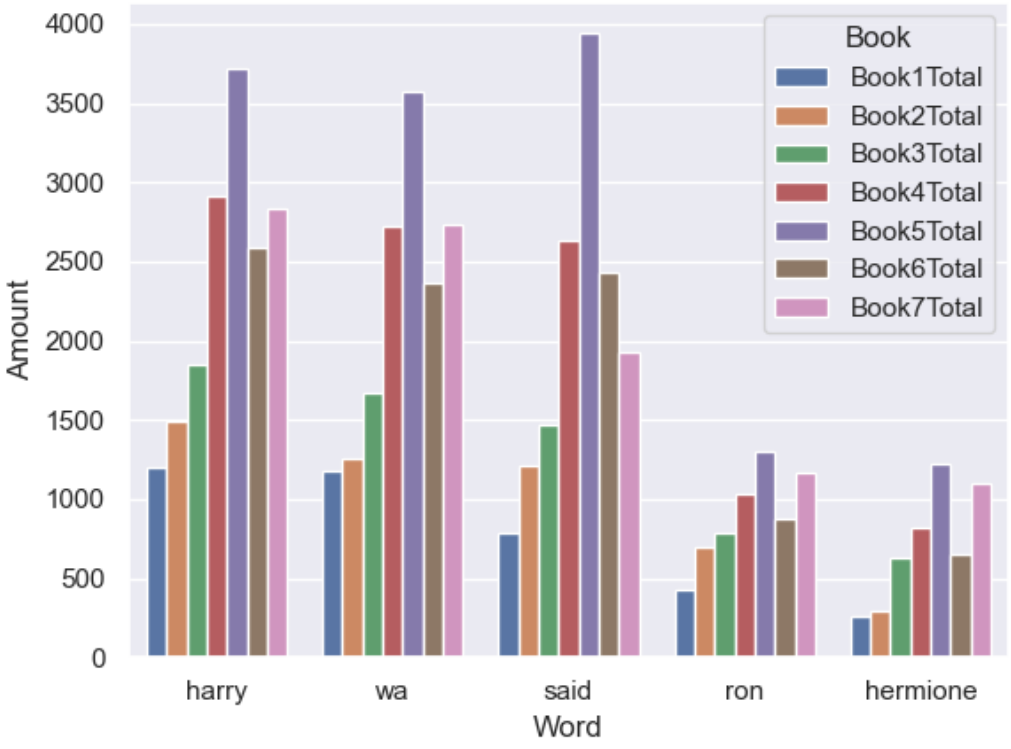

Figure 4.20: Seaborn Grouped Bar Plot, x-axis: five most used words on the series grouped by each book, y-axis: word count

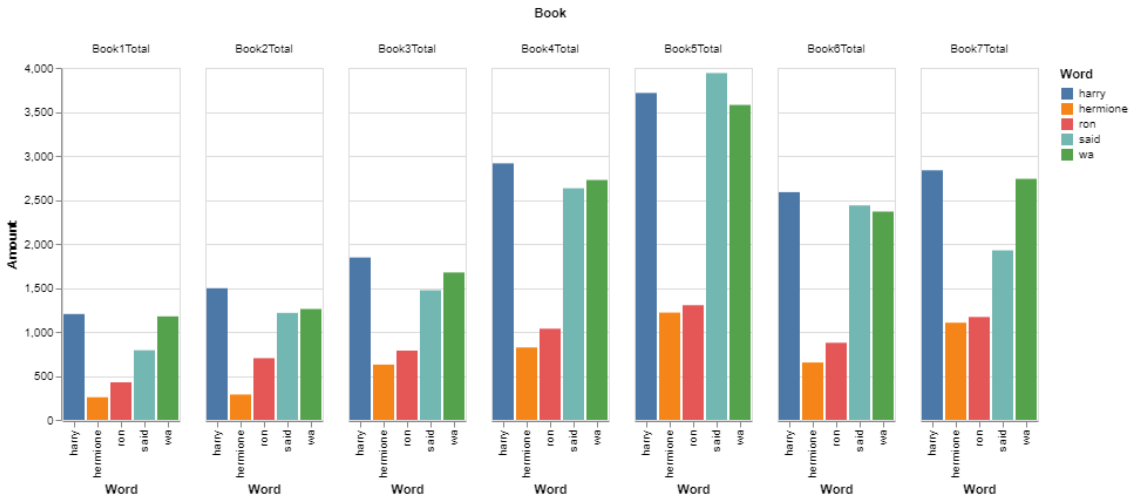

Figure 4.21: Vega-Altair Grouped Bar Plot, x-axis: five most used words on the series grouped by each book, y-axis: word count

4.1.8 Stacked Bar Plot

For the stacked bar plot visualization, we conducted tests using two libraries: Plotly (Figure 4.22) and Vega-Altair (Figure 4.23). To thoroughly evaluate their performance, we considered several key criteria. Below is the table (Table 4.5 displaying the test results along with the generated images. The criteria we focused on during the evaluation were:

- Execution Time
- Execution Time while Generating Image
- Overall Design
- Interactivity

Stacked Bar Plot Results				
Library	Execution Time	Execution Time with Image Generation	Design Ranking	Interactive
Plotly	1.277626s	1.670982s	1	Yes
Vega-Altair	0.0121450s	0.3877755s	2	

Table 4.5: Stacked Bar Plot Results

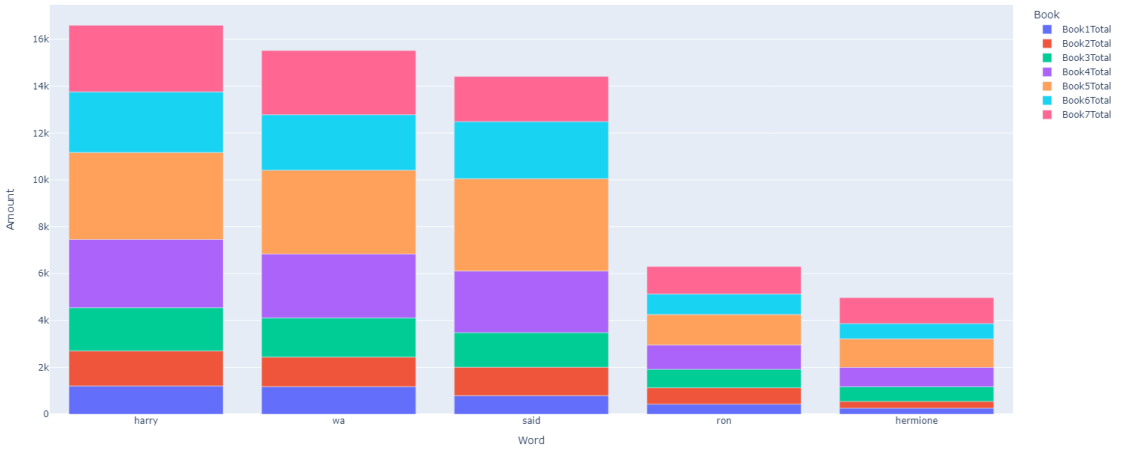

Figure 4.22: Plotly Grouped Bar Plot, x-axis: five most used words on the series stacked by each book, y-axis: word count

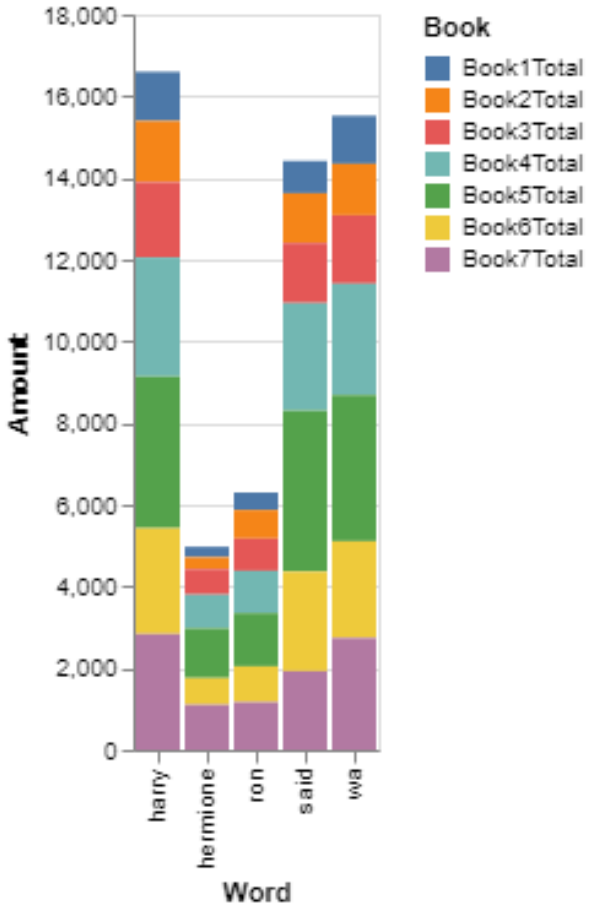

Figure 4.23: Vega-Altair Stacked Bar Plot, x-axis: five most used words on the series stacked by each book, y-axis: word count

4.1.9 Scatter Plot

The scatter plot is a visualization type available in many libraries. We tested five libraries for this visualization: Bokeh (Figure 4.25), Matplotlib (Figure 4.26), Plotly (Figure 4.24), Seaborn (Figure 4.28), and Vega-Altair (Figure 4.27). To evaluate their performance, we considered four key criteria:

- Execution Time
- Execution Time while Generating Image
- Overall Design
- Interactivity

All the images generated by each library are displayed below, accompanied by a table (Table 4.6) that includes the results of the tests conducted. This comparison will provide a comprehensive view of how each library performs in creating scatter plots.

