Constructions 17 (2025)
doi10.24338/cons-xxx

Political Sentiment and Community Analysis

Hriscu Alexandru-Gabriel, Lazurcă Samuel-Ionuț, Nastasiu Stefan, Nestor Maria, Rotaru Florin-Eugen
Alexandru Ioan Cuza, University of Iasi

Abstract

This project aims to analyze and visualize political sentiment and polarization on social media. The system uses sentiment analysis and topic modeling to identify the political alignment of posts and group users based on their opinions. Community detection algorithms map polarized groups, showing how different political communities interact. The results are displayed as interactive visualizations with color-coded clusters representing communities and lines indicating the strength of interaction between them. This tool helps researchers, policymakers, and others better understand political divisions and interactions in online spaces.

1 Introduction

Our application is a real-time dashboard that fetches live social media data (e.g. from Reddit) and analyzes sentiment in posts and comments. The results will be displayed on an interactive graph where nodes represent users and edges represent interactions (likes, replies, mentions, etc.).

The interest in sentiment analysis and community analysis has increased recently due to upcoming political events, an example being the last year's elections in the USA. Due to the influence offered by the media networks, political campaigns use the resources offered by the online environment such as posts and debates on different platforms (e.g. **Reddit**).

Therefore, a very good indicator of the effectiveness of a political campaign is the way in which the users of social media networks perceive a political party or a candidate. To determine political sentiment, the following can be taken into account: the way people respond to a post, observing the number of likes and comments, what kind of comments are at the top of a post and how often a post is shared. Also, the social media post of a candidate can decisively change the political sentiment, for example when he makes a mistake or the way he responds to certain controversies (for example illegal immigration).

Corresponding author(s): Hriscu Alexandru-Gabriel, alexandru.hriscu@student.uaic.ro Lazurcă Samuel-Ionuț, samuel.ionut.lazurca@gmail.com Nastasiu Ștefan, UAIC, stefan.nastasiu@student.uaic.ro Nestor Maria, maria.nestor@student.uaic.ro Rotaru Florin Eugen, florin.rotaru@student.uaic.ro

2 Related work

The field of sentiment analysis has evolved significantly, marked by foundational contributions that have shaped its current methodologies and applications. An example of an application that is useful in the analysis of political sentiment at the level of social media is Crimson Hexagon (Brandwatch) which, although it is also used for other purposes such as following trends in a market space, can be used for politics, therefore being useful for the analysis of political sentiment during the Brexit period (Brandwatch 2017).

Another popular application is Talkwalker (2025), which offers social listening, media monitoring and social benchmarking services. In addition, Facebook has developed a tool called Crowdtangle (2024), which isn't used anymore, to see what kind of posts go viral. It was based on the way the posts are distributed, and its purpose was to identify influencers who commented on a certain topic, while also analyzing the polarization of communities. In addition, it was used to analyze the spread of fake news.

Recent academic projects have demonstrated the effective use of machine learning models to analyze political sentiment and polarization on social media. Research from universities has focused on the application of natural language processing (NLP), sentiment analysis, and community detection in order to map out polarized political communities.

A study by Zhang et al. (2024) explored various approaches to sentiment analysis on Reddit comments related to the 2019 Indian General Elections, leveraging advanced deep learning architectures including Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. Their objective was to gain comprehensive insights into public opinion dynamics and sentiment trends surrounding political events discussed on social media platforms like Reddit. To achieve this, they conducted a series of experiments employing different model architectures, namely CNN, LSTM, LSTM + CNN, and CNN + LSTM. Their experimental results revealed that the hybrid CNN + LSTM model achieved superior performance in sentiment classification, consistently outperforming the individual CNN and LSTM models, obtaining a Macro F1-Score of 0.84.

A study conducted by Marozzo & Bessi (2017) on analyzing political sentiment using social media data describes how posts are categorized based on specific keywords that signal political alignment. These keywords are pre-processed and classified into factions, allowing the system to detect user polarization and predict trends during election campaigns. This research emphasizes using keyword-based approaches in combination with advanced algorithms like multilayer perceptrons (MLPs) and convolutional neural networks (CNNs) to improve sentiment classification and topic modeling. These methodologies help map out the communities of users with distinct political preferences by examining their engagement and interaction patterns on platforms like Twitter or Facebook.

In another project by Belcastro et al. (2019) from an European university, researchers applied community detection algorithms to group users by their interaction networks and identified polarized clusters based on their political senti-

ment. These clusters were then visualized using color-coded graphs that show the strength of interaction between users aligned with different political opinions. These visual tools can be helpful for political scientists to better understand the evolving polarization during important political events, such as elections.

3 Methodology

Our solution is a Django application that uses Reddit posts and comments to create interactive graphs showing political sentiment and social media communities. These graphs are powered by AI models specifically trained on political news for accurate insights.

The user interface is built with ReactJS and Vite, providing an easy-to-use dashboard where users can view, interact with, and save graphs.

We combine several technologies to make this possible. AI models analyze the similarity of Reddit posts, perform sentiment analysis on comments, and detect topics in social media discussions. For creating the graphs, we detect communities based on similarity. The app also uses scheduled tasks to regularly update the data and graphs, keeping everything fresh.

All user information and data used to generate the graphs are securely stored in the Cloud for convenience and scalability.

The following subsections will explain the graph visualization and AI models like Sentence-BERT, LDA, RoBERTa, and LSTM.

