Sentiment Analysis of Mailing Lists

Harini Balaji

Supervisor: Dr Andrei Popescu

A report submitted in fulfilment of the requirements
for the degree of MSc in Data Analytics

in the

Department of Computer Science

September 10, 2024

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Harini Balaji

Signature: HB

Date: 10-09-2024

Abstract

This research examines the application of sentiment analysis and topic modelling within Python special interest group mailing lists, utilizing models such as VADER, BERT, DistilBERT, and Latent Dirichlet Allocation (LDA) for topic modelling. The study evaluates sentiment analysis under two approaches: guided analysis using predefined subtopics and an unsupervised method.

Simpler models like VADER provide efficient sentiment analysis but often miss the nuanced insights required in technical discussions. In contrast, advanced models like BERT and DistilBERT, though capable of capturing deeper context, require fine-tuning to be effective in specialized domains. The use of LDA to structure topics proved essential, with predefined subtopics significantly enhancing the relevance of sentiment analysis, particularly in complex and technical conversations. However, in the unsupervised approach, broader, less coherent topics emerged, affecting the consistency of results.

The methodologies developed offer practical frameworks for improving community engagement, supporting more informed decision-making within the Python development ecosystem, and fostering better-targeted discussions for future growth.

Contents

1 Introduction **1**

1.1 Aims and Objectives 1

1.2 Overview of the Report 1

2 Literature Survey **3**

2.1 Python Community 3

2.1.1 Overview of the Python Community 3

2.1.2 Role of Contributors in the Python Ecosystem 4

2.1.3 Communication and Collaboration within the Community 5

2.1.4 Challenges and Dynamics within the Python Community 5

2.2 Mailing Lists in Python Community 6

2.2.1 Communication and Collaboration through Mailing Lists 6

2.2.2 Governance and Knowledge Sharing via Mailing Lists 7

2.2.3 Challenges Associated with Mailing Lists 8

2.3 Sentiment Analysis and Opinion Mining 8

2.3.1 Overview of Sentiment Analysis 9

2.3.2 Approaches to Sentiment Analysis 10

2.3.3 Sentiment Analysis on Python Mailing Lists 13

2.3.4 Challenges in Sentiment Analysis 14

2.3.5 Future Directions 15

3 Experimental Design and Methodology **16**

3.1 Goals of the Experiments 16

3.1.1 Research Objectives 16

3.1.2 Hypotheses 16

3.2 Choice of Programming Language 17

3.2.1 Python 17

3.2.2 Related Libraries 17

3.3 Dataset 18

3.3.1 Source and Description 18

3.3.2 Data collection Methods 18

3.4 Data Preprocessing 18

3.4.1 Text Cleaning . . . 18
3.4.2 Tokenization and Normalization . . . 18

4 Experiments and Results 20
4.1 Experiment 1: Establishing a Baseline with TextBlob and Evaluating VADER, BERT, and DistilBERT . . . 20
4.1.1 Experiment Purpose . . . 20
4.1.2 Implementation . . . 20
4.1.3 Sentiment Analysis Results . . . 21
4.1.4 Discussions . . . 22
4.2 Experiment 2: Topic Modeling and Sentiment Analysis Using LDA with Predefined Subtopics . . . 23
4.2.1 Experiment Purpose . . . 23
4.2.2 Implementation . . . 23
4.2.3 Sentiment Analysis Results . . . 24
4.2.4 Discussions . . . 24
4.3 Experiment 3: Topic Modeling and Sentiment Analysis Using LDA Without Predefined Subtopics . . . 25
4.3.1 Experiment Purpose . . . 25
4.3.2 Implementation . . . 25
4.3.3 Sentiment Analysis Results . . . 26
4.3.4 Discussions . . . 28
4.4 Comparitive Analysis and Survey . . . 28
4.4.1 Rationale Behind Model Selection . . . 28
4.4.2 Summary of Experimental Results . . . 30
4.4.3 Discussion of Technique Performance . . . 31
4.4.4 Peer Review and Feedback . . . 32
4.4.5 Practical Applications and Benefits of Sentiment Analysis in Open-Source Development . . . 33

5 Project Management and Skill Development 35
5.1 Introduction . . . 35
5.2 Organizational Skills and Time Management . . . 35
5.3 Peer Review and Feedback . . . 36
5.4 Learning Outcomes . . . 36

6 Conclusions 37

Appendices 40

A RESULTS 41

List of Figures

2.1 Onion model of roles in OSS communities [16]. 4
2.2 Hierarchy of NLP Models, with BERT at the top, showcasing its advanced capabilities in understanding context. Source: [10] 11
2.3 An illustration of LDA, showing how it clusters words and documents by topics, revealing underlying themes that are crucial for sentiment analysis in large text datasets.[1]. 12

5.1 Gantt Chart 36

List of Tables

4.1 Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob) 20
4.2 Sentiment Analysis for Mobile Mailing List using VADER 21
4.3 Sentiment Analysis for Mobile Mailing List using BERT 21
4.4 Sentiment Analysis for Mobile Mailing List using DistilBERT 22
4.5 Sentiment Analysis across Subtopics 24
4.6 Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT . . . 27

A.1 Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob) 41
A.2 Sentiment Analysis for Education Mailing List using VADER 42
A.3 Sentiment Analysis for Education Mailing List using BERT 42
A.4 Sentiment Analysis for Education Mailing List using DistilBERT 43
A.5 Sentiment Analysis across Subtopics 43
A.6 Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT . . . 44
A.7 Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob) 45
A.8 Sentiment Analysis for Email Mailing List using VADER 45
A.9 Sentiment Analysis for Email Mailing List using BERT 46
A.10 Sentiment Analysis for Email Mailing List using DistilBERT 46
A.11 Sentiment Analysis across Subtopics 46
A.12 Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT . . . 47
A.13 Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob) 47
A.14 Sentiment Analysis for DB Mailing List using VADER 48
A.15 Sentiment Analysis for DB Mailing List using BERT 48
A.16 Sentiment Analysis for DB Mailing List using DistilBERT 48
A.17 Sentiment Analysis across Subtopics 48
A.18 Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT . . . 49

Chapter 1

Introduction

Sentiment analysis, also referred to as opinion mining, is an essential technique in natural language processing (NLP) that focuses on identifying and extracting subjective information from text. It plays a critical role in various fields, including customer service, marketing, and social media analysis, where understanding public sentiment is crucial for making informed decisions. In the context of this research, sentiment analysis is applied to technical discussions within the Python development community, particularly through mailing lists. These discussions provide a wealth of insights that can be leveraged to understand community sentiments on various topics, helping to improve engagement and drive development efforts.

1.1 Aims and Objectives

The primary aim of this research is to evaluate the performance and effectiveness of various sentiment analysis models when applied to technical discussions in Python development mailing lists. The study focuses on the following key objectives:

- **Model Evaluation:** Assess the accuracy and relevance of sentiment analysis models such as VADER, BERT, DistilBERT and LDA when applied to technical discussions.
- **Impact of Predefined Subtopics:** Explore the effects of using predefined subtopics versus allowing topics to emerge naturally from the data. This objective aims to determine whether structured guidance enhances the performance of sentiment analysis models.

This research not only contributes to the understanding of sentiment analysis in specialized technical domains but also offers practical tools and methodologies that can be adopted by other communities for similar analyses.

1.2 Overview of the Report

The project report is organised as follows:

- **Chapter 2: Literature Review**
  This chapter reviews existing literature relevant to the study, including an exploration of the Python community, the role of mailing lists, and the methodologies and challenges associated with sentiment analysis.

- **Chapter 3: Experimental Design and Methodology**
  This chapter details the research objectives, hypotheses, choice of programming language, and dataset collection. It explains the experimental setup, including data preprocessing.

- **Chapter 4: Experiments and Results**
  This chapter presents the results of the experiments, which include sentiment analysis using VADER, BERT, and DistilBERT, and topic modeling with LDA. It discusses the findings from both predefined and emergent subtopics and provides comparative analysis.

- **Chapter 5: Project Management and Skill Development**
  This chapter focuses on the project management techniques and organizational skills applied during this research, emphasizing how effective management contributed to the successful completion of the project. Additionally, it highlights the personal development and skills gained, which are valuable for future work.

- **Chapter 6: Conclusions**
  The final chapter summarizes the key findings, discusses their implications for the Python community, and suggests areas for future research to improve sentiment analysis in technical contexts.

Chapter 2

Literature Survey

2.1 Python Community

The Python community represents one of the most significant and dynamic forces in the open-source software (OSS) landscape. As a collective of developers, educators, and enthusiasts, the community has been instrumental in driving Python's growth into one of the most widely adopted programming languages globally. This section provides a detailed exploration of the Python community, examining its structure, roles, communication methods, and the challenges it faces as it continues to evolve.

2.1.1 Overview of the Python Community

Python's rise to prominence can be largely attributed to its robust and inclusive community. Over the years, Python has grown to become one of the most popular programming languages worldwide, used by a diverse group of users ranging from beginners to experts across various industries, including web development, data science, artificial intelligence, and more [14].

The language's simplicity, readability, and versatility have made it a favorite among developers and educators alike, contributing to its rapid adoption and widespread use. Central to Python's success is its open-source nature, which has allowed for community-driven development. Unlike proprietary software, Python's development is primarily driven by volunteers who contribute in various capacities. These contributions range from writing code and documentation to reporting bugs and participating in discussions about the language's future direction [7].

The Python Software Foundation (PSF), a non-profit organization, plays a crucial role in supporting these efforts by managing Python's intellectual property, funding community initiatives, and organizing events like PyCon that foster collaboration and knowledge sharing [6].

2.1.2 Role of Contributors in the Python Ecosystem

The Python community is characterized by a diverse range of contributors, each playing a vital role in the ecosystem. The trajectory of contributors within the community often begins with them as passive users, who primarily utilize Python for their projects. Over time, these users may evolve into more active contributors, taking on roles such as bug reporters, bug fixers, and eventually core developers and project leaders [16, 5].