Scatter Plot Results				
Library	Execution Time	Execution Time with Image Generation	Design Ranking	Interactive
Bokeh	0.0256974s	0.3345223s	1	
Matplotlib	0.2596002s	0.5012722s	2	
Plotly	1.6392681s	2.0638875s	1	Yes
Seaborn	0.2056944s	0.3992934s	2	
Vega-Altair	0.0377448s	0.4353310s	2	Yes

Table 4.6: Scatter Plot Results

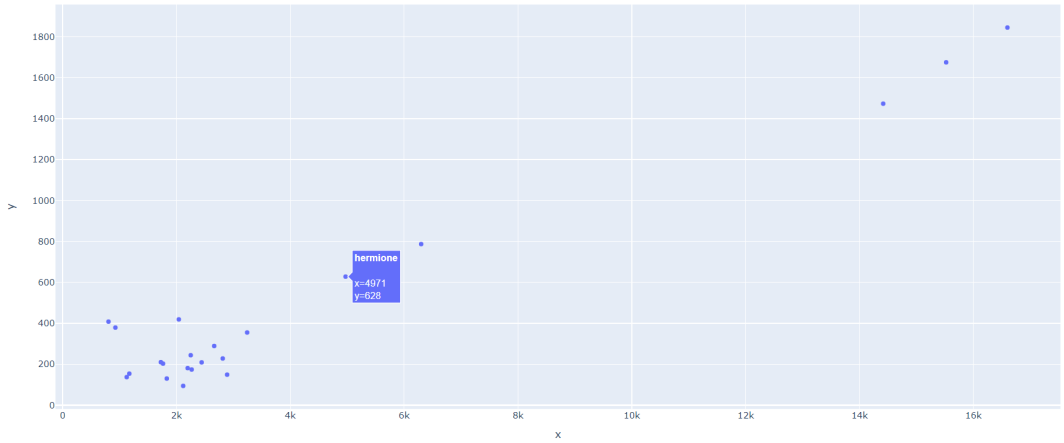

Figure 4.24: Plotly Scatter Plot, x-axis: Amount of most frequent words on series, y-axis: amount of most frequent words on book 3

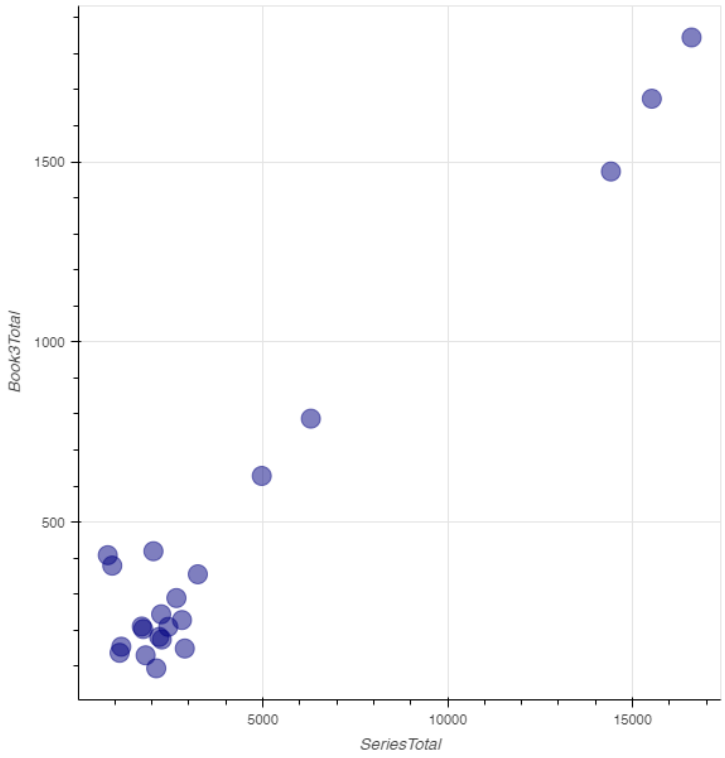

Figure 4.25: Bokeh Scatter Plot, x-axis: Amount of most frequent words on series, y-axis: amount of most frequent words on book 3

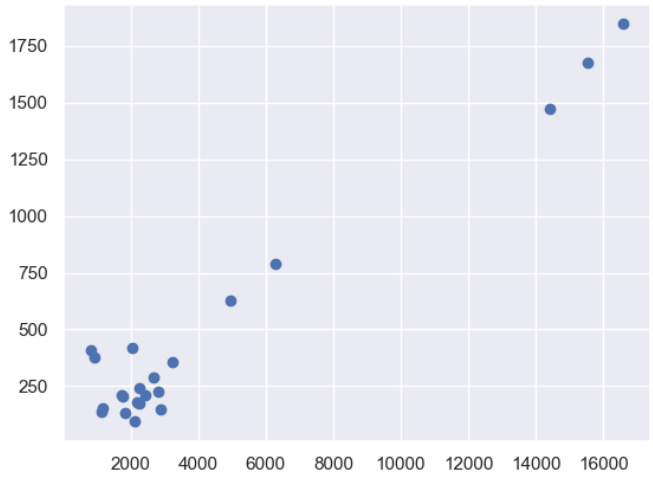

Figure 4.26: Matplotlib Scatter Plot, x-axis: Amount of most frequent words on series, y-axis: amount of most frequent words on book 3

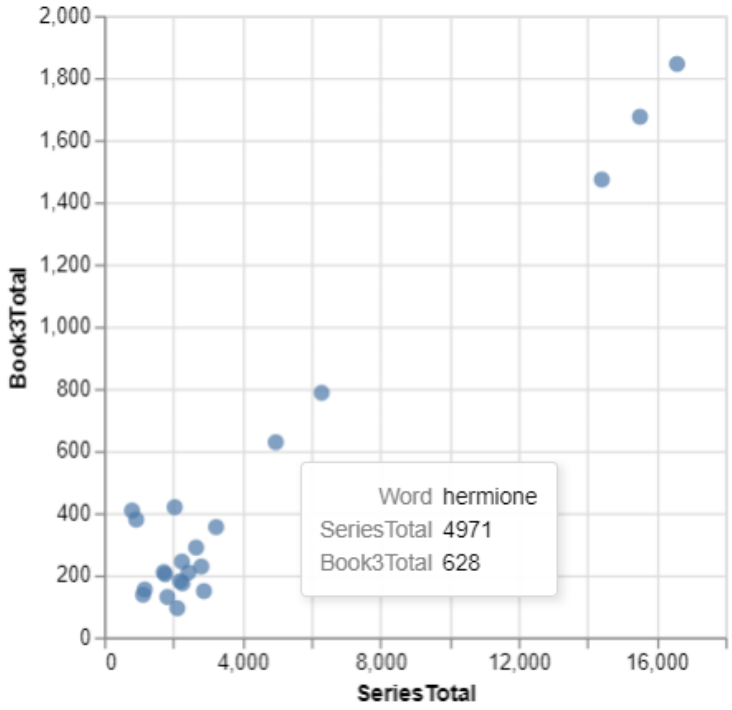

Figure 4.27: Vega-Altair Scatter Plot, x-axis: Amount of most frequent words on series, y-axis: amount of most frequent words on book 3

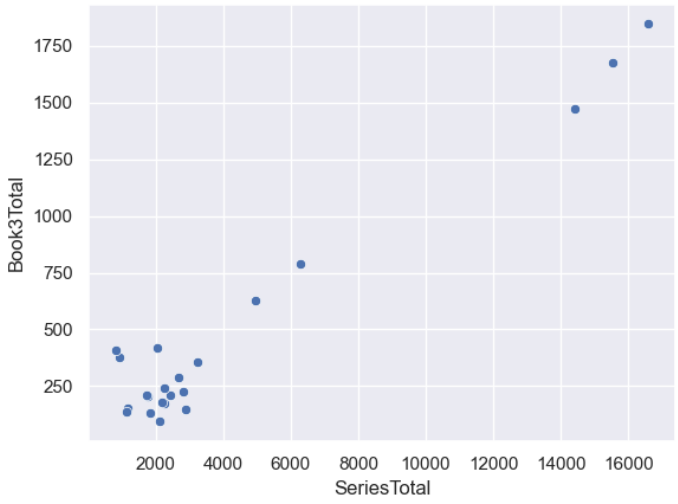

Figure 4.28: Seaborn Scatter Plot, x-axis: Amount of most frequent words on series, y-axis: amount of most frequent words on book 3

4.1.10 Data Quality

This type of visualization focused less on the textual data itself and more on the quality of the data file. To assess data quality, we used the library missingno, which helps users visualize gaps or issues in their datasets and understand why a particular visualization might not be successful.

For testing this library, we used a different dataset that contained numerous missing rows and even entire columns without content. This dataset consists of posts removed from the X social media platform, providing a useful example (Figure 4.29) for examining how missing data can impact visualization outcomes.