3.1 Visualization

3.1.1 Drawing the Graphs

Creating the visualizations is achieved in two phases.

1. **Phase 1 - Creating the graph of posts using the metadata**

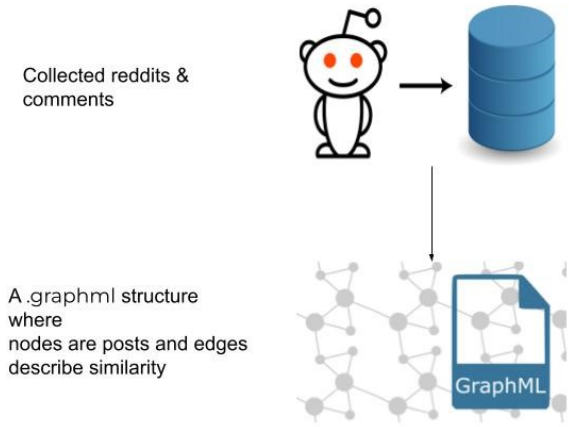

Figure 1: Database scheme

Given a desired topic, we collect the relevant posts from the database. We then create a graph called graph of posts similarity where nodes are reddit

posts and the edges between them are weighted with respect to their similarity. This similarity is obtained by analyzing the cosine similarity of the posts embeddings (details in the next section). Once we have a weighted graph, we detect post communities using a community detection algorithm that relies on edge weights, such as the Louvaine method (Traag et al. 2019). The resulting graph, which is represented in a xml-like format `.graphml`, contains the needed attributes of nodes, such as its community, the title, etc.

2. **Phase 2 - Visualizing the Graph of Post Similarity**
   The drawings are created using the **Kamada-Kawai** (Kamada et al. 1989) layout, a subclass of Force Directed layouts. These work by modeling the nodes and edges as objects with physical properties: attraction via edges (springs) and repulsion via nodes (particles). This class of algorithms reduces a graph drawing to an optimization problem in which the natural physical placement is desired. This will also reduce the edge crossings and produce an expressive visualization (easy to read). For example, if we weight the edges with respect to the post similarity, we can expect to obtain something like the Hairball graph. We provide two different visualizations (drawings) of the graph.

   - ***Hairball* Graph**: A general overview of the similarity graph, showing how posts are connected and also how the communities are represented, being represented by colors:

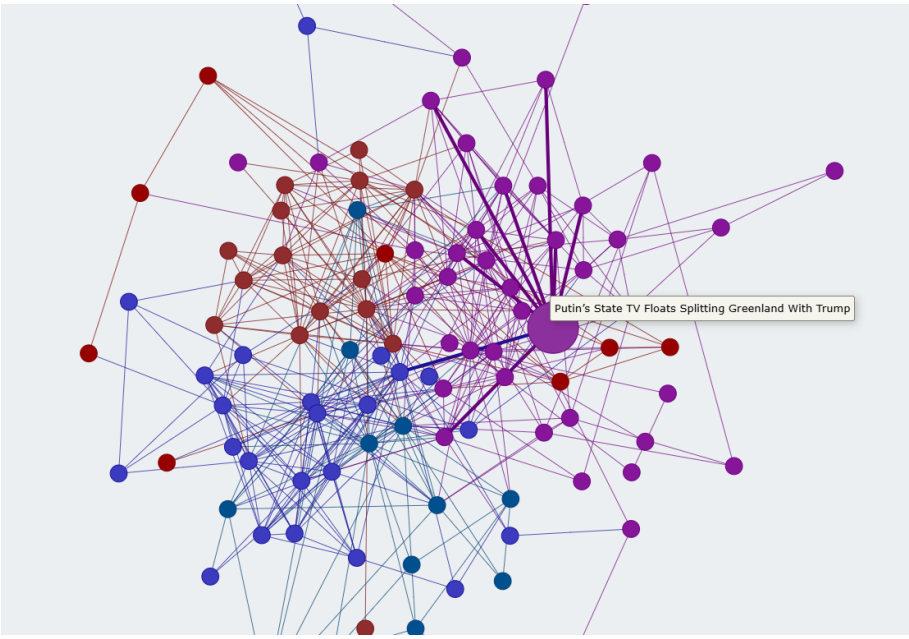

Figure 2: Haibrall Graph drawing representing posts for a topic.

Ideally, the distances between posts represent their similarity. However achieving a well-balanced representation with that constraint is a non-trivial optimization problem.

- **Community Stars**:Also obtained with Kamada-Kawai algorithm. We represent each community as a cluster of posts and a hub-node.