In their seminal work, Ye and Kishida [16] identified several key roles within OSS communities like Python, including:

- **Passive Users**: Individuals who use the software but do not contribute to its development.
- **Bug Reporters**: Users who identify and report bugs but do not necessarily fix them.
- **Bug Fixers**: Contributors who actively fix bugs, requiring a deeper understanding of the codebase.
- **Peripheral Developers**: Those who contribute new features or enhancements sporadically.
- **Active Developers**: Regular contributors who are deeply involved in the development process.
- **Core Members**: Key contributors who guide and coordinate the development efforts.
- **Project Leaders**: Individuals who have a vision for the project and direct its overall development.

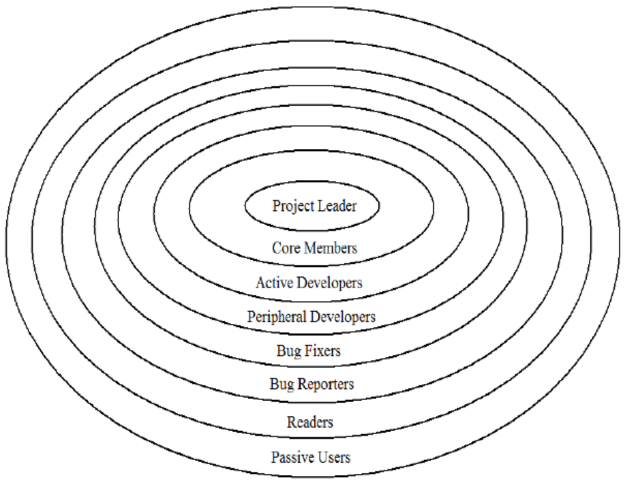


Figure 2.1: Onion model of roles in OSS communities [16].

The roles of core developers and project leaders are particularly crucial as they are responsible for maintaining the language's quality and ensuring that its development aligns with the community's goals. The concept of the ”Benevolent Dictator For Life” (BDFL), a role held by Python's creator Guido van Rossum until his retirement, was central to the community's governance structure. Van Rossum's influence on the language's development was profound, and his leadership helped shape Python into the language it is today [9].

Inclusivity and diversity are also critical aspects of the Python community. Efforts have been made to ensure that contributors from all backgrounds feel welcomed and can participate fully in the community. Initiatives such as codes of conduct, mentorship programs, and diversity grants at events like PyCon are designed to foster an inclusive environment where everyone has the opportunity to contribute [3].

2.1.3 Communication and Collaboration within the Community

Effective communication and collaboration are the lifeblood of the Python community, enabling it to function as a cohesive unit despite being globally dispersed. The community relies on a variety of tools and platforms to facilitate these interactions, ensuring that all members can contribute to the language's development and maintenance.

Mailing lists are among the most important communication tools in the Python community. The python-dev mailing list, in particular, serves as a central forum for discussing Python Enhancement Proposals (PEPs), which are critical for the language's evolution. These discussions are often highly technical, involving detailed debates about the implications of proposed changes. The archives of these mailing lists provide a rich repository of knowledge, documenting the decision-making processes that have shaped Python over the years [7].

In addition to mailing lists, the Python community also utilizes platforms like GitHub, where the Python codebase is hosted. GitHub allows contributors to submit pull requests, review code, and track issues, making it a central hub for development activities. The use of version control systems like Git ensures that all changes are meticulously documented, allowing for transparency and accountability in the development process [14].

Community events such as PyCon, local Python user group (PUG) meetings, and development sprints also play a significant role in fostering collaboration. These events provide opportunities for face-to-face interactions, allowing community members to build relationships, share ideas, and work together on projects. PyCon, in particular, is a global conference that brings together Python enthusiasts from around the world, offering a platform for networking, learning, and collaboration [6].

2.1.4 Challenges and Dynamics within the Python Community

As the Python community has grown, it has faced several challenges related to governance, communication, and inclusivity. One of the primary challenges is managing the decentralized nature of the community. While decentralization allows for flexibility and innovation, it can also lead to fragmentation and inconsistent decision-making. The PEP process, although effective, can sometimes be slow, as it relies on consensus from a diverse group of contributors with varying opinions and priorities [11].

The transition from a BDFL-led governance model to the current Steering Council model introduced additional challenges. After Guido van Rossum stepped down as BDFL, the community had to adapt to a more distributed leadership structure. The Steering Council, composed of elected core developers, now guides the development of Python, ensuring that the language continues to evolve in a manner that reflects the community's collective vision [9].

Another significant challenge is managing the volume of communication within the community. High-traffic communication channels like the python-dev mailing list can be overwhelming, with hundreds of messages exchanged daily. This information overload can deter participation and slow down decision-making processes. To mitigate this, the community has implemented strategies such as moderation, summarization of discussions, and the use of dedicated tools to filter and organize communication [7].

The social dynamics within the community also present challenges, particularly in terms of balancing influence among contributors. Highly active contributors, often referred to as "star contributors," can dominate discussions, potentially stifling diversity of thought. It is crucial for the community to encourage participation from a broader range of contributors to avoid groupthink and ensure that a variety of perspectives are considered in the decision-making process [12].

2.2 Mailing Lists in Python Community

Mailing lists have been a cornerstone of communication, collaboration, and governance within the Python community. These mailing lists cater to a broad range of domains, ensuring that Python continues to evolve in ways that meet the needs of its diverse user base. This section explores the specific mailing lists dedicated to Python's applications in education, email handling, mobile development, and database management, as well as the challenges associated with these platforms.

2.2.1 Communication and Collaboration through Mailing Lists

Python-dev and Python-ideas Mailing Lists (Brief Overview)

The `python-dev` and `python-ideas` mailing lists are central to Python's evolution. While `python-dev` facilitates detailed technical discussions and decision-making around Python Enhancement Proposals (PEPs), `python-ideas` offers an informal space for brainstorming. These mailing lists are vital for communication among core developers and contributors, driving Python's continuous development. However, specialized mailing lists also play a crucial role, reflecting the diverse applications of Python in various fields.

Python Education Mailing List

The Python Education mailing list serves as a vital forum for educators, students, and developers using Python as a teaching tool. It supports discussions on curriculum development, teaching materials, and challenges in teaching programming with Python. Supported by the Python Software Foundation (PSF), this list fosters a global community dedicated to enhancing Python's role in education, making programming more accessible and effective [6].

Python Email Handling Mailing List

Focused on the development and improvement of Python's email libraries, such as `email` and `smtplib`, this mailing list is essential for developers building email automation tools. Discussions involve troubleshooting, proposing enhancements, and sharing best practices. The PSF's support ensures that Python remains a reliable tool for email handling, enhancing its capabilities across various applications [6].

Python Mobile Development Mailing List

This specialized mailing list is dedicated to integrating Python with mobile platforms. As mobile computing grows, this list is increasingly important for developers working with frameworks like Kivy and BeeWare. Discussions focus on mobile-specific APIs and optimizing Python for mobile environments. The PSF supports this list to explore new opportunities and overcome challenges in mobile computing, ensuring Python's relevance in this expanding domain [6].

Python Database Mailing List

The Python Database mailing list enhances Python's integration with various database systems. It is a key resource for developers working on database-driven applications, from simple scripts to complex web applications. Discussions include best practices, database connectors, and new database technologies, ensuring Python's continued excellence in database-related tasks across industries [6].

2.2.2 Governance and Knowledge Sharing via Mailing Lists

Role in Governance and Decision-Making

Mailing lists play a pivotal role in the governance of the Python project, particularly within specialized domains. While the overall governance of Python is primarily managed through the `python-dev` mailing list, specialized lists play a crucial role in decision-making within their respective areas. For instance, decisions regarding the direction of Python's education initiatives, email handling libraries, mobile development tools, and database integrations are often shaped by discussions within these specialized groups. The input gathered from these discussions is vital for ensuring that Python continues to meet the needs of its diverse user base, reflecting the collective vision of the community [7].

Knowledge Sharing and Archiving

The mailing lists for education, email handling, mobile development, and databases serve as rich repositories of knowledge. These archives document the challenges faced by the community, the solutions proposed, and the decisions made over time, providing a valuable resource for current and future developers. Additionally, these mailing lists facilitate the cross-pollination of ideas between different SIGs, driving innovation across the Python ecosystem. The comprehensive nature of these archives ensures that knowledge is preserved and accessible, fostering continuous learning and collaboration within the community [11].

2.2.3 Challenges Associated with Mailing Lists

Despite their numerous benefits, mailing lists within the Python community are not without challenges. Some of the key challenges include:

- **Information Overload:** High-traffic mailing lists like `python-dev` can receive hundreds of messages daily, making it difficult for participants to stay engaged with ongoing discussions. This deluge of information can be overwhelming, particularly for newcomers or less active members, potentially deterring their participation.

- **Social Dynamics:** Often, a few highly active contributors, sometimes referred to as "star contributors," dominate discussions. While these contributors are vital for driving the project forward, their dominance can inadvertently stifle diverse perspectives and discourage contributions from less experienced or newer members. This can lead to a form of groupthink, where only a narrow set of ideas is considered, limiting the community's overall innovation and inclusivity [12].

Strategies to Mitigate Challenges:

- **Moderation:** Moderation is a key tool used to manage the flow of communication on mailing lists. Moderators help ensure that discussions remain focused and that all voices are heard, preventing any single contributor from monopolizing the conversation.

- **Summarization:** The community often employs summarization techniques, where the key points from lengthy discussions are distilled and shared, making it easier for participants to stay informed without needing to read every message.

- **Mailing List Archives:** The use of mailing list archives also helps manage information overload, allowing members to catch up on missed conversations at their own pace and providing a repository of collective knowledge that can be accessed as needed [7].

2.3 Sentiment Analysis and Opinion Mining

This section delves into sentiment analysis, an essential component of Natural Language Processing (NLP), and its relevance to the Python community. Sentiment analysis, also known as opinion mining, is a powerful tool used to gauge public sentiment and opinions from textual data, making it invaluable for various industries and communities.

2.3.1 Overview of Sentiment Analysis

Introduction to Sentiment Analysis

Sentiment analysis, a subfield of Natural Language Processing (NLP), is the process of computationally identifying and categorizing opinions expressed in a piece of text. This categorization often involves determining whether the sentiment is positive, negative, or neutral. The primary goal of sentiment analysis is to understand the underlying emotions, attitudes, and opinions of the text's author, making it a crucial tool in areas like business intelligence, customer feedback systems, market analysis, and social media monitoring. The significance of sentiment analysis extends across various sectors. In the business world, companies use sentiment analysis to monitor customer reviews and social media conversations, helping them to understand consumer perceptions and react accordingly. In politics, sentiment analysis can gauge public opinion on policies, political candidates, and events, often providing insights that traditional polling might miss. Moreover, in healthcare, sentiment analysis can be used to monitor patient feedback, helping healthcare providers improve services and patient care [15].