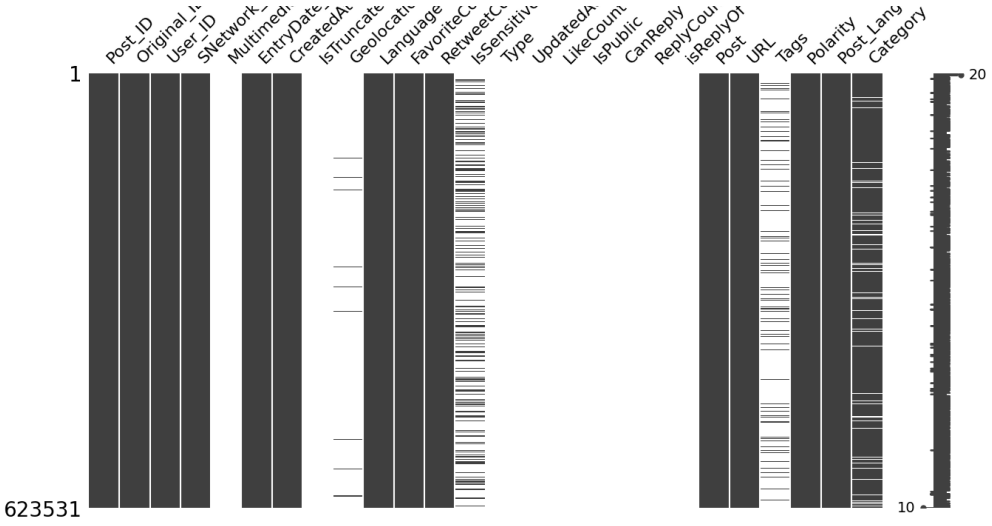

Figure 4.29: Missigno Data Quality

4.2 Final Program

Initially, our goal was to design the program so that when a user selected a specific type of visualization, the system would automatically recommend and select the most suitable library for that visualization. However, as the project evolved, we decided to give users greater flexibility by allowing them to choose the library themselves. This approach empowers users to prioritize based on their specific needs—whether they are focused on aspects like execution speed or interactivity. By offering this choice, we provide users with more control over their experience, enabling them to tailor the visualization process to their own preferences and requirements.

As a result, the final program offers ten different types of visualizations. Within each type, the number of available library options ranges from one to five, depending on the specific visualization. In some cases, only one library provides the necessary support for

a particular type of visualization, while in others, several libraries are available, offering users a wider selection. This ensures that the program is both comprehensive and flexible, catering to various visualization needs with varying levels of support across libraries.

Although we initially planned to test eight libraries, we ultimately decided not to use Plotnine and Pygal. The types of visualizations these libraries offered were already well-covered by others, such as Bokeh, Matplotlib, Plotly, Seaborn, and Vega-Altair. As a result, we chose to focus on the libraries that provided the most diverse and comprehensive visualization capabilities, streamlining our efforts without sacrificing functionality.

This section introduces the program by outlining its workflow, with a supporting flowchart for better clarity. Each step in the process is explained in detail to give a comprehensive breakdown of the development and functionality. The program is available for public use both as a Python package, accessible at TextDataVisualizationGV, and on GitHub at Text Data Visualization, offering users multiple ways to implement and explore the tool.

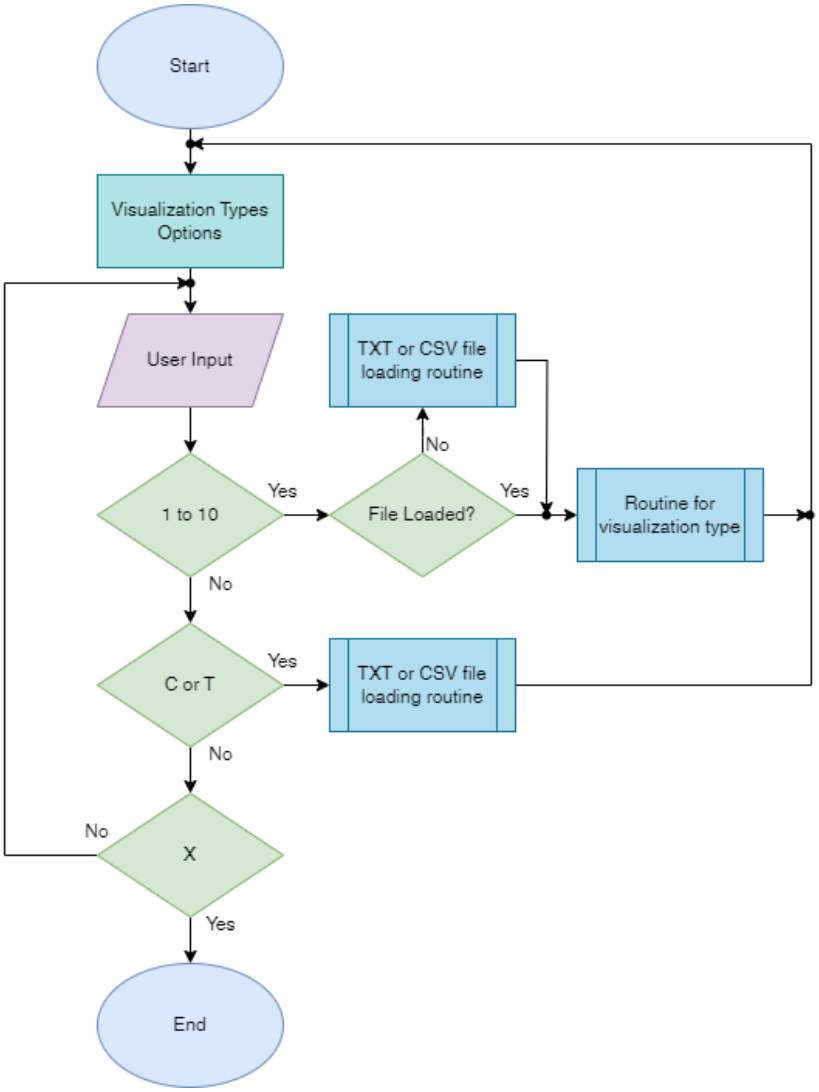

Figure 4.30: Program Flowchart

Start

The program begins with a brief introduction (Figure4.31) , explaining that there are ten different types of visualizations available. It highlights that only the word cloud visualization requires a TXT file, while all other types need a CSV file. Additionally, the program reminds users to save any images they wish to keep before generating new ones, ensuring no visuals are lost unintentionally.

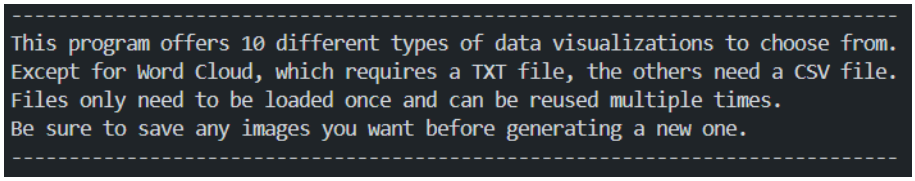

Figure 4.31: Program Introduction

After this introduction, the menu for selecting visualization (Figure 4.32) types is displayed. Initially, the 'C' and 'T' options are hidden and will only appear once the CSV or TXT file has been successfully loaded (Figure 4.33). The available options are:

1 - Word Cloud

2 - Bar Plot

3 - Pie Chart

4 - Bubble Chart

5 - Line Chart

6 - Multiple Lines Chart

7 - Grouped Bar Plot

8 - Stacked Bar Plot

9 - Scatter Plot

10 - Data Quality

C - Load new CSV File

T - Load new TXT File

X - End the Program

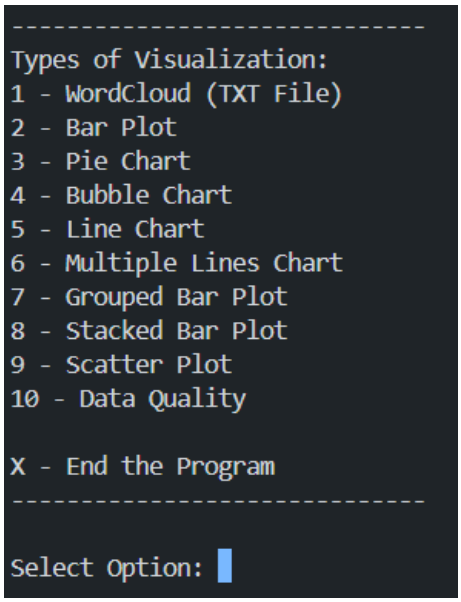

Figure 4.32: Visualization selection before files are loaded.

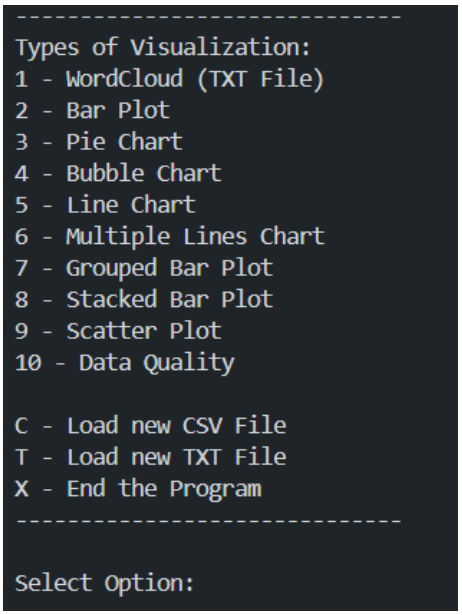

Figure 4.33: Visualization selection after files are loaded.