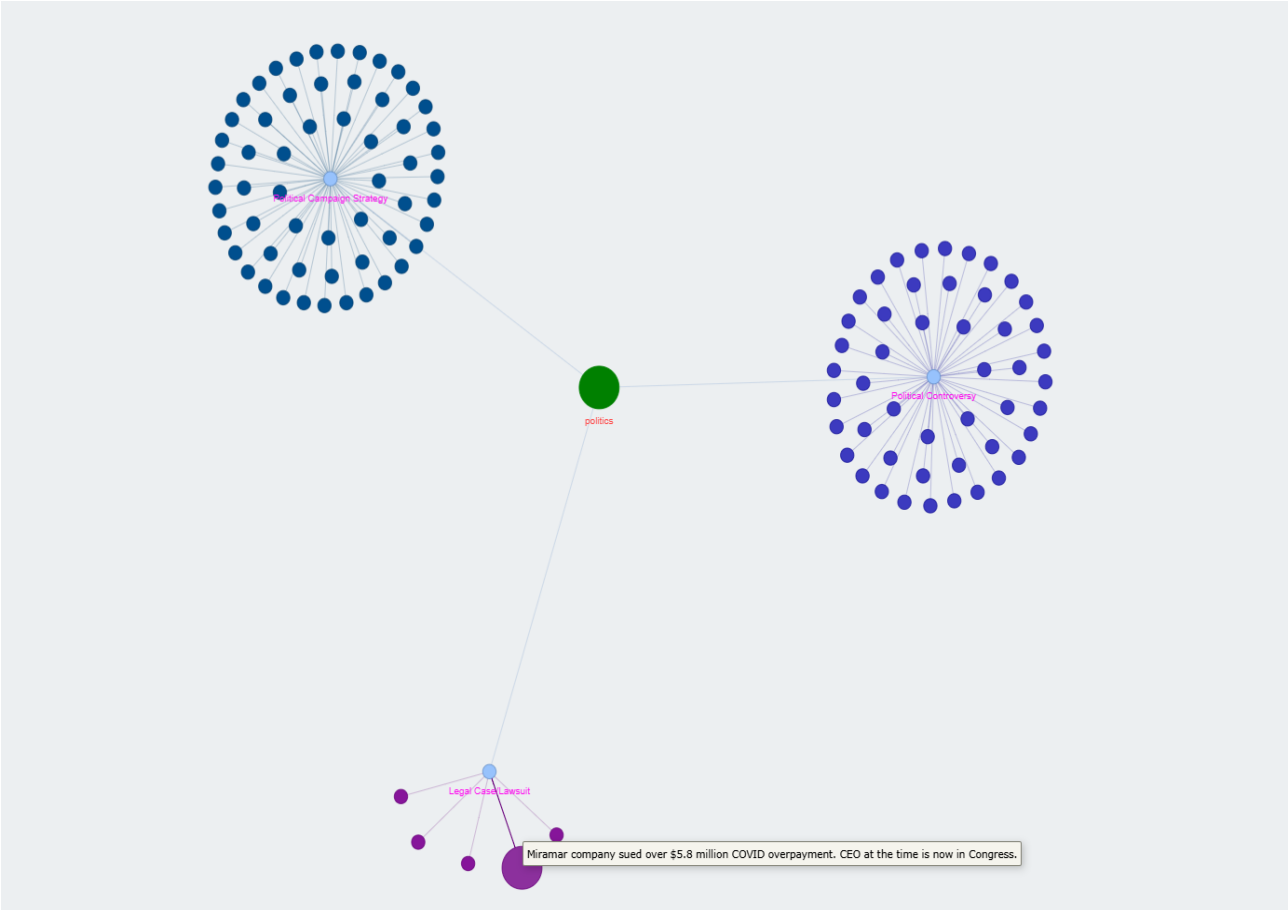

Figure 3: Graph drawing representing communities of posts for a topic.

This approach helps identify patterns and relationships within large sets of text data, providing insights into how different posts within a topic relate to each other.
We not only detect communities, we also describe them, using topic modeling (see section Topic Modeling for details).
We aimed to automatically annotate each cluster with the topic that best describes the community. Below is an example of a cluster on the topic *Legal Case*.