Relevance in the Python Community

In the context of the Python community, sentiment analysis offers valuable insights into the community's collective mood and opinions. The Python community, which includes developers, educators, and users from various backgrounds, often engages in discussions through mailing lists, forums, and other communication platforms. These discussions cover a wide range of topics, from technical debates to broader conversations about the future direction of Python.

By applying sentiment analysis to these discussions, particularly on the special interest group mailing lists,the Python Software Foundation (PSF) and core developers can gain a better understanding of the community's sentiment on various issues. This understanding can help guide decision-making, ensuring that the development and governance of Python align with the community's needs and concerns. For instance, sentiment analysis could reveal widespread dissatisfaction with a proposed change, allowing the PSF to address these concerns before implementing the change [7].

Furthermore, sentiment analysis can help maintain a positive and collaborative environment within the Python community. By monitoring the tone of discussions, the PSF can identify areas where tensions might be rising and intervene if necessary to mediate conflicts or provide additional support. This proactive approach not only fosters a healthier community but also ensures that the development of Python continues smoothly and inclusively [9].

2.3.2 Approaches to Sentiment Analysis

Lexicon-Based Approaches

Overview of Lexicon-Based Methods:

Lexicon-based sentiment analysis is a method that relies on predefined dictionaries, where words are associated with specific sentiment scores. These dictionaries, or lexicons, contain words labeled with positive, negative, or neutral sentiments. When analyzing a piece of text, the lexicon-based approach evaluates the sentiment by matching the words in the text with those in the dictionary and summing the associated sentiment scores. This method is relatively simple and can be highly effective, especially when the text has a clear and direct sentiment.

Lexicon-based approaches are particularly useful for their interpretability; they allow users to easily understand how the sentiment score is derived. However, these methods may struggle with context-dependent sentiments, sarcasm, and nuanced expressions that do not rely solely on specific words with predefined sentiment scores [8].

VADER (Valence Aware Dictionary and Sentiment Reasoner):

VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media contexts. Developed by C.J. Hutto and Eric Gilbert, VADER is known for its effectiveness in handling social media text, which often includes informal language, emoticons, and other non-standard forms of communication.

VADER works by using a lexicon of sentiment-related words, which are rated on their sentiment intensity. In addition to simple word matching, VADER applies heuristics to account for factors like punctuation, capitalization, degree modifiers (e.g., ”very” happy), and the contrastive conjunction ”but,” which can change the sentiment of a sentence. For example, in the sentence ”I love Python, but the documentation could be better,” VADER would recognize that the positive sentiment is tempered by the contrasting ”but.”

VADER's applicability to Python mailing lists comes from its ability to handle the informal and technical language often found in community discussions. Its rule-based enhancements allow it to better capture the sentiment nuances that are common in such contexts, making it a valuable tool for analyzing the mood and opinions within the Python community [8].

Machine Learning Approaches

BERT (Bidirectional Encoder Representations from Transformers):

BERT is a transformer-based model that has revolutionized the field of Natural Language Processing (NLP) by allowing for a deep understanding of context in text. Unlike traditional models, which typically read text in a single direction (left to right or right to left), BERT processes text bidirectionally. This means BERT considers the full context of a word by

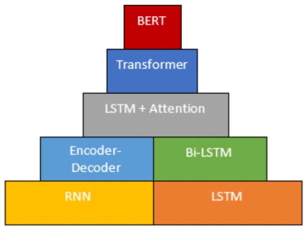


Figure 2.2: Hierarchy of NLP Models, with BERT at the top, showcasing its advanced capabilities in understanding context. Source: [10]

looking at the words before and after it, which is crucial for understanding the nuances of language.

In sentiment analysis, BERT excels at capturing the context and subtleties in the text that simpler models might miss. For example, BERT can differentiate between the sentiments expressed in sentences like ”Python is simple to learn” and ”Python is not difficult to learn,” understanding that both convey a positive sentiment despite their structural differences.

Applying BERT to Python mailing lists enables a more sophisticated analysis of the sentiments expressed in discussions. Since these discussions often involve technical jargon and context-specific references, BERT's ability to grasp the full context of the text is particularly beneficial. It allows for a more accurate sentiment classification, reflecting the true mood of the community on various topics [4].

DistilBERT:

DistilBERT is a smaller, faster, and more efficient version of BERT, developed through a process called knowledge distillation. In this process, a smaller model (DistilBERT) is trained to mimic the behaviors of a larger, more complex model (BERT). The result is a model that retains most of BERT's capabilities but is less resource-intensive, making it more practical for large-scale or real-time applications.

In sentiment analysis, DistilBERT provides a balance between performance and efficiency. It maintains the ability to understand context and nuances in text, similar to BERT, but with reduced computational overhead. This makes it ideal for scenarios where quick processing is essential, such as real-time sentiment analysis of active Python mailing lists.

Using DistilBERT to analyze sentiments in the Python community offers the benefits of advanced contextual understanding while being more adaptable to the community's needs. It is particularly useful when dealing with the high volume of data generated by mailing lists, as it can deliver timely insights without requiring extensive computational resources [13].

Latent Dirichlet Allocation(LDA):

The paper ”Latent Dirichlet Allocation” (LDA) by Blei, Ng, and Jordan (2003) provides a comprehensive and foundational approach to topic modeling, which is especially relevant for sentiment analysis in mailing lists. LDA is a generative probabilistic model that assumes each document in a corpus is a mixture of topics, with each topic characterized by a distribution over words. This method is particularly effective in uncovering the underlying thematic structure within large text datasets, such as mailing lists, where discussions often span multiple topics with varying sentiment tones [2].

In the context of sentiment analysis for mailing lists, LDA is instrumental in identifying distinct topics within the text, which can subsequently be analyzed for sentiment. By applying LDA, one can categorize different segments of mailing list discussions into coherent topics, even when the discussions are unstructured and cover a broad range of issues. This categorization enables more targeted sentiment analysis, allowing for the sentiment of each identified topic to be assessed individually, thereby providing more granular insights into the community's perspectives [2].

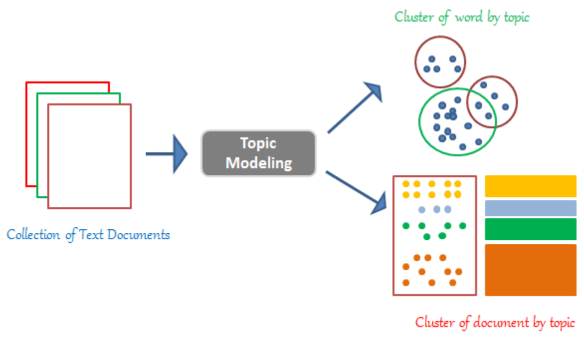


Figure 2.3: An illustration of LDA, showing how it clusters words and documents by topics, revealing underlying themes that are crucial for sentiment analysis in large text datasets.[1].

LDA's strength in handling the "bag of words" assumption is particularly advantageous for sentiment analysis, where the context of word usage is crucial. The model considers each word in the context of its topic, facilitating a better capture of the sentiment associated with specific topics, particularly in complex discussions typical of technical mailing lists. This contrasts with simpler models that may overlook the nuances of context, leading to less accurate sentiment predictions [2].

Furthermore, LDA's ability to generalize beyond the training data is significant for analyzing mailing lists, which are often dynamic and evolving. The model's generative nature allows it to assign probabilities to unseen documents, making it a robust tool for ongoing sentiment analysis as new discussions emerge [2].

In summary, LDA's ability to uncover latent topic structures in mailing lists makes it an indispensable tool in sentiment analysis. It enhances the accuracy of sentiment predictions by providing contextually relevant topic categorizations and ensures that the analysis remains robust over time as new data becomes available. By integrating LDA with sentiment analysis techniques like VADER, BERT, and DistilBERT, a more nuanced and precise understanding of community sentiment can be achieved, leading to better-informed decisions based on the analyzed text data [2].

Hybrid Approaches

Combining Lexicon-Based and Machine Learning Methods:

Hybrid approaches to sentiment analysis combine the strengths of lexicon-based methods like VADER with advanced machine learning models like BERT and DistilBERT. The integration of these methods can significantly enhance the accuracy and depth of sentiment analysis by leveraging the interpretability of lexicon-based tools and the contextual understanding of machine learning models.

In practice, a hybrid approach might involve using VADER to provide an initial sentiment score based on the presence of sentiment-laden words, followed by refining this score using a model like BERT that takes the full context into account. This combination allows for a more nuanced analysis, capturing both the explicit sentiments expressed in the text and the subtler, context-dependent nuances that a lexicon-based method alone might miss.

For the Python community, where discussions can range from highly technical debates to informal exchanges, a hybrid approach ensures that sentiment analysis is both accurate and comprehensive. By addressing the limitations of individual methods, hybrid approaches offer a more robust tool for understanding the complex sentiments within the community [15].

2.3.3 Sentiment Analysis on Python Mailing Lists

Sentiment analysis can be a powerful tool for understanding the dynamics within the Python community by analyzing the discussions taking place on various mailing lists. For instance, applying sentiment analysis to the Python Education mailing list could help identify the community's attitudes towards different educational initiatives, resources, and teaching strategies. By tracking positive and negative sentiments over time, the Python Software Foundation (PSF) and educators can gain insights into the effectiveness of educational tools and the community's needs [7].

Similarly, sentiment analysis on the Python Email Handling mailing list can provide insights into the challenges and satisfaction levels regarding Python's email libraries. For example, spikes in negative sentiment might indicate widespread frustration with a particular issue, prompting quicker resolutions and improvements. The analysis of sentiment on the Python Mobile Development and Python Database mailing lists can likewise help developers and project maintainers understand the community's reception of new tools, updates, and best practices in these areas. By identifying trends in sentiment, stakeholders can make more informed decisions, prioritize development efforts, and address community concerns more effectively [7, 12].