User Input

Once the program is introduced and the options are presented, the user must select from three main actions: choosing a visualization type, loading new files, or exiting the program. To choose a visualization, the user enters a number from 1 to 10, corresponding to the available types. To load new files, they input 'C' for a CSV file or 'T' for a TXT file. If they wish to exit, they simply enter 'X'. If the user inputs anything other than these valid options, the program will notify them that their choice is invalid and prompt them to try again. This section outlines the actions that occur based on the user's selection. Whether they choose a visualization type, load new files, or exit the program, each option triggers a specific response, guiding the user through the next steps.

Visualization Choice

If the user inputs a number between 1 and 10, different routines are activated for each visualization type. Below is a brief explanation of what happens for each option:

- 1 - Word Cloud:

  Once the user selects the Word Cloud (Figure 4.34) option, the program first asks whether they want to use an image to shape the words. If they choose to do so, the program explains that the image must be in JPG format, featuring a black design on a white background.

Following this selection, users are presented with the option to change the color palette. If they opt to customize the colors, they can choose from eight different palettes. After making these selections, the program proceeds to generate the word cloud image.

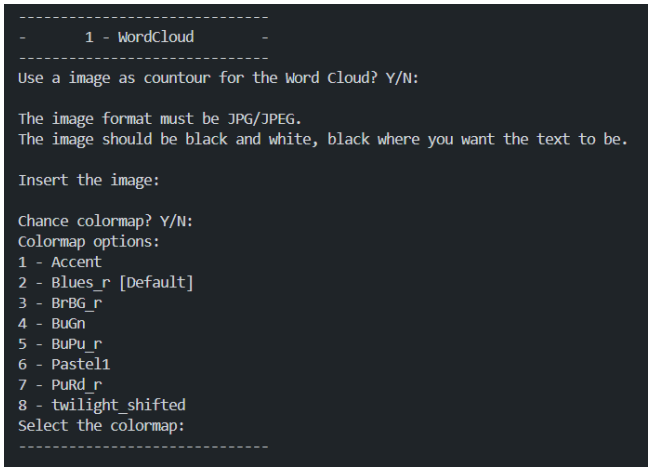

Figure 4.34: Program Word Cloud Routine

- 2 - Bar Plot:

The Bar Plot (Figure 4.35) is the first visualization option that allows users to select their preferred library for generating the plot. The available libraries are Bokeh, Matplotlib, Plotly, Seaborn, and Vega-Altair. Once the user chooses a library, the program prompts them to specify the x-axis, y-axis, and the number of rows they wish to visualize. After these inputs are provided, the program generates the bar plot image.

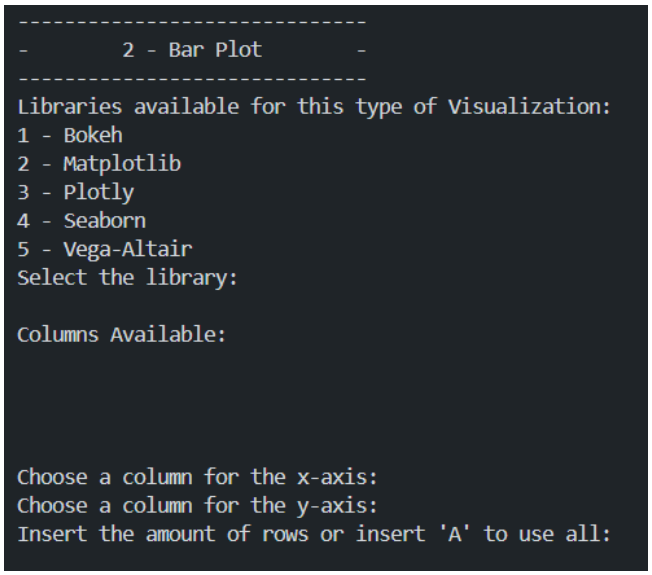

Figure 4.35: Program Bar Plot Routine

- 3 - Pie Chart:

  The Pie Chart (Figure 4.36) option allows users to choose from four different libraries: Bokeh, Matplotlib, Plotly, and Vega-Altair. After selecting a library, the program prompts the user to provide labels for each slice of the pie chart, along with the corresponding values that determine the size of each slice. Users are then asked to specify the number of rows they wish to visualize. Once all the inputs are provided, the program generates the pie chart image.

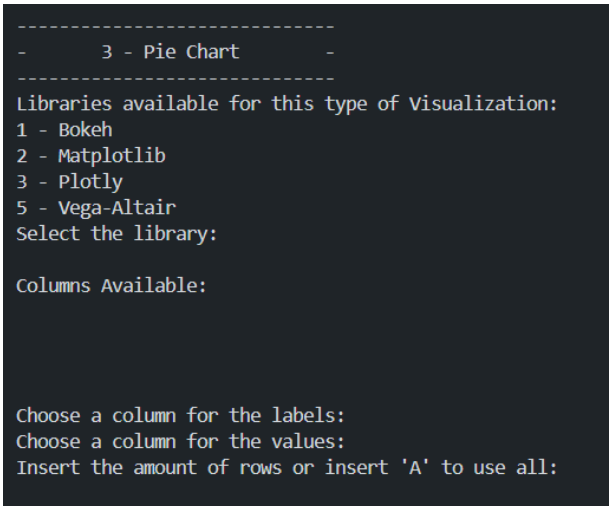

Figure 4.36: Program Pie Chart Routine

- 4 - Bubble Chart:

  The Bubble Chart (Figure 4.37) option does not allow for library selection, as it exclusively uses Plotly. Users begin by specifying the x-axis, y-axis, and the size of the bubbles. Following this, they have two additional choices: whether to name the bubbles and whether to group them. Once these selections are made, the program generates the bubble chart image.

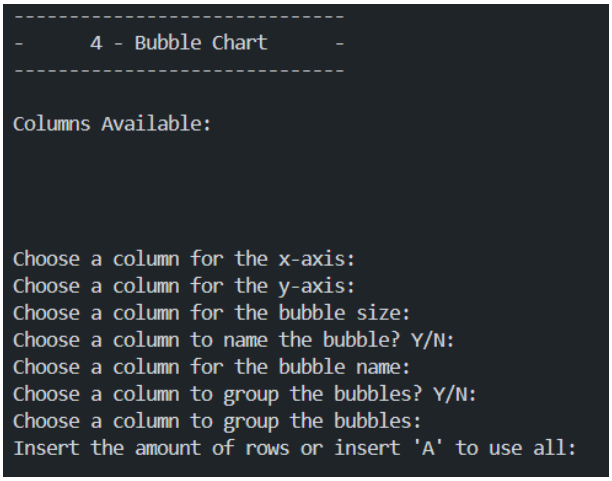

Figure 4.37: Program Bubble Chart Routine

- 5 - Line Chart

  As one of the most common types of data visualization, the Line Chart (Figure 4.38) offers five library options: Bokeh, Matplotlib, Plotly, Seaborn, and Vega-Altair. After selecting a library, the user is prompted to input the x-axis and y-axis values, followed by the number of rows they wish to visualize. Once these inputs are provided, the program generates the line chart image.

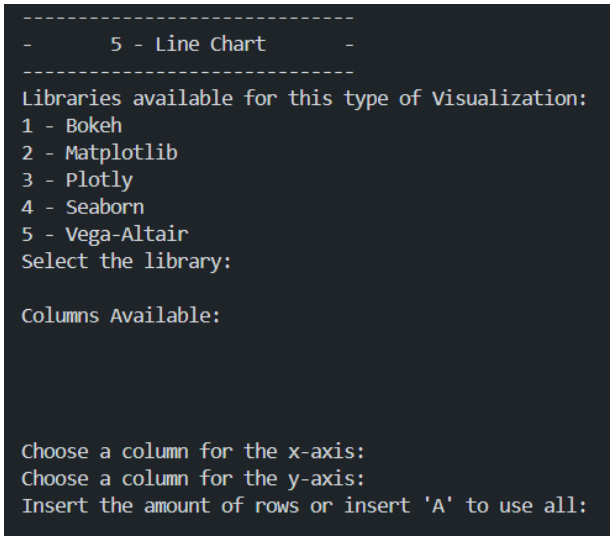

Figure 4.38: Program Line Chart Routine

- 6 - Multiple Lines Chart

  The Multiple Line Chart (Figure 4.39) provides two library options: Bokeh and Plotly. Unlike the standard Line Chart, where multiple lines can be added if the rows contain the same elements on the x-axis, this visualization allows users to create multiple lines by selecting additional columns directly.

  Therefore for the Multiple Line Chart, after selecting the x-axis, the program asks how many new lines (columns) the user would like to add. The user then inputs each column separately. Finally, they specify the number of rows to visualize. Once all selections are made, the program generates the multiple line chart image.