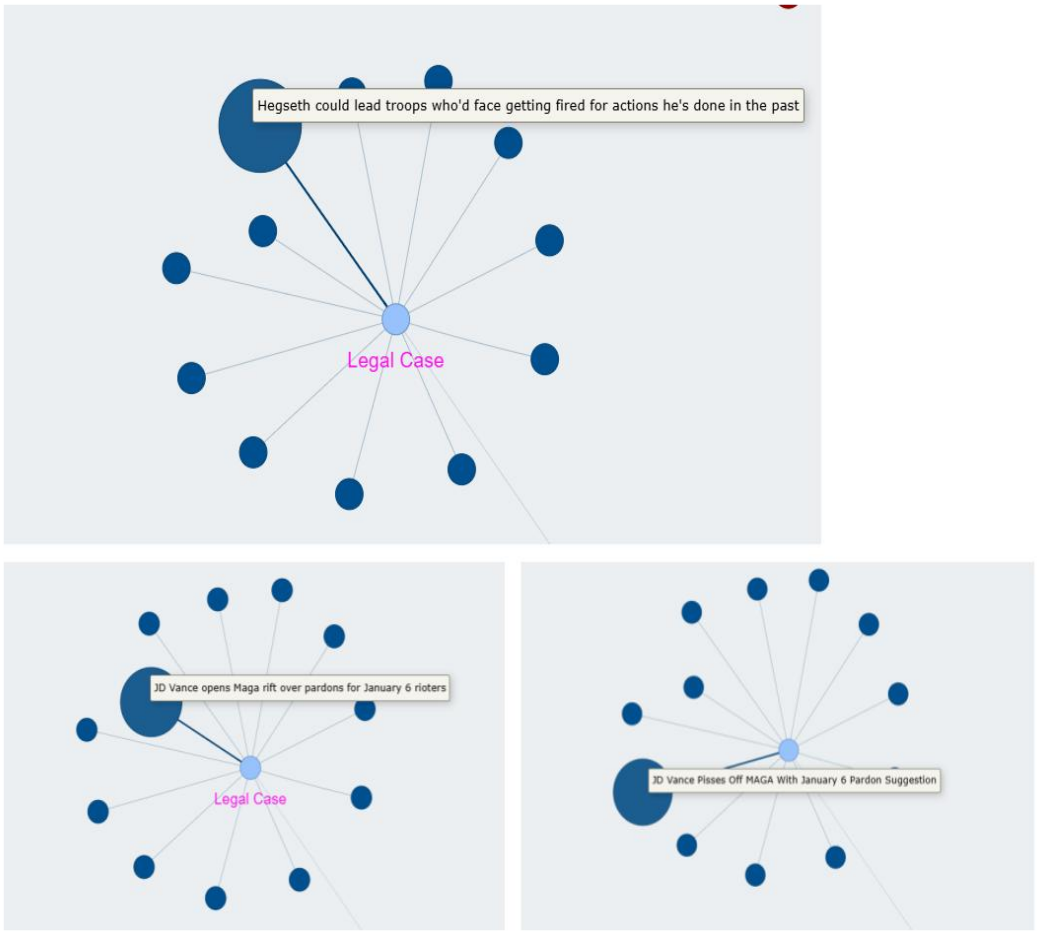

Figure 4: Graph drawing representing communities of posts for a topic.

(*side note:* the colors for the communities and for the background have been specifically selected to achieve high contrast for readability)

3.1.2 Interactions

One important aspect of our visualization is that they are interactive. The main element is **visualizing comments of the posts**. By clicking on a post, the user is presented a drawing of the clustered comments - positive, negative and neutral comments. The predominant group is easy to spot - it is bigger in size.

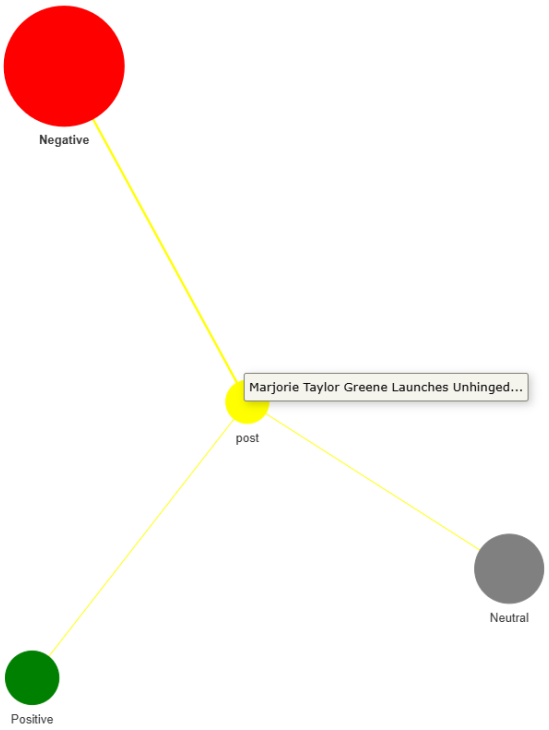

Figure 5: Graph drawing representing comments for posts.

Which are expandable and collapsible.

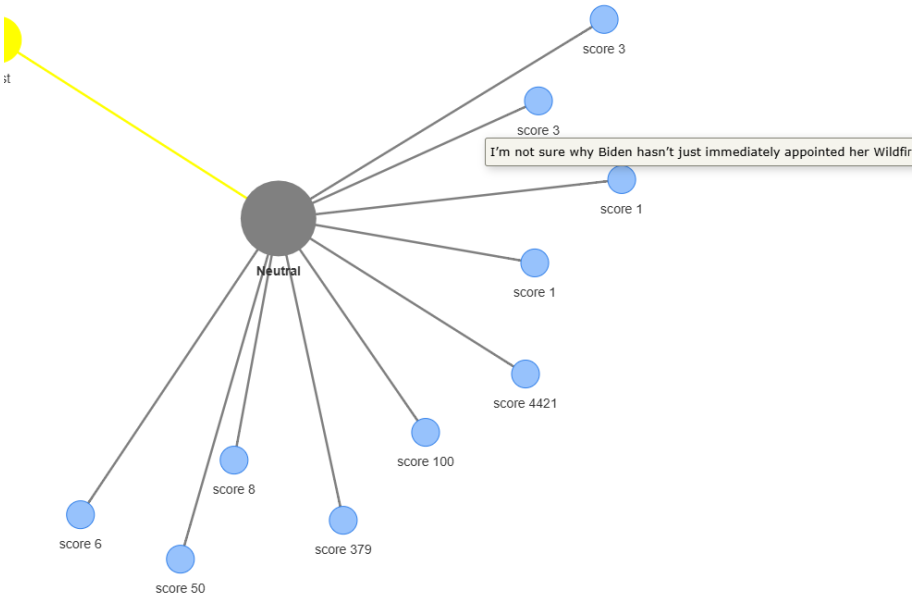

Figure 6: Graph drawing representing comments for posts - expanded.

Other ways of interacting are zooming-in, hovering the nodes and edges (which highlights them), dragging and dropping the nodes.