Overall, sentiment analysis serves as a valuable feedback mechanism, enabling the Python community to stay responsive to its members' needs and sentiments. It can highlight areas where the community is thriving, as well as spots where more attention or support may be needed [9].

2.3.4 Challenges in Sentiment Analysis

Contextual and Linguistic Challenges:

- **Detecting Sarcasm and Irony:** One of the most significant challenges in sentiment analysis is detecting sarcasm and irony. These linguistic features can completely reverse the sentiment of a statement, making it difficult for sentiment analysis tools to interpret them correctly. For instance, the phrase "Just what we needed, another Python update," when spoken sarcastically, conveys a negative sentiment, even though the words themselves might be interpreted as positive. Current sentiment analysis models, including those based on VADER or BERT, often struggle to accurately identify sarcasm, leading to potential misclassification of sentiment [8].

- **Contextual Understanding:** Another major challenge is the need for a deep understanding of context. Sentiment can shift dramatically depending on the surrounding text and the specific topic under discussion. For example, the sentiment expressed in technical discussions on Python mailing lists can vary based on the subject matter, with the same phrase carrying different sentiments in different contexts. Traditional sentiment analysis models might overlook these nuances, especially in technical or specialized domains like Python development, leading to oversimplified or inaccurate sentiment scores [4].

Domain-Specific Challenges:

- **Domain Dependence:** Sentiment analysis models trained on general datasets may not perform well in specialized domains like Python mailing lists without domain-specific tuning. The language used in these mailing lists often includes technical jargon, abbreviations, and context-specific references that general models are not equipped to handle. As a result, these models may fail to accurately capture the sentiment expressed in these specialized discussions, underscoring the need for domain-specific sentiment analysis tools [15].

- **Information Overload:** Mailing lists, particularly high-traffic ones like those in the Python community, generate large volumes of messages daily. This information overload can complicate the application of sentiment analysis, as the sheer volume of data makes it difficult to process and analyze all relevant content effectively. Moreover, the diverse topics and varying levels of technical detail in these messages can further challenge sentiment analysis models, which may struggle to maintain accuracy across such a wide range of content [12].

2.3.5 Future Directions

Domain-Specific Model Development:

Given the challenges identified, there is a clear need for further research into developing sentiment analysis models tailored specifically for the Python community. These models should be fine-tuned to handle the unique language, technical jargon, and context-specific discussions that characterize Python mailing lists. By creating models that are better suited to these specialized domains, researchers can improve the accuracy and relevance of sentiment analysis, making it a more powerful tool for community management and decision-making [9].

Improving Contextual and Sarcasm Detection:

Another important area for future research is the development of more sophisticated techniques for detecting context and sarcasm in text. This could involve enhancing existing models like BERT or VADER with additional training on datasets rich in sarcastic and context-dependent language. Alternatively, new models could be developed that specifically target these challenges, using advanced NLP techniques to better understand the subtleties of human communication. By addressing these issues, sentiment analysis can become more reliable and insightful, even in the complex and varied discussions found within the Python community [4].

Chapter 3

Experimental Design and Methodology

3.1 Goals of the Experiments

3.1.1 Research Objectives

The experiments in this study are designed to evaluate the performance of various sentiment analysis models within the context of technical mailing lists, with a particular focus on comparing their effectiveness under different methodological approaches. The first experiment aims to assess the capabilities of VADER, BERT, and DistilBERT in accurately capturing sentiments expressed in technical discussions. The second experiment explores the impact of using predefined subtopics on the performance of topic modeling and sentiment analysis, aiming to determine whether guiding the analysis with specific categories enhances the relevance of the results. The third experiment shifts to an unsupervised approach, allowing topics to emerge naturally without predefined categories, to evaluate how this affects the coherence and accuracy of sentiment analysis. Collectively, these experiments seek to identify the most effective strategies for sentiment analysis in specialized, technically oriented textual datasets.

3.1.2 Hypotheses

The experiments in this study are guided by the hypothesis that advanced sentiment analysis models, such as BERT and DistilBERT, will outperform simpler models like VADER in accurately capturing the nuanced sentiments within technical mailing lists. Due to their deep learning architecture and high-level contextual understanding, BERT and DistilBERT are expected to provide more precise sentiment classifications, particularly in complex and context-specific discussions that are characteristic of technical domains.

Furthermore, it is hypothesized that the use of predefined subtopics will enhance the performance of these models by providing structured guidance, thereby improving the relevance and accuracy of the sentiment analysis. This structured approach is anticipated to yield better results compared to an unsupervised method where topics emerge naturally without predefined guidance. The results of these hypotheses will help determine the most effective sentiment analysis strategy for specialized, technical content.

3.2 Choice of Programming Language

3.2.1 Python

Python was chosen for these experiments due to its preeminence in data science and natural language processing (NLP). Its simplicity, readability, and extensive library ecosystem make it ideal for implementing machine learning models and handling text analysis tasks. Python's integration with powerful libraries like NLTK, Scikit-Learn, and PyTorch ensures efficient processing, from data preprocessing to deploying advanced models like BERT and DistilBERT. Moreover, since this study focuses on Python mailing lists, using Python aligns with the subject matter, leveraging the language's strengths to ensure precise and effective sentiment analysis.

3.2.2 Related Libraries

NLTK:

The Natural Language Toolkit (NLTK) was employed for essential preprocessing tasks, including tokenization and stop-word removal, to prepare the dataset for analysis. It also provided basic sentiment analysis capabilities, offering a preliminary sentiment overview before applying advanced models.

TextBlob:

TextBlob was utilized for its simplicity in assessing text polarity and subjectivity. It provided quick sentiment evaluations, serving as an efficient tool for initial sentiment analysis and for validating results from more complex models.

Scikit-Learn:

Scikit-Learn played a key role in implementing machine learning models for tasks like topic modeling and classification. It offered powerful tools for vectorizing text and applying algorithms like Latent Dirichlet Allocation (LDA), making it a crucial component in the analysis pipeline.

Regex (re):

The regular expressions library, `re`, was used for text processing tasks, such as pattern matching and string manipulation. It was particularly useful in parsing and segmenting the raw email data into individual emails based on identifiable patterns, enabling more structured analysis.

3.3 Dataset

3.3.1 Source and Description

The dataset utilized in this study consists of emails from the Python Special Interest Group (SIG) mailing lists, specifically focusing on discussions related to mobile development, database management, educational applications, and email handling. These mailing lists capture a wide range of technical discussions from the years 2015 to 2024, providing valuable insights into the sentiments and opinions of Python developers across these specialized areas. The dataset is comprehensive, comprising thousands of emails, which ensures a robust foundation for sentiment analysis. This diverse collection allows for a detailed exploration of sentiments expressed in the context of Python's mobile, database, educational, and email handling domains.

3.3.2 Data collection Methods

The dataset was collected using a systematic approach involving web scraping, text processing, and file handling. The `requests` library was employed to fetch email archives from the Python Special Interest Group (SIG) mailing lists, covering the period from 2015 to 2024. The `re` library (regular expressions) was used to parse the raw data, splitting it into individual emails based on identifiable patterns. This ensured each email was isolated for easier processing and analysis. Finally, Python's built-in file handling functions were utilized to save the processed emails into structured text files, making the dataset ready for subsequent sentiment analysis.

3.4 Data Preprocessing

3.4.1 Text Cleaning

The dataset underwent a comprehensive text cleaning process to prepare it for analysis. This involved systematically removing all extraneous elements such as newline characters, redundant whitespace, and irrelevant content like email headers and footers. To ensure uniformity across the dataset, standardization steps were applied, including converting all text to lowercase and eliminating punctuation. These cleaning procedures were crucial in transforming the raw, unstructured email data into a consistent and analyzable format, providing a solid foundation for the subsequent sentiment analysis and topic modeling tasks.

3.4.2 Tokenization and Normalization

After cleaning, the text was subjected to tokenization and normalization to further refine it for analytical purposes. Tokenization, carried out using NLTK's tokenizer, broke down the text into individual tokens, enabling detailed analysis. Stop words—common words that do not contribute significantly to the meaning—were removed to enhance the focus on meaningful content. Additionally, normalization techniques, such as converting the text to lowercase and filtering out non-alphanumeric characters, were applied to standardize the data. These preprocessing steps were essential in preparing the dataset for effective application in machine learning models and sentiment analysis, ensuring that the data was in the optimal format for accurate and insightful analysis.

Chapter 4

Experiments and Results

4.1 Experiment 1: Establishing a Baseline with TextBlob and Evaluating VADER, BERT, and DistilBERT

4.1.1 Experiment Purpose

The purpose of this experiment is to compare the performance of three sentiment analysis models—VADER, BERT, and DistilBERT—in analyzing sentiments across various subtopics in mobile development discussions. The goal is to understand how each model handles sentiment classification and to identify which model provides the most accurate and nuanced sentiment analysis for this specific domain.

4.1.2 Implementation

Top 10 Most Discussed Topics:
The following table presents the top 10 most discussed topics in the mobile development mailing list, along with the total number of mentions and their sentiment distribution as identified by TextBlob:

Topic	Mentions	TextBlob Positive	TextBlob Negative
python	209	146	63
android	181	133	48
build	107	83	24
platform	99	86	13
ios	69	61	8
kivy	59	53	6
toga	27	27	0
rubicon	25	25	0
beeware	25	23	2
cross-platform	15	14	1

Table 4.1: Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob)

This table highlights the distribution of mentions and the polarity of sentiments (positive or negative) for each topic using TextBlob. These results serve as a baseline for comparison with the sentiment analysis outcomes from VADER, BERT, and DistilBERT.