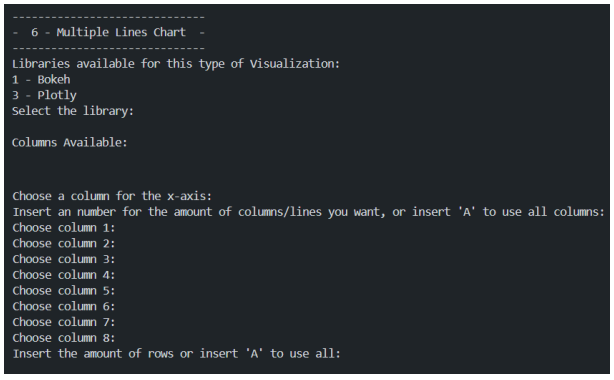

Figure 4.39: Program Multiple Line Chart Routine

- 7 - Grouped Bar Plot

  The Grouped Bar Plot (Figure 4.40) provides users with two library options: Seaborn and Vega-Altair. Once a library is chosen, the program guides the user through the process of setting up the visualization. First, it asks for the x-axis and y-axis values. Then, the user is prompted to select a column that will be used to group the data, ensuring that the values are properly categorized. Finally, the program requests the number of rows the user wants to visualize. After all inputs are entered, the grouped bar plot image is generated.

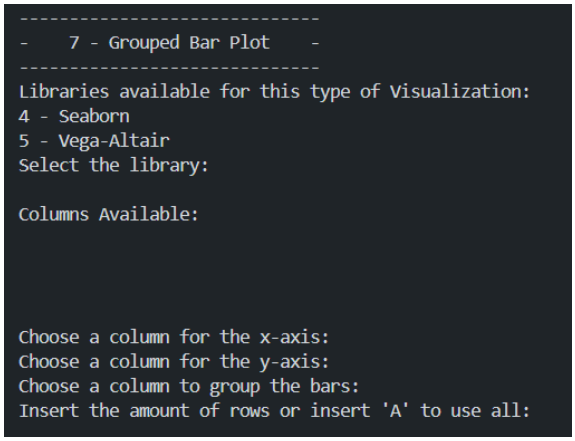

Figure 4.40: Program Grouped Bar Plot Routine

- 8 - Stacked Bar Plot

  The Stacked Bar Plot (Figure 4.41) offers two library options: Plotly and Vega-Altair. After selecting a library, the user is prompted to choose two columns—one for the x-axis and one for the y-axis. Following this, they need to select a column that will be used to stack the bars. Lastly, the user specifies the number of rows they want to visualize. Once all inputs are provided, the program generates the stacked bar plot image.

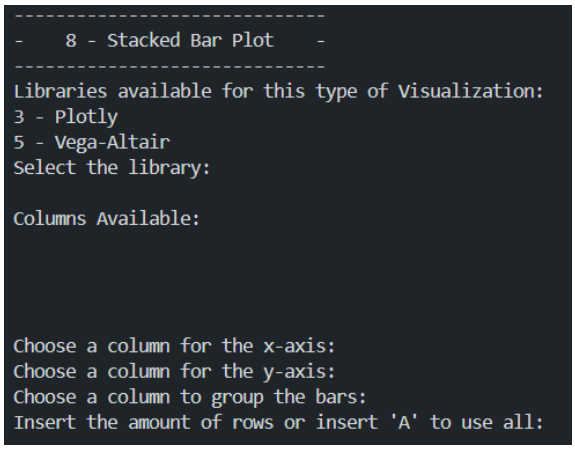

Figure 4.41: Program Stacked Bar Plot Routine

- 9 - Scatter Plot

  The Scatter Plot (Figure 4.42) is a widely used visualization, which allowed for five library options: Bokeh, Matplotlib, Plotly, Seaborn, and Vega-Altair. After selecting a library, the user simply needs to choose the columns for the x-axis and y-axis, as well as specify the number of rows they wish to visualize. Once these selections are made, the program generates the scatter plot image.

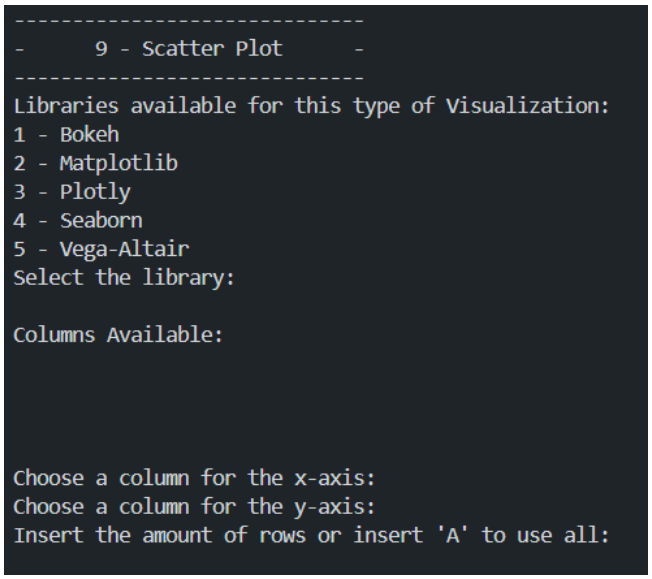

Figure 4.42: Program Scatter Plot Routine

- 10 - Data Quality

  This visualization type Data Quality (Figure 4.44), as previously mentioned, focuses on the structure of the data file itself, specifically highlighting missing information. Unlike other visualizations, it does not prompt the user to select specific columns, as it automatically displays all columns in the dataset. Users are given the option to sample the first 500 rows for a quick preview. After this selection, the program generates the visualization, showcasing the missing data points.

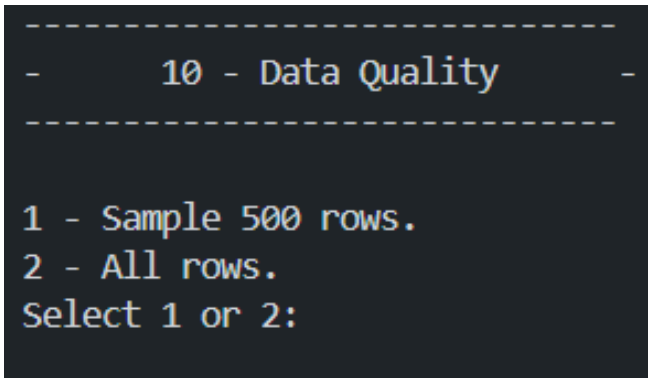

Figure 4.43: Program Data Quality Routine

- C - Load new CSV File

  This option is only shown after the program has successfully loaded a CSV file. If the user selects a visualization type but the required file has not yet been loaded, the program will automatically prompt them to enter the routine for loading the file. Choosing this option also allows the user to load a new CSV file. When loading a CSV file, the user is asked to specify the file's delimiter and select the type of encoding being used.

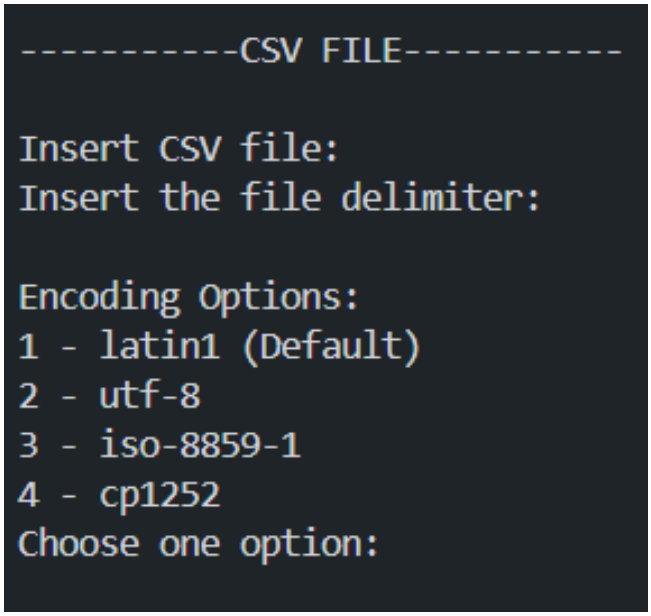

Figure 4.44: Program CSV File Reload Option

- T - Load new TXT File

  This option also only appears after the user has successfully loaded the TXT file at least once. Choosing this option allows the user to load a new TXT file. Unlike the CSV file, there are no additional actions required after the user inputs the TXT file.

- X - End the Program The final option allows the user to exit the program by simply entering 'X'. This will terminate the program and close the session.

4.3 Massive Text Dataset Results

As discussed earlier, the primary objective of this thesis is to develop a robust solution for text data visualization, addressing the growing need to efficiently analyze the vast amounts of textual data generated daily. Given the exponential increase in data creation, it is critical to have tools that can visualize large datasets in a clear and meaningful way. Although the tool we developed is primarily designed to work with pre-processed and processed

datasets, we conducted a practical test using larger datasets to evaluate its performance and versatility.

For this evaluation, we selected two datasets related to Spotify, which provided a rich source of textual and numerical data for visualization. To explore the tool's capabilities, we applied six different types of visualizations: Bar Plot, Pie Chart, Scatter Plot, Stacked Bar Plot, Multiple Line Chart, and Data Quality.

Through this test, we aimed to assess not only the tool's ability to create insightful visualizations but also its performance when handling large-scale data. By applying these varied visualization techniques, we gained a clearer understanding of how the tool operates in practice, identifying areas where it excelled as well as potential areas for further enhancement.