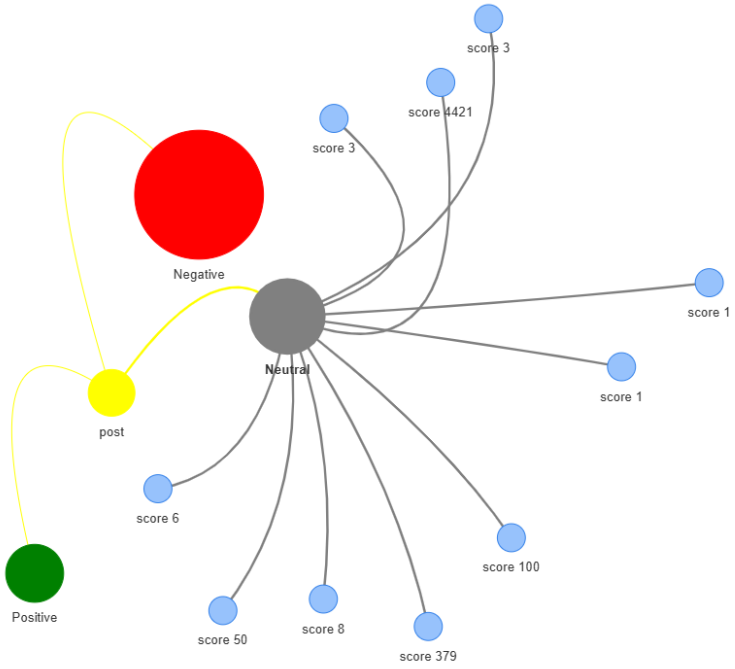

Figure 7: Interacting with the graph drawings.

The comments graphs and their respective graph drawings are generated using the same algorithms as the post comments - Kamada-Kawai force directed algorithm. For details regarding the sentiment detection, refer to the Sentiment analysis section.

3.2 Post Similarity

The **Sentence-BERT** model plays a crucial role in the community detection process by calculating the similarity between Reddit posts. Each post in a given topic is treated as a node in a graph, and the similarity scores between posts are used as the weights of the edges connecting the nodes. The graph is then processed using a community detection algorithm to identify clusters of related posts.

The SentenceTransformer model used is all-MiniLM-L6-v2, a lightweight and efficient variant of BERT optimized for semantic similarity tasks. The model computes embeddings (dense vector representations) for each post's text. These embeddings capture the semantic meaning of the text, enabling effective comparison of post content. The similarity between the embeddings is calculated using the cosine similarity metric, resulting in a similarity matrix. The matrix contains pairwise similarity scores for each pair of posts within a topic, scaled and stored in a dictionary for graph creation.

3.3 Topic Modeling

Topic Modeling is a statistical technique used for discovering the abstract topics that occur in a collection of documents. It is an unsupervised learning ap-

proach, meaning it does not require labeled data, making it particularly useful for exploratory analysis. In this project, topic modeling was used to identify key themes in Reddit posts, helping to analyze discussions and trends related to political sentiment and polarization.

Latent Dirichlet Allocation (LDA) is the most common probabilistic approach for topic modeling. The documents are described as a bag-of-words, and the idea of LDA is that documents are represented as random mixtures over latent topics, and each topic is characterized by a distribution over words Blei et al. (2003). This algorithm creates clusters of words that are most probably to represent the same topic.

Although alternative algorithms such as Non-negative Matrix Factorization (NMF) and BERTopic exist, LDA was chosen for this project because it is widely used and well documented, making it a reliable and accessible choice. It is robust for smaller datasets, such as the Reddit posts used in this analysis. Also, it generates interpretable results, which is important for understanding and explaining the extracted topics.

3.3.1 Implementation Details

1. **Dataset**: The model was developed using data collected from the r/politics Subreddit, including posts from 2022-2023 and the latest posts from 2025. Since the Reddit API limits data collection to the most recent 100 posts, historical data from 2022-2023 was sourced via Pushshift.io (2025).
2. **Preprocessing**: Normalization, tokenization, and lemmatization ensures consistency and improves the quality of topic modeling. Additionally, bigrams and trigrams were generated to capture meaningful word combinations. Frequent and uninformative words were then removed from the corpus, resulting in a refined dataset for analysis.
3. **Gensim LDA**: The Latent Dirichlet Allocation (LDA) model was implemented using Gensim, selected for its simplicity and robust integration with Python.
4. **Naming Topics**: Google Gemini's API was utilized to generate descriptive names for the identified topics.
5. **Visualization**: The pyLDAvis library was used to create interactive and intuitive visualizations of the topics, enabling in-depth exploration of the results.

3.4 Sentiment Analysis

Sentiment analysis is a natural language processing (NLP) technique used to determine the sentiment conveyed in a piece of text. It categorizes the sentiment as positive, negative, or neutral and is widely applied in fields such as social media monitoring, customer feedback analysis, and market research. Sentiment analysis involves extracting subjective information from text to gauge public opinion and emotional tone.

This project implements two sentiment analysis models: a Long Short-Term Memory (LSTM) deep learning model and a pretrained RoBERTa-BERT model.

Both approaches have distinct advantages and limitations, offering a balance between resource consumption and accuracy. Several other architectures (such as *CNN*, *CNN-LSTM* or a custom *BERT*) were trained, but the LSTM and the pretrained RoBERTa-BERT models had the best performance out of all.

3.4.1 LSTM Model

The Long Short-Term Memory (**LSTM**) model is a type of recurrent neural network (RNN) well-suited for sequence data, such as text. LSTMs can capture long-term dependencies in text, making them a good choice for sentiment analysis.