4.1.3 Sentiment Analysis Results

Sentiment Analysis Using VADER:
The VADER model was applied to the dataset, and the results for each topic are summarized in the following table:

Topic	Positive	Negative	Neutral
android	157	22	2
python	183	23	3
build	93	14	0
platform	90	9	0
ios	68	1	0
kivy	55	4	0
toga	26	1	0
rubicon	25	0	0
beeware	25	0	0
cross-platform	15	0	0

Table 4.2: Sentiment Analysis for Mobile Mailing List using VADER

These results suggest that VADER generally finds a higher proportion of positive sentiments across most topics, with minimal neutral sentiments detected.
Sentiment Analysis Using BERT:

Topic	1 star	2 stars	3 stars	4-5 stars
android	141	14	13	13
beeware	16	4	1	4
build	85	6	5	11
cross-platform	13	2	0	0
ios	46	8	8	7
kivy	46	5	3	5
platform	68	14	6	11
python	159	16	18	16
rubicon	17	5	0	3
toga	18	5	1	3

Table 4.3: Sentiment Analysis for Mobile Mailing List using BERT

Explanation of Star Ratings:
- **1 star:** Very negative sentiment - **2 stars:** Negative sentiment - **3 stars:** Neutral sentiment
- **4 stars:** Positive sentiment - **5 stars:** Very positive sentiment

Sentiment Analysis Using DistilBERT:

Topic	NEGATIVE	POSITIVE
android	174	7
beeware	24	1
build	103	4
cross-platform	15	0
ios	62	7
kivy	57	2
platform	95	4
python	199	10
rubicon	23	2
toga	27	0

Table 4.4: Sentiment Analysis for Mobile Mailing List using DistilBERT

4.1.4 Discussions

The sentiment analysis results for mobile development discussions reveal distinct differences in how VADER, BERT, and DistilBERT interpret the same content across various subtopics.

- **VADER:** This lexicon-based model tends to detect more positive sentiments across most topics, such as "android," "python," and "build." It assigns fewer negative or neutral sentiments, indicating a bias toward positive interpretation. VADER's reliance on pre-built sentiment dictionaries might cause it to overlook the nuanced or critical aspects of technical discussions, simplifying the complexity of the discourse and potentially underestimating the challenges discussed.

- **BERT:** BERT provides a more balanced but critical sentiment analysis, especially in subtopics like "android," "build," and "cross-platform," where it identifies a greater amount of negative sentiment. Its transformer-based architecture allows for a deeper understanding of the context, leading to the detection of subtleties and criticisms that VADER often misses. However, this depth can sometimes skew BERT's results towards a more negative interpretation, even in cases where the sentiment may not be overtly critical.

- **DistilBERT:** Like BERT, DistilBERT also detects more negative sentiment in technical topics such as "android" and "build." It performs well in capturing negative tones but shows some balance with positive sentiment detection in subtopics like "python" and "rubicon." However, its sensitivity to the critical aspects of the discussion can lead to an overemphasis on negative sentiment, requiring fine-tuning to reduce this bias in technical mailing list analysis.

In conclusion, VADER tends to provide a more positive outlook on the sentiment, presenting an optimistic view of the discussions. On the other hand, BERT and DistilBERT delve deeper into the content, uncovering more critical perspectives. These transformer-based models excel at capturing subtle emotional tones but may sometimes amplify negative sentiment, particularly in technical discussions, unless fine-tuned to the specific domain.

4.2 Experiment 2: Topic Modeling and Sentiment Analysis Using LDA with Predefined Subtopics

4.2.1 Experiment Purpose

The goal of this experiment is to evaluate the influence of predefined subtopics on the effectiveness of topic modeling and sentiment analysis using Latent Dirichlet Allocation (LDA). By setting predefined subtopics, the experiment aims to guide the LDA model to generate more accurate and contextually relevant topics, which are then analyzed for sentiment.

4.2.2 Implementation

Methodology Overview:

1. **Data Preprocessing:**
    - The preprocessed emails from the mobile development discussions were loaded into a DataFrame.
    - Empty or irrelevant strings were removed to ensure that the dataset was clean and ready for analysis.
2. **Vectorization:**
    - A CountVectorizer was applied to transform the text data into a document-term matrix, using a vocabulary limited to words appearing in at least two documents but not in more than 95% of the corpus.
3. **LDA Model Setup:**
    - The LDA model was configured to discover 10 topics, aligned with the predefined subtopics such as Android, iOS, Python, and others relevant to mobile development.
    - After fitting the LDA model, the most significant words for each topic were identified, ensuring that they corresponded to the expected subtopics.
4. **Topic Assignment:**
    - Each document was assigned a dominant topic based on the LDA output, which was then mapped to one of the predefined subtopics.

- This mapping helped categorize each document under a relevant topic for further sentiment analysis.

5. **Sentiment Analysis:**

   - Sentiment analysis was performed using VADER, BERT, and DistilBERT models to evaluate the sentiments associated with each document within the identified topics.
   - The sentiment for each document was classified based on the outputs of these models, focusing on the most frequent sentiment detected within each subtopic.

4.2.3 Sentiment Analysis Results

The table offers a comparative summary of sentiment analysis results for key topics, evaluated using VADER, BERT, DistilBERT.It highlights the different interpretations of sentiment across models, showcasing the variations between lexicon-based, transformer-based, and topic modeling approaches.

Topic	**VADER Sentiment**	**BERT Sentiment**	**DistilBERT Sentiment**
android	0.9957	2 star	NEGATIVE
beeWare	0.6369	1 star	NEGATIVE
build	0.9504	1 star	NEGATIVE
cross-platform	0.7906	1 star	NEGATIVE
ios	0.9516	1 star	NEGATIVE
kivy	0.9829	3 star	NEGATIVE
platform	0.296	1 star	NEGATIVE
python	0.9996	3 star	POSITIVE
rubicon	0.9969	2 star	POSITIVE
toga	0.0	2 star	NEGATIVE

Table 4.5: Sentiment Analysis across Subtopics

VADER Score Range:

- **Positive Score**: Between 0 and 1, where 1 means a very positive sentiment.
- **Negative Score**: Below 0
- **Neutral Sentiment**: Closer to 0, or scores near zero.

4.2.4 Discussions

The sentiment analysis results across predefined subtopics reveal notable differences in how each model interprets the sentiment of discussions.

- **VADER** consistently detects more positive sentiments across most subtopics, especially in "android," "python," and "kivy." Its lexicon-based approach, while efficient, may oversimplify the sentiment and fail to capture the depth of technical challenges or frustrations within the discussions, which leads to a generally more optimistic interpretation.
- **BERT** and **DistilBERT**, on the other hand, lean heavily towards negative sentiments in most subtopics, including "android," "build," and "cross-platform." These models, being more contextually sensitive, often overemphasize critical or negative language, which may result in skewed sentiment outcomes. This tendency to flag discussions as negative could be a reflection of how these models interpret technical debates or problem-solving dialogues as inherently negative, even when the discussions might be more neutral or solution-oriented.
- Interestingly, **DistilBERT** showed some variation from BERT by detecting more positive sentiment in subtopics like "rubicon" and "python." Despite this, both models generally identified more negativity than VADER, suggesting that advanced models, while more nuanced, can sometimes over-interpret the challenges discussed in mailing lists as overly negative.

In summary, while VADER offers a more positive, albeit simplistic, sentiment reading, BERT and DistilBERT tend to overestimate negativity, indicating that these models may require further fine-tuning to balance their interpretations in technical discussions.

4.3 Experiment 3: Topic Modeling and Sentiment Analysis Using LDA Without Predefined Subtopics

4.3.1 Experiment Purpose

The purpose of this experiment is to analyze the effectiveness of Latent Dirichlet Allocation (LDA) when applied without predefined subtopics, allowing topics to emerge naturally from the data. The experiment aims to compare these results with those obtained in Experiment 2, where predefined subtopics guided the LDA process. The comparison will help in understanding the implications of using an unsupervised approach versus a guided topic modeling approach on both the quality of topics generated and the associated sentiment analysis.

4.3.2 Implementation

Methodology Overview

1. **Data Preprocessing:**
   The raw email data from the mobile development mailing list was preprocessed to remove noise and irrelevant content. This resulted in a clean corpus ready for topic modeling.

2. **Vectorization:**
   A CountVectorizer was used to create a document-term matrix (DTM), which serves as the input for the LDA model. The vectorization was performed with parameters that excluded overly common or rare terms to enhance the quality of topics discovered.

3. **LDA Model Setup:**
   The LDA model was run with 10 topics, without any predefined subtopics. This allows the model to identify topics based purely on the distribution of words in the corpus. The model was configured with specific priors and a large number of iterations to ensure convergence and stability.

4. **Subtopic Generation:**
   Instead of mapping topics to predefined subtopics, we used a Named Entity Recognition (NER) pipeline to automatically generate names for the topics based on the most significant words within each topic. This process identifies entities like organizations, products, and locations to create meaningful subtopic names.

5. **Sentiment Analysis:**
   Sentiment analysis was conducted on the content of each document using VADER, BERT, and DistilBERT models. The sentiment scores from these models were analyzed individually to determine the dominant sentiment for each emergent subtopic.

4.3.3 Sentiment Analysis Results

The LDA model was applied without predefined subtopics, allowing topics to naturally emerge from the data. The table below presents the identified topics, their broader categorization, and the corresponding sentiment analysis scores from VADER, BERT, and DistilBERT models.

The LDA model generated topics based on the natural structure of the text data, without any predefined guidance. Each topic was categorized into broader categories relevant to mobile development, such as Android Development, Python Libraries, and iOS Development. Sentiment analysis was conducted using VADER, BERT, and DistilBERT, revealing significant differences in how each model interprets the sentiment within these topics.

Identified Topic	**VADER Sentiment**	**BERT Sentiment**	**DistilBERT Sentiment**
native way things api specific 2017 developers mobile thread python https apis david gui using	0.9957	2 stars	NEGATIVE
toolchain class code new wed android jun https voc python jython david bytecode 2017 java	0.9516	1 star	NEGATIVE
voc working https mobile app android class add libffi patch ios cpython python project support	0.9996	2 stars	NEGATIVE
device patches working import module file feb http https kivy wrote 2015 build android python	0.9504	1 star	NEGATIVE
scrubbed attachment django importerror named module cross packages hi devices jul sep fri python 2015	0.296	1 star	NEGATIVE

Table 4.6: Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT

4.3.4 Discussions

- **Topic Coherence:** Without predefined subtopics, the LDA model generated broader and sometimes less coherent topics. While some topics like ”Android Development” and ”Python Libraries” were well-defined, others were more ambiguous or combined multiple subtopics.
- **Sentiment Variability:** Sentiment analysis results varied significantly across the topics, with VADER generally indicating more positive sentiments, while BERT and DistilBERT provided more critical sentiment assessments. The lack of predefined guidance likely led to more diverse and sometimes conflicting sentiment scores within the same topic.
- **Implications:** The unsupervised approach allowed for the discovery of natural topic structures, which can be useful for exploratory analysis. However, the results were less focused compared to the predefined subtopic approach, indicating that guided topic modeling might be preferable when specific, well-defined topics are needed for sentiment analysis.