The first task was to create Data Quality (Figures 4.45 and **??**) visualizations for each CSV file, assessing the completeness and integrity of the data. We also documented all available columns (Figures 4.47 and 4.48) for each dataset, ensuring we understood the full scope of the data provided for analysis.

For the first dataset, Soptify Songs with Timesptamps and Lyrics, we generated a Bar Plot (Figure 4.49) and a Pie Chart (Figure 4.50) to evaluate the program's ability to handle simple visualizations with large datasets. For the second, smaller dataset, titled Spotify Songs with Attributes and Lyrics, we tested additional visualizations, including a Multiple Line Chart (Figure 4.51), Scatter Plot (Figure 4.52), and Stacked Bar Plot (Figure 4.53), allowing us to explore six of the ten visualization types integrated into the program.

Given the importance of execution time for this test, we employed the Python TimeIt tool to accurately measure the execution time for each step. The timing results for the various operations are presented in the table (Table **??**) below, offering a clear comparison of the program's performance across different visualization tasks.

Massive Dataset Test Results		
Action	Dataset	Execution Time
Load file	Dataset 1	39.0933365s
Load file	Dataset 2	20.9813086s
Data Quality Visualization	Dataset 1	29.7249814s
Data Quality Visualization	Dataset 2	2.2431138s
Bar Plot Visualization	Dataset 1	0.5114219s
Pie Chart Visualization	Dataset 1	0.5114219s
Multiple Line Chart Visualization	Dataset 2	0.9256321s
Scatter Plot Visualization	Dataset 2	0.6712983s
Stacked Bar Plot Visualization	Dataset 2	1.0234129s

Table 4.7: Massive Dataset Test Results

Although the program successfully handled large datasets in terms of data processing, the visualizations generated were far from optimal. The sheer volume of information made it difficult for the charts to clearly and concisely display the data. Many of the visualizations

struggled to effectively communicate the insights from the data, as shown in the images below. This highlights the need for further optimization in the program's handling of large datasets, particularly regarding clarity and information representation.

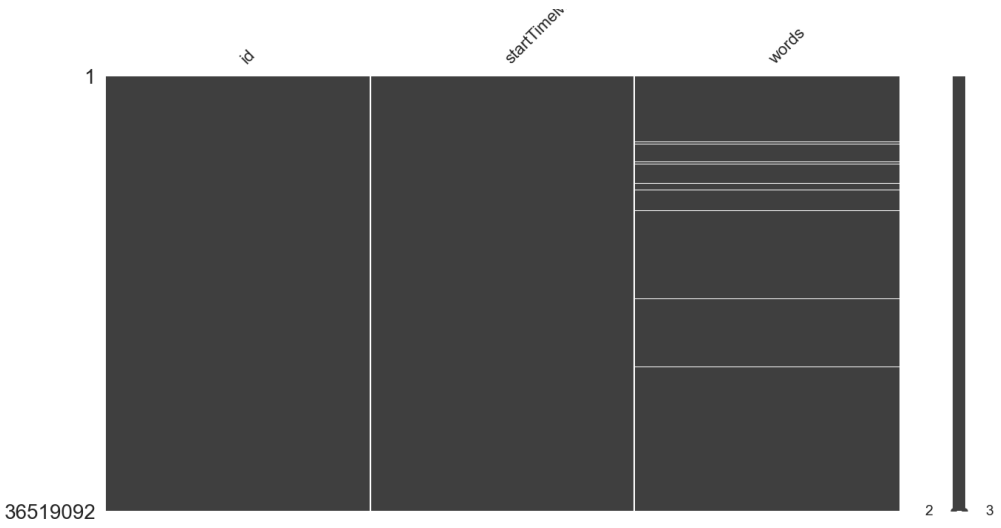

Figure 4.45: Data Quality for Spotify dataset 1, songs with lyrics and timestamps.

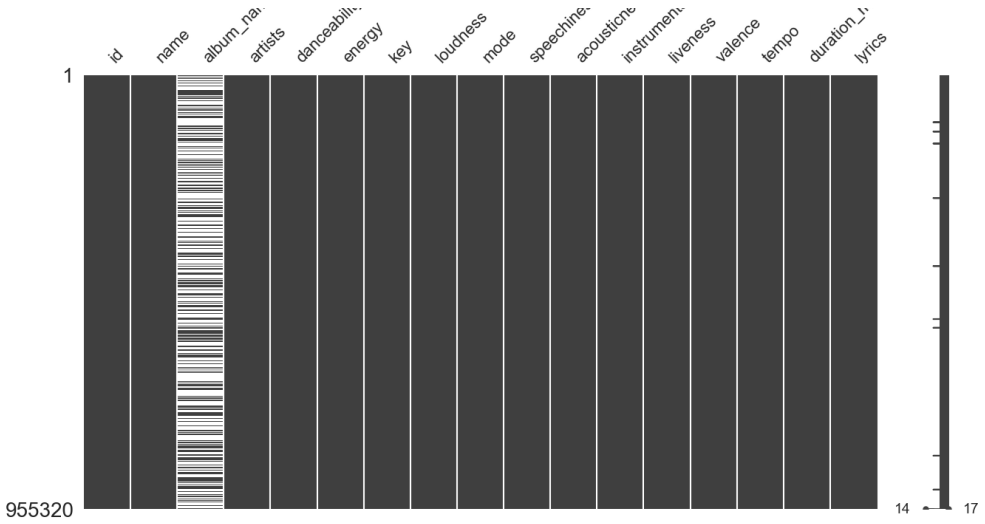

Figure 4.46: Data Quality for Spotify dataset 2, songs with attributes and lyrics.

```
Columns Available:
1 - id
2 - startTimeMs
3 - words
```

Figure 4.47: Columns available for Spotify dataset 1.

```
Columns Available:
1 - id
2 - name
3 - album_name
4 - artists
5 - danceability
6 - energy
7 - key
8 - loudness
9 - mode
10 - speechiness
11 - acousticness
12 - instrumentalness
13 - liveness
14 - valence
15 - tempo
16 - duration_ms
17 - lyrics
```

Figure 4.48: Columns available for Spotify dataset 2.

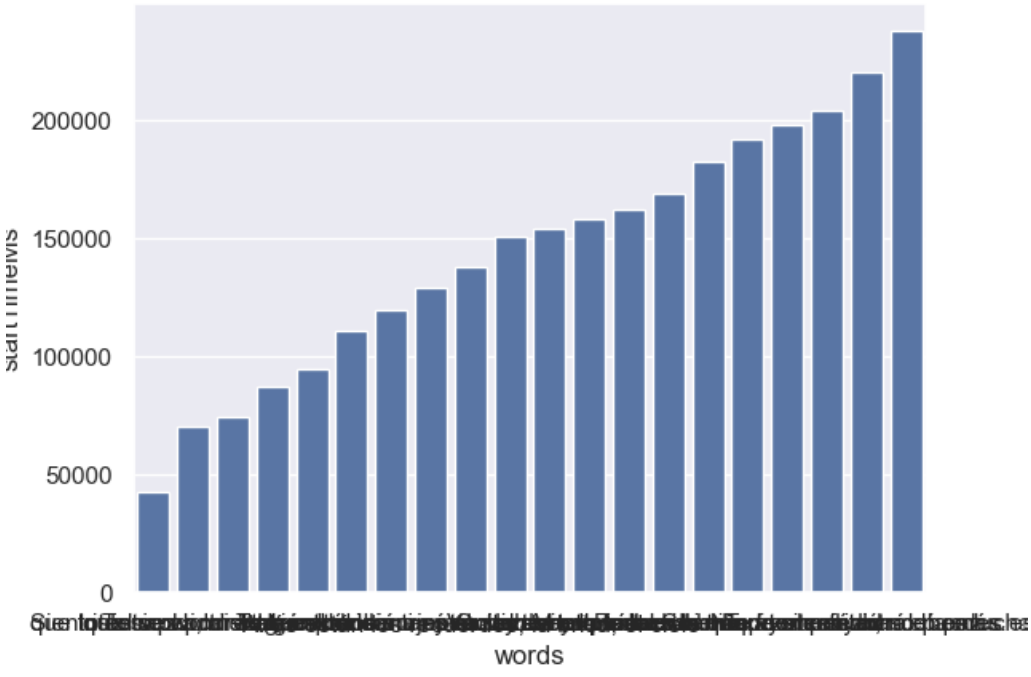

Figure 4.49: Barplot for Spotify dataset 1, songs with lyrics and timestamps.

Figure 4.50: Pie Chart for Spotify dataset 1, songs with lyrics and timestamps.