Key Components:

- **Framework**: The model was implemented and trained using the PyTorch framework, which provides flexibility and efficient handling of deep learning models.
- **Dataset**: The model was trained using the Sentiment140 dataset, which contains labeled tweets for sentiment analysis.
- **Embeddings**: A Word2Vec model was trained to generate word embeddings. These embeddings convert words into dense vectors that capture semantic meaning and are used as input to the LSTM.
- **Preprocessing**: Before training, the input text underwent a comprehensive preprocessing pipeline to normalize and clean the data. The key preprocessing steps included:
  - **Lowercasing:** Converting text to lowercase for uniformity. Removing Mentions and URLs: Stripping out Twitter mentions and links to reduce noise.
  - **Hashtag Extraction:** Splitting hashtags into meaningful words (e.g., #HappyDay ”Happy Day”).
  - **Abbreviation Expansion:** Converting common abbreviations (e.g., ”brb” → ”be right back”) using a predefined dictionary.
  - **Emoji Conversion:** Translating emojis into descriptive text (e.g., ”:grinning_face:”).
  - **Removing Special Characters:** Eliminating non-alphanumeric characters to simplify tokenization.
  - **Tokenization:** Breaking text into individual word tokens.
  - **Optional Stopword Removal:** Filtering out common stop words like ”the” and ”and” if required.
  - **Lemmatization:** Converting words to their base forms to reduce word inflection (e.g., ”running” → ”run”).
  - **Architecture:** The LSTM model includes a bidirectional LSTM layer, allowing it to capture contextual information from both past and future tokens in a sequence. The architecture comprises:
    - An embedding layer to incorporate the Word2Vec embeddings.
    - A bidirectional LSTM layer to handle sequential dependencies.
    - A fully connected output layer to produce sentiment predictions.
  - **Training:** The model was trained for 30 epochs with a batch size of

64, using Binary Cross-Entropy Loss to optimize predictions. Xavier initialization was employed to improve the convergence of the model.

Performance Metrics Visualization:
During training, the model's performance was tracked using loss and accuracy metrics for both the training and validation datasets. The following graphs were generated to visualize the LSTM model's performance:

- **Loss Plot:** The graph displays the training and validation loss across epochs, indicating how well the model learns over time and whether it overfits the training data.
- **Accuracy Plot:** The graph shows the training and validation accuracy, illustrating the model's ability to correctly classify sentiment as training progresses.

The graphs help evaluate the model's learning dynamics and fine-tune hyperparameters to improve performance.

These visualizations were generated using the plot_metrics() function, which plots both Loss vs. Epochs and Accuracy vs. Epochs side by side for easy comparison.

3.5 RoBERTa-BERT Model

The RoBERTa-BERT model is a transformer-based architecture that leverages deep contextualized word representations. It is pretrained on a large corpus and fine-tuned on specific tasks, including sentiment analysis.

Key Components:

- **Pretrained Model:** The RoBERTa-BERT model is a variant of BERT (Bidirectional Encoder Representations from Transformers) optimized for better performance on downstream tasks.
- **Transfer Learning:** The model benefits from transfer learning, where a pretrained model is adapted to a new task with minimal additional training.
- **Architecture:** The model uses self-attention mechanisms to capture complex relationships in text, making it highly effective for understanding sentiment context.
- **Preprocessing:** To ensure compatibility with the pretrained RoBERTa-BERT model, a custom preprocessing class, BertPreprocessor, was implemented. This class handles text formatting specific to Reddit-style data and converts it into a format compatible with BERT-based models.

Preprocessing Steps for BERT:
The following preprocessing steps were applied to prepare text for BERT:

- Placeholder Replacement: Reddit-specific placeholders were converted to BERT-compatible placeholders. For example:
  - u/username → @user
  - Links starting with http or www were replaced with a generic http

token.
  - Subreddit mentions like r/askreddit were replaced with a subreddit token.

Advantages:

- **Contextual Understanding:** It captures nuanced language patterns and understands context at a deeper level.

Limitations:

- **Resource Intensive:** The model requires more computational resources.

The RoBERTa-BERT model provides a more accurate sentiment analysis solution but requires a powerful GPU for efficient training and inference. We chose the RoBERTa-BERT model as our primary sentiment analysis method because we have the resources necessary and because we want to benefit from the greater accuracy. But changing to LSTM analysis is possible anytime, in case it is needed.

4 Results

4.1 LDA Model Evaluation

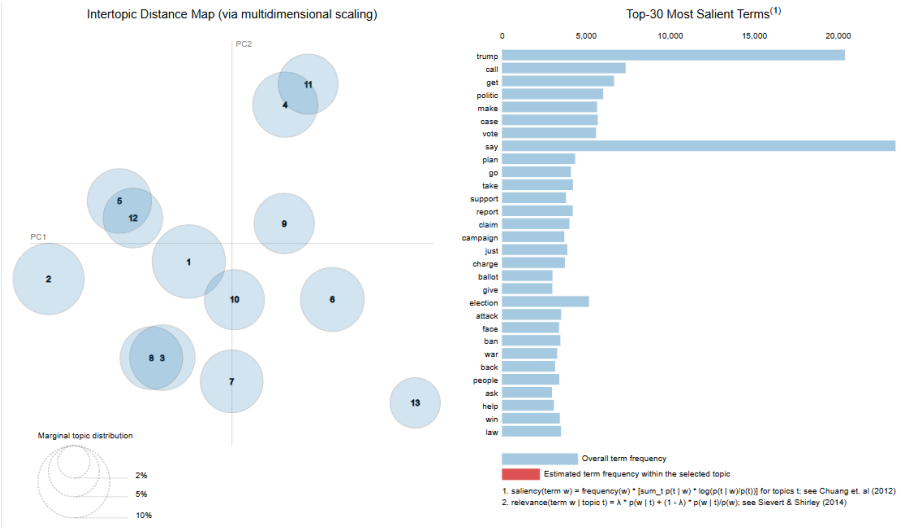

Figure 8: Topics graph

1. **Visual Interpretations**: pyLDAvis visualizations were analyzed to interpret the quality of topics.
2. **Logical Grouping:** Words within each topic were reviewed to ensure they were logically related.
3. **Subjective Nature:** While manual and visual evaluations are subjective, they are necessary to refine the results and validate the coherence of topics.