In addition to the results discussed in this section, additional sentiment analysis outcomes for other mailing lists are available in the Appendix, complementing the primary results presented here.

4.4 Comparitive Analysis and Survey

4.4.1 Rationale Behind Model Selection

The decision to use VADER, BERT, DistilBERT, and TextBlob for sentiment analysis in this study was driven by the need to compare the performance of models across varying levels of complexity and sophistication. By employing lexicon-based, transformer-based, and machine learning models, the study aims to comprehensively understand how different sentiment analysis techniques perform on the same dataset, focusing on Python mailing lists related to development, education, and community discussions.

TextBlob

TextBlob was selected as a baseline model due to its simplicity and ease of use. It is a rule-based sentiment analysis tool that relies on a lexicon to classify sentiment as positive, neutral, or negative. TextBlob provides a quick and relatively effective sentiment classification for general-purpose text analysis without requiring extensive training or computational resources. Since the goal was to explore how different models handle technical and domain-specific discussions, TextBlob serves as a useful comparison to the more sophisticated models. Its performance helps set the benchmark for how simpler models handle unstructured, developer-centric text data like those found in Python mailing lists.

VADER

VADER (Valence Aware Dictionary and sentiment Reasoner) was chosen because it is specifically designed for social media and informal text, making it highly effective at analyzing sentiments in short messages and comments. Unlike TextBlob, VADER accounts for the intensity of sentiment and handles nuances like exclamation points, capitalization, and even the polarity of emoticons. VADER performs well in contexts where quick, rule-based sentiment analysis is needed without the overhead of training data. The primary advantage of VADER is its ability to accurately identify positive, negative, and neutral sentiments in informal, technical discourse while remaining lightweight and efficient.

BERT

BERT (Bidirectional Encoder Representations from Transformers) was included as a representative of the more advanced, deep learning-based models. BERT is known for its contextual understanding, allowing it to capture nuances in sentiment that simpler models like TextBlob and VADER might miss. It uses a bidirectional approach to analyze the entire sentence, rather than parsing it in a left-to-right or right-to-left manner, which results in a deeper understanding of context, especially in complex and technical discussions. BERT's pre-training on a vast corpus of text enables it to handle the intricacies and jargon often found in technical mailing lists, making it ideal for identifying sentiment in highly specialized domains like Python development.

DistilBERT

DistilBERT was selected as a medium-complexity model, positioned between the simplicity of VADER and TextBlob and the deep-learning complexity of BERT. DistilBERT is a smaller, faster version of BERT that retains much of the same accuracy but with fewer parameters, making it more computationally efficient. Given the large volume of data in mailing lists, DistilBERT provides a balance between performance and resource consumption. The choice to include DistilBERT was motivated by its potential to offer high-quality sentiment analysis without the heavy computational load of BERT, thus making it a practical alternative for real-time or large-scale analysis.

LDA (Latent Dirichlet Allocation)

Latent Dirichlet Allocation (LDA) was chosen to aid in topic modeling, which plays an essential role in structuring the data for sentiment analysis. LDA helps identify the underlying topics within mailing list discussions, making it easier to categorize and focus the sentiment analysis on specific subtopics. By organizing the text into coherent topics, LDA enhances the precision of sentiment analysis, ensuring that the sentiment is evaluated in the appropriate context.

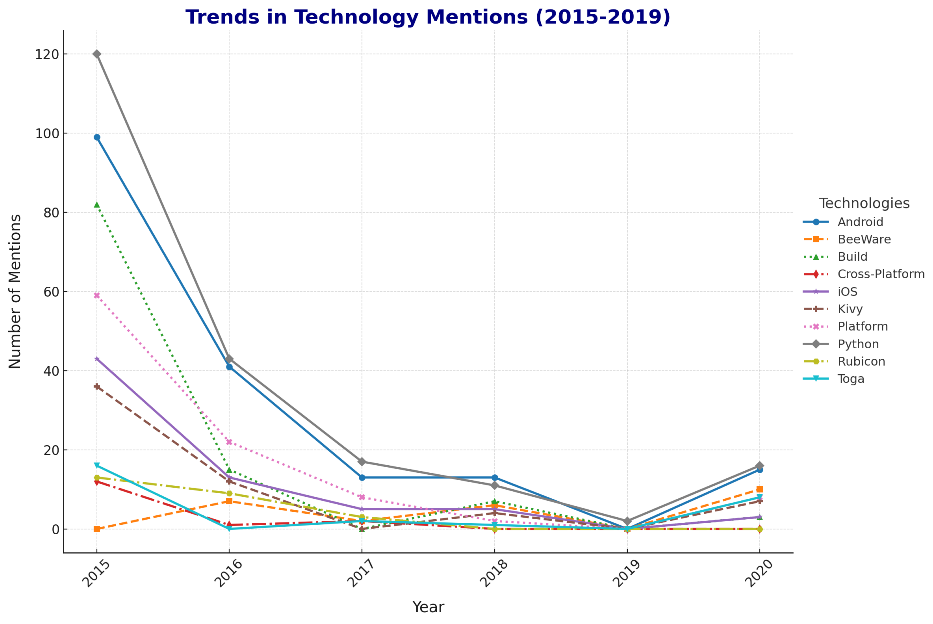
4.4.2 Summary of Experimental Results

The experiments conducted in this study revealed mixed outcomes regarding the initial hypotheses. While the advanced sentiment analysis models, BERT and DistilBERT, did not consistently outperform simpler models like VADER, the hypothesis that predefined subtopics enhance performance was validated. Predefined subtopics offered structured guidance, improving the relevance and accuracy of sentiment analysis, especially in complex technical discussions. This resulted in more focused topic generation, providing insights into specific areas of concern or interest in Python development.

The temporal analysis of discussion activity from 2015 to 2019 showcases significant variations in the intensity of discussions across different technologies. The early years of the analysis, particularly 2015, witnessed the highest level of discussion, especially regarding technologies like Android and Python, which had over 100 mentions. A sharp decline in activity is observed from 2016 onward, with all technologies experiencing reduced mentions. By 2017, the intensity of discussions had stabilized, remaining relatively low through 2019, with some minor fluctuations, especially in the Python and Toga communities, which saw slight increases in 2019.

These peaks and declines in discussion activity had a direct impact on the robustness of sentiment analysis results. In 2015 and 2016, the high volume of conversations provided a more comprehensive dataset, allowing for richer sentiment analysis. In contrast, the lower levels of activity in the following years meant that the sentiment data became more sparse and potentially less reflective of broader trends.

The chart below visualizes these trends in discussion activity from 2015 to 2019:



The sentiment analysis for this period showed variations across models, with VADER, TextBlob, BERT, and DistilBERT each offering different insights. While VADER consistently detected positive sentiments in most topics, the transformer-based models (BERT and DistilBERT) identified more negative sentiments, likely due to their nuanced understanding of complex discussions. This contrast highlights the varying capabilities of each model and emphasizes the importance of selecting a model that aligns with the complexity of the text being analyzed.

For example, discussions around Python, Android, and platform issues were generally positive, reflecting the constructive nature of the community's engagement. However, the technical difficulties associated with specific topics, such as cross-platform development or building processes, were flagged as negative by BERT and DistilBERT, reflecting the models' sensitivity to critical or challenging content.

The overall sentiment associated with these topics is as follows:

- **Python**: Predominantly positive sentiment.
- **Android**: Mostly positive, but with significant negative sentiment due to challenges discussed.
- **Build**: Majority positive, though notable negative sentiments reflect the difficulties.
- **Platform**: Generally positive sentiment.
- **Kivy**: Positive, with some concerns noted.
- **iOS**: Mostly positive, with fewer negative mentions.
- **Rubicon**: Entirely positive, indicating strong approval.
- **Toga**: Entirely positive, reflecting enthusiasm.
- **BeeWare**: Mainly positive, with a few challenges discussed.
- **Cross-Platform**: Mostly positive sentiment, though critical in specific contexts.

This detailed breakdown of sentiments across key topics demonstrates the value of combining multiple models for sentiment analysis, as each model brings a different perspective to the same discussions.

4.4.3 Discussion of Technique Performance

The performance of each sentiment analysis technique showed distinct strengths and weaknesses, as outlined below:

- **VADER and TextBlob:** Both of these rule-based models proved efficient in handling sentiment analysis across the technical mailing lists. They excelled in identifying general sentiments, particularly for topics like Android development, Python, and building processes. VADER, in particular, detected consistently positive sentiments, reflecting its tendency to assign higher positivity scores, especially in years where the community engaged in constructive discussions. TextBlob offered a balanced view by identifying both positive and negative sentiments. However, these models rely on predefined lexicons and rules, which can lead to oversimplified sentiment assessments, particularly in complex or ambiguous technical discussions.

- **BERT and DistilBERT:** These transformer-based models provided more nuanced sentiment analysis compared to VADER and TextBlob. BERT and DistilBERT captured a broader range of sentiment tones, identifying critical and complex sentiments that simpler models missed. However, they also detected significantly higher levels of negative sentiment, suggesting that BERT and DistilBERT may be over-sensitive to the critical tone often present in technical mailing lists. This tendency towards negative sentiment highlights the need for domain-specific fine-tuning to ensure these models accurately capture the context of technical discussions.
- **LDA (Latent Dirichlet Allocation):** LDA played a crucial role in organizing the text data into coherent topics, enhancing the overall accuracy and focus of the sentiment analysis. In experiments where predefined subtopics were used, LDA allowed for more precise analysis by guiding the sentiment models to focus on the most relevant areas. This was particularly beneficial in technical discussions, where the ability to categorize conversations into topics like Python, Android, or cross-platform issues helped refine the sentiment analysis results. In unsupervised settings without predefined subtopics, LDA generated broader and sometimes less coherent topics, which affected the consistency of the sentiment analysis. Overall, LDA's role in structuring large, unorganized datasets into meaningful topics was invaluable for analyzing the vast amount of text in Python mailing lists.