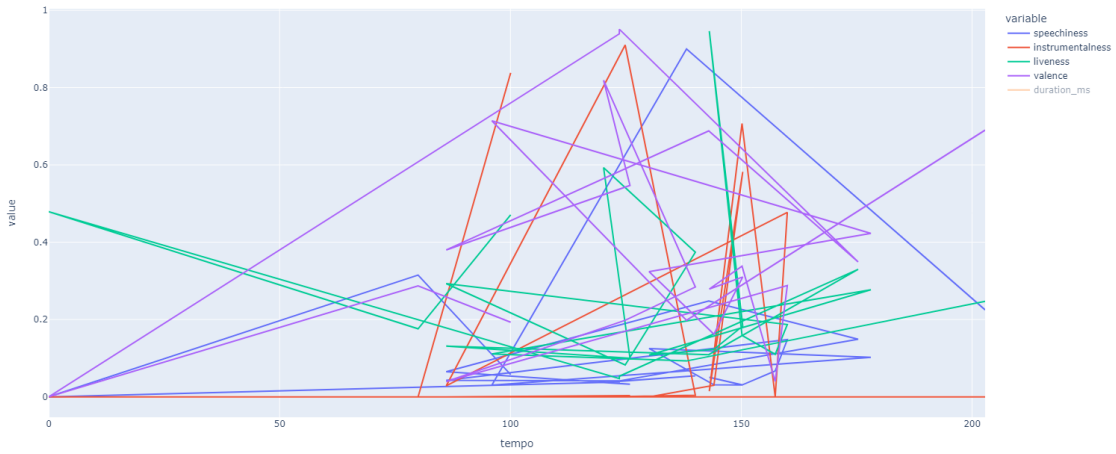

Figure 4.51: Multiple Line Chart for Spotify dataset 2, songs with attributes and lyrics.

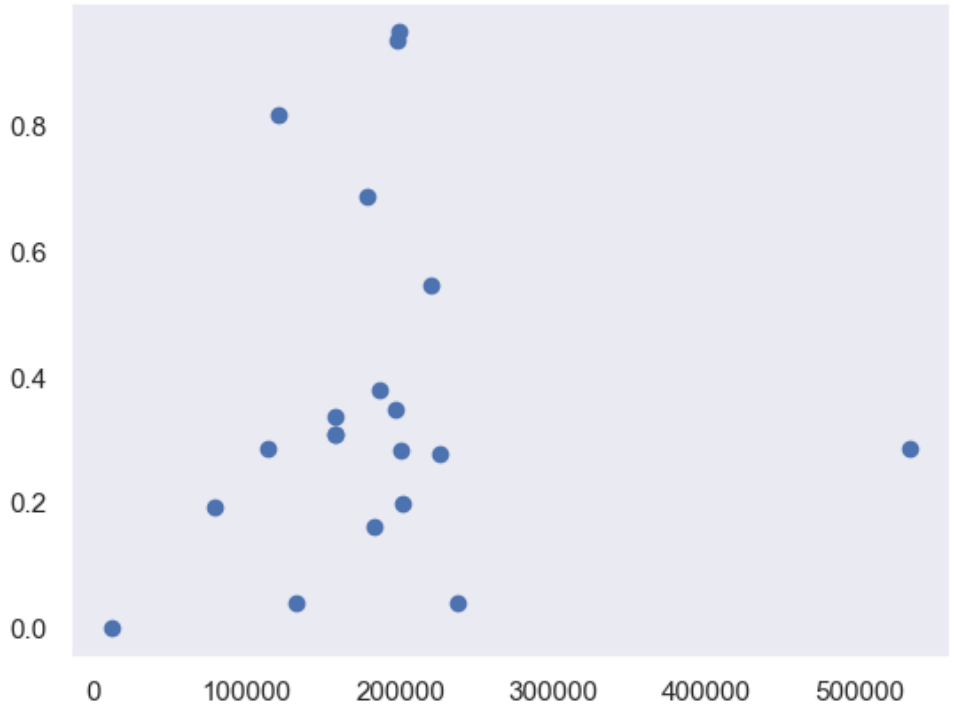

Figure 4.52: Scatter Plot for Spotify dataset 2, songs with attributes and lyrics.

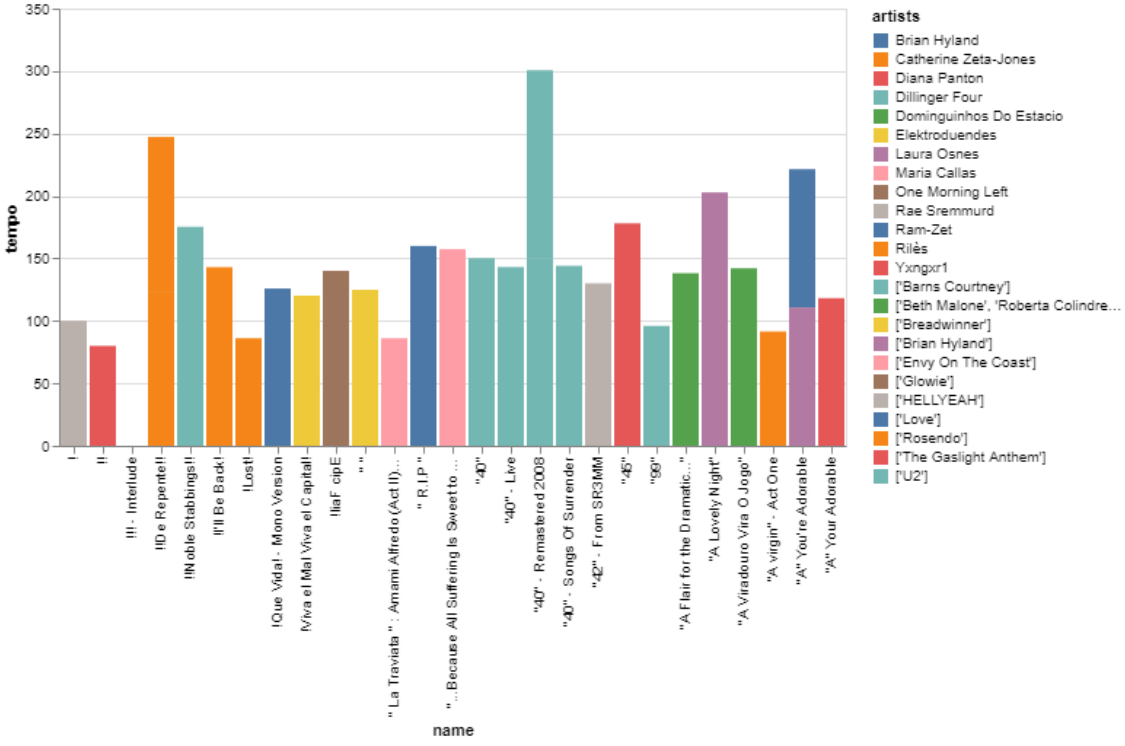

Figure 4.53: Stacked Bar Plot for Spotify dataset 2, songs with attributes and lyrics.

Chapter 5

Discussion

This chapter outlines the key observations made based on the results presented in the previous chapter. We reflect on the program's performance, the effectiveness of the libraries used for visualizations. These insights help guide potential improvements and future work.

The libraries with most visualization types, therefore, more versatile, were Plotly and Vega-Altair. Execution Time, Vega-Altair was the best overall. However execution time with image genaration both Vega-Altair and Seaborn were the best. In design, plotly was by far the best one, offering the most information for each visualization. And Bokeh was the only library that show limitations of any kind.

In evaluating the libraries based on their versatility, performance, and design capabilities, Plotly and Vega-Altair emerged as the most versatile options, offering a broader range of visualization types compared to the others. This versatility makes them more adaptable to various visualization needs, accommodating a wide array of use cases and data representation methods.

When examining execution time, particularly in relation to general performance, Vega-Altair stood out as the most efficient, delivering consistently faster results across different types of visualizations. This is a significant advantage for users dealing with large datasets, where processing speed becomes a critical factor. On the other hand, when considering execution time in relation to image generation, both Vega-Altair and Seaborn demonstrated superior performance. These libraries were able to generate visual outputs quickly without compromising quality, making them ideal for scenarios that demand real-time or near real-time visualizations.

From a design perspective, Plotly outshined the other libraries, offering the most detailed and informative visualizations. It provides a highly interactive user experience with features such as hover tooltips, zooming, and panning, which enhance the level of insight users can derive from the data. Its capacity to deliver visually rich and dynamic outputs makes it particularly well-suited for presentations or instances where a deeper exploration of the data is required.

However, it is important to note that Bokeh exhibited some limitations, both in terms of versatility and performance. Unlike Plotly or Vega-Altair, Bokeh struggled with certain types of visualizations and didn't offer the same level of interactivity or design sophistication. Its limitations were most apparent when handling complex datasets, where it failed

to match the speed and responsiveness of the other libraries. This highlights a key area for improvement or consideration for future work.

In summary, while Plotly and Vega-Altair stand out as the most comprehensive and high-performing libraries, each with its own strengths—Plotly for design and interactivity, and Vega-Altair for speed and versatility—Seaborn and Matplotlib also performed well in terms of efficient image generation.

Bokeh, however, revealed certain constraints, especially when handling more complex or larger datasets. These limitations, both in functionality and flexibility, became apparent when attempting to create highly detailed visualizations. While Bokeh still offers valuable tools for more basic or moderately complex visualizations, it was clear that for more sophisticated tasks requiring high levels of interactivity, detail, or speed, Plotly, Vega-Altair, Seaborn, and Matplotlib were preferable alternatives.

Ultimately, the findings from this analysis underscore the importance of selecting a library based on the specific requirements of the project. Whether the priority is speed, visual interactivity, design, or scalability, each library presents its own unique value proposition. For future iterations of the tool developed in this thesis, the possibility of including more flexible support for multiple libraries, allowing users to select according to their needs, remains a valuable direction to explore.