4.2 The Use of AI for Unit Testing

AI played a significant role in the testing process of our application, particularly in automating the generation of unit tests and assisting with documentation for technologies we were not yet familiar with, such as Django, MongoDB, and Locust. This allowed us to efficiently write and organize tests, ensuring that we covered critical areas of the application.

One of the key achievements was reaching over 80% test coverage for the Base API. This high coverage was essential in ensuring the robustness of the core functionalities of the application. AI tools helped streamline the creation of test cases, automatically identifying potential edge cases and suggesting improvements for our test suite. However, we did encounter some limitations. While AI assisted in generating useful test cases, there were instances where the AI produced code that resulted in runtime errors, requiring manual debugging and intervention.

Despite the advantages, there were also notable downsides to relying on AI for testing. One issue was that the AI-generated tests sometimes lacked precision or failed to fully account for edge cases, leading to incomplete test coverage in certain scenarios. Furthermore, the AI often suggested test structures that were not optimal or appropriate for the specific context of the application, resulting in additional time spent reviewing and refining the generated tests.

Another downside was the AI's occasional inability to generate fully compatible or error-free code. This required manual adjustments to ensure that the generated tests integrated properly with the rest of the codebase. Additionally, the AI sometimes struggled with understanding complex business logic or domain-specific nuances, which affected the accuracy and relevance of the generated tests.

While AI was helpful in generating tests, especially when dealing with unfamiliar frameworks and languages, it wasn't always reliable in producing optimized or error-free scripts. Despite these challenges, AI played an important role in ensuring that the testing process remained thorough and efficient, contributing to a solid foundation of automated tests. The ability to quickly generate tests and documentation helped us focus on improving the functionality and performance of the application, even if some manual intervention was needed to address the limitations of AI-generated content.

4.3 Performance and Stress Testing

To evaluate the performance and reliability of the application under different levels of load, we conducted stress tests using Locust. These tests were run with a local database and focused on three API endpoints: `GET /api/v1/drawings`, `GET /api/v1/drawings?name={drawing}`, and `GET /api/v1/favourites`.

The tests measured key metrics such as the number of users, user spawn rate, the total number of requests, and the number of failures. Below are the results of these tests, presented in tabular format:

4.3.1 Results for `GET /api/v1/drawings`

Number of Users	Spawn Rate	Total Requests	Failures
100	10	1026	0
500	20	1559	50
1000	50	1065	345

Table 1: Stress Test Results for `GET /api/v1/drawings`

4.3.2 Results for `GET /api/v1/drawings?name={drawing}`

Number of Users	Spawn Rate	Total Requests	Failures
100	10	1001	0
500	20	1399	45
1000	50	1115	359

Table 2: Stress Test Results for `GET /api/v1/drawings?name={drawing}`

4.3.3 Results for `GET /api/v1/favourites`

Number of Users	Spawn Rate	Total Requests	Failures
100	10	1089	1
500	20	1566	65
1000	50	812	351

Table 3: Stress Test Results for `GET /api/v1/favourites`

4.3.4 Performance tests conclusions

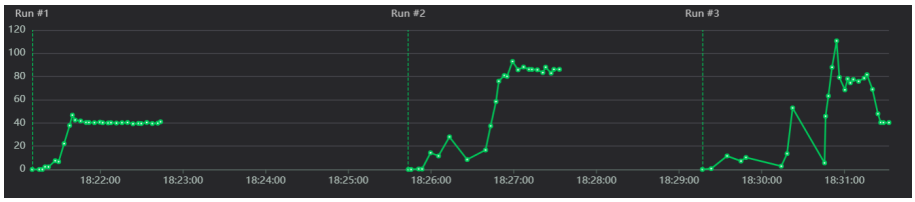

Figure 9: Failures graph

During testing of the `POST /api/v1/login` route, it was observed that the percentage of failures increased significantly as the number of users scaled up. The primary reason for this was the reliance on a remote Neon Postgres database, which is limited in capacity and prone to higher latency compared to a local database. The most common cause of failure was **Connection Refused**, which occurred when the database reached its resource or connection limits.

This limitation highlights the challenges of using a remote database for high-traffic routes and emphasizes the importance of choosing a database solution that is robust and scalable enough to handle increased load. Addressing these issues,

such as by upgrading the database's capacity or optimizing connection handling, would be critical for improving the application's performance under stress.

When delivering graphs, the choice of database significantly impacts response times. When graphs are saved in a local database, their retrieval is nearly instantaneous due to the minimal latency and high-speed access of the local environment. In contrast, using the free tier of MongoDB Atlas, which operates as a remote database with limited resources, can introduce substantial delays. The additional network latency, coupled with the performance constraints of the free tier, can result in delivery times that are up to a minute longer compared to the local database. This difference underscores the importance of selecting a database solution that aligns with the application's performance requirements, especially for features like graph delivery where speed is crucial for user experience.