4.4.4 Peer Review and Feedback

To further validate the findings of this study, a small group of fellow students reviewed the sentiment analysis results and provided informal feedback. While not a formal peer review process, their insights helped highlight some key points and offered additional perspectives on the challenges discussed within the mailing lists.

General Positive Sentiment

The informal feedback confirmed that users generally expressed positive sentiments about the topics discussed in the mailing lists. The students noted that constructive discussions were prominent across various areas, particularly around Python development.

Challenges Discussed

The challenges that were frequently raised in the mailing lists were also echoed by the students who reviewed the findings. These challenges included:

- **Android Development:** Ongoing issues related to building and deploying Python apps on Android, such as the limitations of existing tools like Kivy.
- **Building Processes:** The complexity of maintaining reliable cross-platform build processes, which was a significant concern in the mailing lists.
- **Platform Support:** Both the mailing lists and feedback highlighted the need for improved mobile platform support within Python.
- **Kivy's Limitations:** The feedback reflected concerns discussed in the mailing lists about Kivy's complexity and how it compares to other frameworks, such as Toga.
- **Cross-Platform Parity:** Achieving consistent performance across mobile platforms was a common challenge raised in the mailing lists and reinforced by the feedback provided.

Feedback on Models and Techniques

The informal review also pointed out that while advanced models like BERT and DistilBERT provide deeper insights, they often detected more negative sentiments than the rule-based models like VADER and TextBlob. The reviewers suggested that simpler models may be preferable for certain technical domains, given their speed and ability to produce more straightforward sentiment classifications.

4.4.5 Practical Applications and Benefits of Sentiment Analysis in Open-Source Development

The findings from this study highlight the significant potential of sentiment analysis as a strategic tool in open-source software development, particularly in communities like Python's, where collaboration and feedback drive continuous improvement. Sentiment analysis offers a way to systematically assess community feedback from mailing lists, providing valuable insights that can streamline development, enhance decision-making, and foster stronger community engagement.

One of the key benefits of sentiment analysis is its ability to pinpoint areas of concern within a project. Negative sentiments in discussions, particularly around key components such as mobile development frameworks (e.g., Kivy, Cross-Platform tools), can highlight pain points or unresolved issues. Project maintainers can use this information to prioritize fixes or improvements, ensuring that the most pressing concerns are addressed. For instance, persistent dissatisfaction with cross-platform support can prompt the community to invest resources in improving compatibility or developing new features that align with users' needs.

Conversely, positive sentiment serves as a signal of success, guiding development efforts towards initiatives that resonate with the community. If sentiment analysis reveals strong approval of frameworks like Toga or Rubicon, project leaders can focus on expanding or promoting these projects. This data-driven approach allows for the strategic allocation of resources, ensuring that popular and successful tools receive the support necessary for further development and adoption.

Sentiment analysis also helps track the evolution of community engagement over time. By analyzing sentiment trends across different periods, project leaders can assess the impact of new updates, policies, or feature introductions on community sentiment. A rise in positive sentiment might reflect the success of new features or enhancements, while spikes in negative sentiment can signal issues that need attention, such as bugs, communication gaps, or dissatisfaction with recent changes.

Additionally, sentiment analysis plays a crucial role in maintaining healthy, constructive communication within open-source communities. In environments where collaboration is essential, the ability to assess the tone and sentiment of discussions helps ensure that conversations remain respectful and productive. Early detection of frustration or negativity enables maintainers to intervene and prevent discussions from becoming unconstructive, thereby fostering a positive and collaborative atmosphere.

Beyond current feedback, sentiment analysis can also detect emerging trends within the community. For instance, an increase in positive sentiment around a specific tool or feature could indicate growing interest and potential for wider adoption. Project leaders can act on these insights by prioritizing resources to support or promote such tools, ensuring that community-driven innovations are nurtured and supported.

By leveraging sentiment analysis, open-source communities like Python can make informed decisions that align with the needs and preferences of their users. This approach strengthens collaboration, improves the quality of projects, and helps maintain an active,

Chapter 5

Project Management and Skill Development

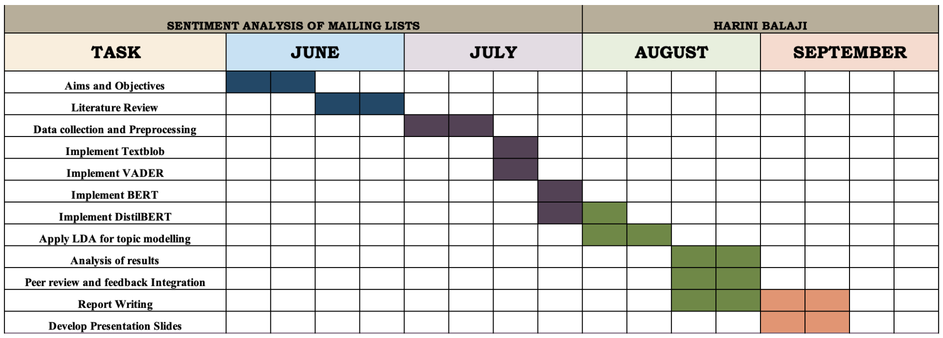
5.1 Introduction

Effective project management is crucial for ensuring the success of any research or development endeavor. Throughout this project, I developed and applied various organisational and time-management skills. By utilising modern project management techniques and tools, I was able to stay on track, meet deadlines, and respond efficiently to feedback and challenges. This chapter outlines my approach to project management

5.2 Organizational Skills and Time Management

During the course of the project, I employed several strategies to ensure that tasks were completed on time and in a structured manner.

- **Task Prioritization:** One of the key skills I developed was the ability to prioritize tasks based on their importance and deadlines. By breaking down the project into smaller, manageable tasks, I was able to focus on high-priority items first, ensuring that progress was made consistently.
- **Gantt Chart:** I utilized a Gantt chart to map out project timelines, set milestones, and track progress. This visual tool provided a clear roadmap, allowing me to anticipate potential delays and adjust the timeline accordingly. The Gantt chart helped ensure that all aspects of the project, from data collection to the final report, were aligned with the overall goals.



SENTIMENT ANALYSIS OF MAILING LISTS									HARINI BALAJI							
TASK	JUNE				JULY				AUGUST				SEPTEMBER			
Aims and Objectives	🟦	🟦														
Literature Review			🟦	🟦												
Data collection and Preprocessing					🟪	🟪										
Implement Textblob							🟪									
Implement VADER							🟪									
Implement BERT								🟪								
Implement DistilBERT								🟪	🟩							
Apply LDA for topic modelling									🟩	🟩						
Analysis of results											🟩	🟩				
Peer review and feedback Integration											🟩	🟩				
Report Writing											🟩	🟩	🟧	🟧		
Develop Presentation Slides													🟧	🟧		

Figure 5.1: Gantt Chart

5.3 Peer Review and Feedback

Incorporating feedback from peers and mentors played an essential role in refining the project.

- **Peer Review:** Throughout the project, I engaged a few fellow students to review and provide constructive feedback on various components, such as the experimental design, result analysis, and report structure. Their insights allowed me to view the project from different perspectives and make improvements where necessary.
- **Handling Criticism:** Learning how to handle and implement critical feedback was a valuable experience. For instance, comments regarding the clarity of the report and how sentiment analysis models were applied helped improve the precision of the report.

5.4 Learning Outcomes

The project provided an excellent opportunity to learn and apply a range of skills, which will be invaluable for future work:

- **Analytical Skills:** I enhanced my ability to conduct in-depth analyses, particularly in terms of interpreting sentiment analysis results across different models like BERT, VADER, and TextBlob.
- **Collaboration and Communication:** Regular communication with peers, combined with the integration of feedback, has improved my collaborative skills. This was particularly important when refining the report structure and discussing experiment outcomes.
- **Adaptability:** Throughout the project, I had to adapt to changes in the scope and timeline. For instance, certain models required additional time for tuning and testing, which required me to adjust my original schedule without compromising on quality.

Chapter 6

Conclusions

This research has delivered an in-depth exploration of sentiment analysis and topic modelling tailored specifically to Python mailing lists. By conducting a thorough comparison of sentiment analysis models such as VADER, BERT, and DistilBERT, the study reveals the unique strengths and limitations of each model. VADER, known for its simplicity and efficiency, is well-suited for broad sentiment analysis, yet it may lack the precision required for technical discussions. On the other hand, BERT and DistilBERT provide more nuanced sentiment analysis but demand significant fine-tuning to navigate the intricacies and technical language used in specialised domains.

The study also highlights the importance of leveraging predefined subtopics to guide sentiment analysis. This structured approach significantly improves the relevance of the results, especially in contexts where detail and clarity are essential. The techniques and tools developed through this work present practical frameworks that can be adapted for continuous community engagement and further development. This research not only deepens the understanding of sentiment analysis in technical contexts but also offers actionable frameworks and insights that can benefit the Python community and other highly specialized fields.

Bibliography

[1] Bansal, H. Latent dirichlet allocation. https://medium.com/analytics-vidhya/latent-dirichelt-allocation-1ec8729589d4, 2020. Accessed: 28-Sep-2021.

[2] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[3] Canedo, E. D., et al. Barriers faced by women in software development projects. *IEEE Transactions on Software Engineering* (2020).

[4] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2019).

[5] Ducheneaut, N. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW)* (2005).

[6] Foundation, P. S. Python events and conferences, 2023. https://www.python.org/events/.

[7] Guzzi, A., et al. Communication in open source software development mailing lists. *IEEE Transactions on Software Engineering 39*, 2 (2013), 165–180.

[8] Hutto, C., and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media* (2014), pp. 216–225.

[9] Jiang, Z. M., Adams, B., and Hassan, A. E. Examining the evolution of code review practices in open source projects. In *Proceedings of the 39th International Conference on Software Engineering* (2017), pp. 135–146.

[10] McCormick, C. Bert research ep.1-key concepts & sources, 2019. Available: https://mccormickml.com/2019/11/11/bert-research-ep-1-key-concepts-and-sources/ [Accessed: 28-Sep-2021].

[11] Mens, T., Claes, M., Grosjean, P., and Serebrenik, A. Studying evolving software ecosystems based on ecological models. In *Proceedings of the 2014 Software Evo-*

lution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE 2014) (2014), pp. 403–406.

[12] Rigby, P., and Hassan, A. What can oss mailing lists tell us? a preliminary psychometric text analysis. In *Proceedings of the Fourth International Workshop on Mining Software Repositories* (2007), pp. 23–26.