This comprehensive analysis helps to ensure that the final solution is robust and adaptable, meeting the diverse needs of future users in the field of text data visualization. This conclusion played a crucial role in our decision to develop the final program with multiple libraries integrated. By offering a variety of libraries, we ensure that users have the flexibility to select the tool that best suits their specific requirements, whether they prioritize speed, design, interactivity, or ease of use.

For instance, users who need faster processing times for larger datasets might prefer Vega-Altair, while those seeking highly interactive and visually appealing outputs can choose Plotly. Seaborn remains a solid choice for quick, static visualizations with minimal overhead, and even though Bokeh showed some limitations, it still provides value for simpler, more focused tasks.

In essence, the inclusion of multiple libraries allows the tool to cater to a wider range of use cases, enabling researchers, analysts, and developers to tailor their visualizations according to their needs. This flexibility increases the practical usability of the program, positioning it as a versatile and scalable solution in the field of text data visualization.

Considering the final test conducted, it became clear that the current program is not well-suited for handling large amounts of unprocessed data. The visualizations produced were not intuitive, and the amount of information displayed was limited and unclear. This limitation highlights an important area for improvement and points to a potential direction for future work.

A logical next step would be to develop a complementary solution that focuses on preprocessing and processing massive textual data before visualizing it. By implementing such a tool, the data could be cleaned, structured, and prepared in a way that enhances the effectiveness of the visualizations. This would not only improve the clarity and usability of the visual outputs but also expand the program's ability to manage and make sense of large-scale, raw text datasets.

Chapter 6

Conclusions and Future Work

As outlined in the introduction, this thesis had two primary goals: first, to gain a comprehensive understanding of massive text data visualization, and second, to develop a specialized solution dedicated solely to text data visualization.

To achieve the first objective, we conducted an extensive review of numerous academic papers, which provided us with a deeper understanding of the subject. Our analysis revealed several common characteristics across the studies. Most notably, a majority of the papers followed a consistent workflow, which included data acquisition, pre-processing, processing, and visualization stages. By focusing on the steps of pre-processing, processing, and visualization, we developed a categorization framework that helped us organize the papers, incorporating the specific objectives presented in each one.

Through this categorization, we identified that natural language processing (NLP) was the most widely used approach for processing textual data. Despite the diversity of research objectives—ranging from public sentiment analysis on COVID-19 vaccines to customer profiling for business purposes—we were able to classify these objectives into distinct categories. This classification highlights the expansive nature of the field, revealing multiple opportunities for further research. Textual data provides researchers with a rich foundation to explore various topics such as topic modeling, trend detection, sentiment analysis, geographic correlations, and business intelligence, among others.

The most crucial aspect of this research, however, was the visualization of text data. While the research objectives varied significantly, the types of visualizations employed by researchers were relatively consistent. Word clouds, graphs, and network diagrams were the most frequently used visualization methods, reflecting their versatility in highlighting patterns and relationships within textual data. Although many studies did not explicitly specify the tools used to create these visualizations, it was clear that different tools were often employed within the same research. This observation underlines the need for flexible, multi-purpose tools that can accommodate various types of visualizations.

For the second objective—developing a solution for text data visualization—we selected ten different visualization types, each of which had proven to be relevant during our literature review. These types ranged from Word Clouds to Grouped Bar Plots, providing users with a wide variety of options suited to different research needs. By offering multiple Python libraries for each visualization type, we ensured that users could choose based on their specific requirements, such as execution speed or interactivity. This flexibility allows users to adapt the tool to their unique research contexts, whether they require quick

results or more interactive, exploratory visualizations.

Additionally, we made the solution accessible by packaging it as a Python library to simplify dependency management, making it easier for users to integrate the tool into their workflows. The code was also made available on GitHub, encouraging collaboration and allowing users to customize and extend the tool as needed. This open-source approach promotes innovation and provides a foundation for further development in the field of text data visualization.

To evaluate the solution's performance, we tested it with a large dataset to observe how it handled substantial volumes of text data. While the solution was able to process the data, the results were not as informative as initially hoped, largely due to the necessity of thorough pre-processing and processing steps. This finding suggests that, while the tool is effective, its utility can be significantly enhanced by ensuring that the input data is properly prepared beforehand. As a result, a key area for future work would be to develop an additional tool or module that supports the pre-processing and processing stages, making it easier for users to prepare their data for visualization.

Another important avenue for future development is the creation of a user-friendly frontend interface. Currently, the tool requires users to have some level of programming experience, which may limit its accessibility for non-technical users. Developing a graphical interface would greatly enhance the tool's usability, enabling researchers and practitioners without programming expertise to benefit from its capabilities. This step would also help bridge the gap between technical complexity and practical application, expanding the tool's reach.

In conclusion, this thesis successfully achieved its dual objectives. It provided a comprehensive evaluation of existing research on text data visualization, identifying key trends, workflows, and challenges faced by researchers. In addition, it introduced a new tool that offers flexible, customizable options for visualizing textual data, contributing a valuable resource to the field. The work also highlighted areas for future improvement, such as enhancing pre-processing capabilities and developing a more accessible interface, laying the groundwork for continued advancement in text data visualization research and practice.

Bibliography

[1] G. Voltoline, S. a. Pais, B. Silva, and J. Cordeiro, "Understanding massive text visualization," in *Proceedings of the 2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ser. ASONAM '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 706–711. [Online]. Available: https://doi.org/10.1145/3625007.3627482 ix

[2] D. Stojanovski, I. Dimitrovski, and G. Madjarov, "Tweetviz: Twitter data visualization," *Proceedings of the data mining and data warehouses*, 2014. xvii, 11, 12, 16, 19, 23

[3] N. Garg and R. Rani, "Analysis and visualization of twitter data using k-means clustering," in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2017, pp. 670–675. xvii, 12, 15, 23

[4] C. Conner, J. Samuel, A. Kretinin, Y. Samuel, and L. Nadeau, "A picture for the words! textual visualization in big data analytics," *arXiv preprint arXiv:2005.07849*, 2020. xvii, 12

[5] K. Sahu, Y. Bai, and Y. Choi, "Supervised sentiment analysis of twitter handle of president trump with data visualization technique," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0640–0646. xvii, 12, 14, 16, 19, 23, 24

[6] R. Patil, N. Gada, and K. Gala, "Twitter data visualization and sentiment analysis of article 370," in *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*. IEEE, 2019, pp. 1–4. xvii, 13, 16, 18, 23

[7] A. Ashok, M. Guruprasad, C. Prakash, and S. Shylaja, "A machine learning approach for disease surveillance and visualization using twitter data," in *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*. IEEE, 2019, pp. 1–6. xvii, 16, 20, 23, 24

[8] A. Sechelea, T. Do Huu, E. Zimos, and N. Deligiannis, "Twitter data clustering and visualization," in *2016 23rd international conference on telecommunications (ICT)*. IEEE, 2016, pp. 1–5. xvii, 12, 18, 23

[9] J. Zhang, Y. Wang, M. Shi, X. Wang *et al.*, "Factors driving the popularity and virality of covid-19 vaccine discourse on twitter: text mining and data visualization study," *JMIR public health and surveillance*, vol. 7, no. 12, p. e32814, 2021. xvii, 17, 19, 23, 24

[10] B. Sijtsma, P. Qvarfordt, and F. Chen, "Tweetviz: Visualizing tweets for business intelligence," in *Proceedings of the 39th International ACM SIGIR conference on*

Research and Development in Information Retrieval, 2016, pp. 1153–1156. xvii, 15, 20, 23, 24

[11] Ž. Krstić, S. Seljan, and J. Zoroja, "Visualization of big data text analytics in financial industry: a case study of topic extraction for italian banks," *ENTRENOVA-ENTerprise REsearch InNOVAtion*, vol. 5, no. 1, pp. 35–43, 2019. 23

[12] K. C. K. Ven, A. N. K. Ying, N. Q. Jie, S. Y. Lun, S. L. C. Yuen, D. Handayani, N. Hamzah, M. Lubis, and T. Mantoro, "Depression identification through social media posts: Data preprocessing for data visualization of tweets," in *2021 IEEE 7th International Conference on Computing, Engineering and Design (ICCED)*. IEEE, 2021, pp. 1–6. 23

[13] A. Almjawel, S. Bayoumi, D. Alshehri, S. Alzahrani, and M. Alotaibi, "Sentiment analysis and visualization of amazon books' reviews," in *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2019, pp. 1–6. 23

[14] S. Sagiroglu and D. Sinanc, "Big data: A review," in *2013 international conference on collaboration technologies and systems (CTS)*. IEEE, 2013, pp. 42–47.

[15] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1825–1837, 2018.

[16] L. Ma and Y. Zhang, "Using word2vec to process big text data," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 2895–2897.

[17] D. Angus, S. Rintel, and J. Wiles, "Making sense of big text: a visual-first approach for analysing text data using leximancer and discursis," *International Journal of Social Research Methodology*, vol. 16, no. 3, pp. 261–267, 2013. 9, 17

[18] N. Cao and W. Cui, *Introduction to text visualization*. Springer, 2016, vol. 1. 9

[19] N. Bikakis, G. Papastefanatos, and O. Papaemmanouil, "Big data exploration, visualization and analytics," *Big Data Res*, vol. 18, no. 10.1016, 2019. 10