4.4 Choosing the Right Approach for Data Refreshing

Scheduling background tasks is a critical part of our application, handled using Celery Beat. We have two main tasks: one for the Data Collector and another for the Base API.

The Data Collector is a specialized application designed for Reddit data collection. It retrieves posts and comments from political sub-reddits and processes them using sentiment analysis models before inserting the data into a MongoDB database hosted in the Cloud. This ensures that the stored data is not only up-to-date but also enriched with sentiment insights, making it immediately usable for further analysis and visualization.

The Base API task takes this processed data and generates updated graphs. These tasks are scheduled to run at regular intervals, such as hourly, with customizable timing.

This approach was chosen to minimize user wait times. Performing live data extraction, running sentiment analysis, or generating large HTML graph files on demand would be computationally expensive and slow. By processing and storing the data in the background and refreshing it periodically, we maintain an efficient, responsive, and resource-optimized application.

There is, however, an instance where the HTML files are generated on demand - the graph of comments of a post. This is because it isn't very clever to generate comment graphs for ALL posts, as there might be a lot of them. This is why we generate the graph and the drawing when the post is inspected. However, once these have been generated, they are persisted to the database and the next time they are needed, less time will be necessary for their retrieval.

5 Future Works

Several improvements could significantly enhance the functionality and user experience of the application. One key area is the expansion of topics for which graphs are generated. While the current focus is on political sentiment, the application could cover additional topics like economics, health, climate change, or

trending social issues. This broader scope would attract a wider audience and make the tool more versatile.

Reducing the delivery time for graphs is another critical improvement. Implementing advanced caching methods, such as in-memory caches like Redis or CDNs, could store pre-generated graphs and data, ensuring faster response times. This would greatly enhance the user experience, especially when dealing with larger datasets.

The application could also benefit from new features, including enhanced visualizations like heat-maps or timelines and deeper statistical insights such as trends or engagement metrics. Adding user subscriptions for topic updates or integrating live data capabilities could further enrich the application. Deploying the application in a robust cloud environment with scalability measures like containerization and load balancing would ensure its reliability under increased traffic.

Additionally, refining the AI models used for sentiment analysis and topic detection could improve accuracy, especially by leveraging newer algorithms or transfer learning techniques. Enhancing the user interface, adding multilingual support, and introducing community-oriented features like data sharing or collaborative dashboards would also make the application more engaging and accessible.

These updates would not only improve the application's performance and scalability but also extend its functionality, making it a more powerful and user-friendly tool for analyzing and visualizing social media dynamics.

6 Conclusions

In conclusion, our application provides a unique and interactive way to visualize public sentiment and community divisions on social media, particularly in the context of politics. By leveraging AI-driven analysis and graph visualization, it enables users to gain valuable insights into how opinions are formed and shared within social networks. This foundation not only showcases the potential for deeper understanding of political discourse but also highlights the broader applicability of such tools in analyzing other topics of interest.

With further development, including the integration of additional features such as expanded topic coverage, enhanced visualizations, and improved scalability, the application has the potential to evolve into a comprehensive industrial-level solution. By incorporating advanced analytics and diversifying the range of subjects analyzed, it could serve as a powerful tool for businesses, researchers, and policymakers alike.

Ultimately, this application lays the groundwork for a scalable, multi-purpose platform that transforms how we interpret and interact with the vast and dynamic data generated by social media. It underscores the importance of leveraging technology to make sense of complex social dynamics and offers a promising avenue for future innovation.

License

References

Belcastro, Loris, Riccardo Cantini, Fabrizio Marozzo, Domenico Talia & Paolo Trunfio. 2019. Discovering political polarization on social media: A case study. In *2019 15th international conference on semantics, knowledge and grids (skg)*, 182–189. doi:10.1109/SKG49510.2019.00038.

Blei, David M., Andrew Y. Ng & Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3(null). 993–1022.

Brandwatch. 2017. Brexit data: Post-truth politics and the eu referendum. https://www.brandwatch.com/blog/react-brexit-post-truth/.

Crowdtangle. 2024. Crowdtangle dataset on meta transparency center. https://transparency.meta.com/ro-ro/researchtools/other-datasets/crowdtangle/.

Kamada, Tomihisa, Satoru Kawai et al. 1989. An algorithm for drawing general undirected graphs. *Information processing letters* 31(1). 7–15.

Marozzo, Fabrizio & Alessandro Bessi. 2017. Analyzing polarization of social media users and news sites during political campaigns. *Social Network Analysis and Mining* 8. doi:10.1007/s13278-017-0479-5.

Pushshift.io. 2025. Pushshift.io: A reddit search api. https://pushshift.io/.

Talkwalker. 2025. Talkwalker - leading consumer intelligence platform. https://www.talkwalker.com/.

Traag, V. A., L. Waltman & N. J. van Eck. 2019. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports* 9(1). doi:10.1038/s41598-019-41695-z. http://dx.doi.org/10.1038/s41598-019-41695-z.

Zhang, Ning, Jize Xiong, Zhiming Zhao, Mingyang Feng, Xiaosong Wang, Yuxin Qiao & Chufeng Jiang. 2024. Dose my opinion count? a cnn-lstm approach for sentiment analysis of indian general elections. *Journal of Theory and Practice of Engineering Science* 4. 40–50. doi:10.53469/jtpes.2024.04(05).06.