[13] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop* (2019).

[14] van Rossum, G., and Drake, F. L. *Python 2.1 Documentation*. BeOpen PythonLabs, 2001.

[15] Walaa Medhat, A. H., and Korashy, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal 5*, 4 (2014), 1093–1113.

[16] Ye, Y., and Kishida, K. Toward an understanding of the motivation of open source software developers. In *Proceedings of the 25th International Conference on Software Engineering (ICSE 2003)* (2003), pp. 419–429.

Appendices

Appendix A

RESULTS

Education Mailing Lists

Sentiment Analysis Using Textblob:

Topic	Total Mentions	Negative Sentiment	Positive Sentiment
coding	207	28	179
curriculum	133	10	123
edtech	4	0	4
education	222	20	202
learning	265	23	242
python	954	211	742
schools	108	9	99
students	299	24	275
teachers	166	9	157
teaching	296	28	268

Table A.1: Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob)

Sentiment Analysis Using VADER:

Topic	Negative Sentiment	Neutral Sentiment	Positive Sentiment
coding	2	0	205
curriculum	3	0	130
edtech	0	0	4
education	9	0	213
learning	9	0	256
python	49	1	904
schools	5	0	103
students	12	0	287
teachers	4	0	162
teaching	8	0	288

Table A.2: Sentiment Analysis for Education Mailing List using VADER

Sentiment Analysis Using BERT:

Topic	1 Star	2 Stars	3 Stars	4 Stars	5 Stars
Coding	166	19	10	11	1
Curriculum	105	13	8	7	0
Edtech	3	0	0	0	1
Education	191	10	5	13	3
Learning	203	27	16	17	2
Python	776	66	41	62	9
Schools	97	5	4	2	0
Students	236	24	15	21	3
Teachers	134	14	8	9	1
Teaching	247	15	11	22	1

Table A.3: Sentiment Analysis for Education Mailing List using BERT

Sentiment Analysis Using DistilBERT:

Topic	**Negative Sentiment**	**Positive Sentiment**
Coding	174	33
Curriculum	111	22
Edtech	4	0
Education	187	35
Learning	222	43
Python	859	95
Schools	97	11
Students	263	36
Teachers	142	24
Teaching	256	40

Table A.4: Sentiment Analysis for Education Mailing List using DistilBERT

Topic Modeling and Sentiment Analysis Using LDA with Predefined Subtopics:

Subtopic	**VADER Sentiment**	**BERT Sentiment**	**DistilBERT Sentiment**
coding	0.9999	1 star	NEGATIVE
curriculum	0.9978	1 star	NEGATIVE
edtech	0.9999	1 star	NEGATIVE
learning	0.9938	1 star	NEGATIVE
python	0.9988	1 star	NEGATIVE
schools	0.9922	1 star	NEGATIVE
students	0.9998	1 star	NEGATIVE
teachers	0.9851	1 star	NEGATIVE
teaching	0.9996	1 star	NEGATIVE

Table A.5: Sentiment Analysis across Subtopics

Topic Modeling and Sentiment Analysis Using LDA without Predefined Subtopics:

Subtopic	VADER Sentiment	BERT Sentiment	DistilBERT Sentiment
2018 code variable left list turtle right wrote like 2016	0.9998	1 star	NEGATIVE
jan types unicode teachers education work students type ideas 2019	0.9988	1 star	NEGATIVE
lowe mailing distro chalmer summit thu wrote jan 2015 list	0.9922	1 star	NEGATIVE
numbers thonny use students nov concept 2017 projects files project	0.9938	1 star	NEGATIVE
print mailing wrote language using self list def like class	0.9999	1 star	NEGATIVE
use language think jun list data like code 2018 wrote	0.9999	1 star	NEGATIVE
use urner new school using 2016 wrote like code math	0.9996	1 star	NEGATIVE
web unsubscribe email topic migrated new ui mailman mailing lists	0.9851	1 star	NEGATIVE
wrote class jul students teaching use code list like 2017	0.9978	1 star	NEGATIVE

Table A.6: Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT

Main Challenges Discussed in the Mailing Lists

- **Python in Education**: The most frequently discussed topic, with significant positive sentiment. However, there were concerns regarding its integration into curriculums and the support required for teachers.
- **Teaching and Learning**: Discussions centered on effective methods for teaching Python and coding, with some concerns about resources and support for educators.

- **Student Engagement**: Positive discussions about how students engage with Python, but also some negative sentiment about the challenges students face when learning programming.
- **Curriculum Development**: There were discussions about the need for well-structured curriculums that include Python, with concerns about the lack of standardization.
- **Coding in Schools**: Coding was a significant topic, with both positive excitement about its inclusion in education and concerns about the readiness of schools to implement such programs.

Email Mailing Lists

Sentiment Analysis Using Textblob:

Subtopic	**Negative Sentiment**	**Positive Sentiment**
attachments	1	5
compat32	2	6
email	5	17
feedparser	0	3
headers	0	2
memory	1	4
parsing	5	12
policy	2	8
python	5	16

Table A.7: Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob)

Sentiment Analysis Using VADER:

Subtopic	**Negative Sentiment**	**Positive Sentiment**
attachments	1	5
compat32	2	6
email	3	19
feedparser	1	2
headers	0	2
memory	2	3
parsing	2	15
policy	2	8
python	2	19

Table A.8: Sentiment Analysis for Email Mailing List using VADER

Sentiment Analysis Using BERT:

Topic	1 star	2 stars
attachments	4	2
compat32	6	2
email	16	6
feedparser	2	1
headers	1	1
memory	4	1
parsing	12	5
policy	7	3
python	15	6

Table A.9: Sentiment Analysis for Email Mailing List using BERT

Sentiment Analysis Using DistilBERT:

Identified Topic	Negative Sentiment	Positive Sentiment
attachments	6	0
compat32	8	0
email	21	1
feedparser	2	1
headers	2	0
memory	4	1
parsing	17	0
policy	9	1
python	20	1

Table A.10: Sentiment Analysis for Email Mailing List using DistilBERT

Topic Modeling and Sentiment Analysis Using LDA with Predefined Subtopics:

Subtopic	VADER Sentiment	BERT Sentiment	DistilBERT Sentiment
email	0.9848	1 star	NEGATIVE
policy	0.9931	1 star	NEGATIVE

Table A.11: Sentiment Analysis across Subtopics

Topic Modeling and Sentiment Analysis Using LDA without Predefined Subtopics:

Subtopic	**VADER Sentiment**	**BERT Sentiment**	**DistilBERT Sentiment**
encoding right use working way module wed header using issues	0.9892	1 star	NEGATIVE
object old simple raised work uses like msg message line	0.9848	1 star	NEGATIVE
pretty great creating read libraries docs barry post think module	0.9931	1 star	NEGATIVE

Table A.12: Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT

Main Challenges Discussed in the Mailing Lists

The main challenges discussed in the mailing lists include:

- **Handling Large Attachments**: The difficulty of processing large email attachments without consuming too much memory.
- **Compatibility Issues**: The transition from Python 2 to Python 3 brought several compatibility challenges, particularly with email parsing and policies.
- **Memory Optimization**: Strategies to optimize memory usage in email processing, especially with large payloads.
- **Header Parsing**: Complexities in parsing email headers correctly, especially with non-ASCII data.
- **UTF-8 Encoding**: Handling and processing UTF-8 encoded subjects and content within emails.
- **Policy Framework Adjustments**: The need to adjust or rewrite parts of the policy framework to better handle real-world email processing scenarios.

Database Mailing Lists

Sentiment Analysis Using Textblob:

Topic	**Total Mentions**	**Negative Sentiment**	**Positive Sentiment**
Database	172	62	110

Table A.13: Top 10 Most Discussed Topics with Sentiment Distribution (TextBlob)

Sentiment Analysis Using VADER:

Identified Topic	VADER Sentiment (Negative)	VADER Sentiment (Positive)
Database	5	167

Table A.14: Sentiment Analysis for DB Mailing List using VADER

Sentiment Analysis Using BERT:

Subtopic	BERT: 1 Star	BERT: 2 Stars	BERT: 3 Stars	BERT: 4 Stars	BERT: 5 Stars
database	150	4	1	16	1

Table A.15: Sentiment Analysis for DB Mailing List using BERT

Sentiment Analysis Using DistilBERT:

Topic	NEGATIVE	POSITIVE
database	169	3

Table A.16: Sentiment Analysis for DB Mailing List using DistilBERT

Topic Modeling and Sentiment Analysis Using LDA with Predefined Subtopics:

Subtopic	VADER Sentiment	BERT Sentiment	DistilBERT Sentiment
Database	0.998	1 star	NEGATIVE

Table A.17: Sentiment Analysis across Subtopics

Topic Modeling and Sentiment Analysis Using LDA without Predefined Subtopics:

Subtopic	VADER Sentiment	BERT Sentiment	DistilBERT Sentiment
2016 accept connection added unicode changes http database binary python	0.4939	1 star	NEGATIVE
26 different thu using driver support like use jul 2018	0.872	1 star	NEGATIVE
implement 2022 wrote lemburg attribute database connection python https autocommit	0.9779	1 star	NEGATIVE
mar support wrote version 2022 fri python 28 oct https	0.9022	1 star	NEGATIVE
new oracle issues module number database pip release python https	0.7845	1 star	NEGATIVE
output procedure sets sequence input procedures return stored result parameters	-0.6486	1 star	NEGATIVE
python dec 2022 numbers order paramstyle apr executemany numeric https	0.998	1 star	NEGATIVE
sql product egenix connect server odbc database python http mxodbc	0.9926	1 star	NEGATIVE
tests classes phd use oleg python person https http sqlobject	0.5267	1 star	NEGATIVE
tuples interfaces wrote services api lemburg async database python http	0.9325	1 star	NEGATIVE

Table A.18: Sentiment Analysis of Subtopics using VADER, BERT, and DistilBERT

Main Challenges Discussed in the Mailing Lists

- **Database Handling**: The discussions revolved around the challenges of database management, including performance issues, integration with Python, and possibly the adoption of various database interfaces like SQLObject and mxODBC. The negative sentiment reflects concerns and difficulties experienced by the developers in these areas.