Smart News Summarizer & Analyzer

Arsenii Ahamalov

CVUT–FIT

ahamaars@fit.cvut.cz

December 18, 2025

1 Goals

The goal of this project is to create an application that can do the following: 1. Download current articles from selected news APIs (with support for local caching). 2. Perform text preprocessing (cleaning, tokenization, stop word removal, lemmatization). 3. Generate concise summaries using the TextRank method. 4. Group articles into thematic clusters using TF-IDF and K-Means clustering. 5. Visualize results (topic distribution, keywords, wordclouds). 6. Provide an interactive GUI in the form of a Streamlit dashboard. The application enables users to quickly get an overview of a larger set of news articles, identify the main topics, and read short summaries instead of full texts. Before reading this report, I recommend watching the short dashboard demo (GIF/video) included in the project materials to get an intuitive feeling for the user interface and workflow.

2 Input and Output

2.1 Input

The system expects articles from the Guardian Open Platform API. Articles can be fetched using various filters: search query, section (e.g., "politics", "sport"), and date range. The API returns articles in JSON format, which are then parsed and stored locally in CSV format for caching and offline access.

2.2 Output

The processed data is saved in CSV format (data/processed/articles_with_topics.csv) containing all article metadata, cleaned text, summaries, and assigned topic labels. Topic modeling metadata is saved as JSON (data/processed/topic_modeling_metadata.json) including cluster parameters, keywords per topic, and silhouette scores. Visualizations are generated as PNG images in data/processed/plots/.

2.3 Dashboard

The Streamlit dashboard provides interactive visualization and exploration. Users can browse articles by topic, view summaries, explore visualizations, search and filter articles, and download processed data. The dashboard also allows downloading new articles from the API and processing them through the complete pipeline.

3 Processing Pipeline

3.1 Processing Steps

The original articles are processed with multiple sequential operations: 1. Article fetching – Download articles from Guardian API and parse JSON response. 2. Text preprocessing – Remove HTML tags, normalize text, compute statistics. 3. NLP processing – Tokenization, stop word removal, lemmatization using NLTK and spaCy. 4. Summarization – Generate summaries using TextRank algorithm (10% of original length). 5. Topic modeling – TF-IDF vectorization, PCA dimensionality reduction, K-Means clustering. See an example of the processing results in figures 1 to 4.

3.2 Article Fetching

Articles are retrieved from the Guardian Open Platform API using the requests library. The system supports filtering by section, query keywords, and date range. Fetched articles are parsed to extract title, content, author, publication date, and URL. Data is stored locally in CSV format for persistent storage and caching.

3.3 Text Preprocessing

Text cleaning removes HTML tags, special characters, and normalizes whitespace. The clean_text() function converts text to lowercase and removes punctuation. Processed text is stored alongside original content. Statistics including text length and word count are computed for each article.

3.4 NLP Processing

Tokenization splits text into words using NLTK's word_tokenize(). Stop words are removed using a comprehensive list (NLTK stopwords plus custom additions for news articles). Lemmatization uses spaCy's en_core_web_sm model to reduce words to their base forms. The preprocess_text_pipeline() function combines all steps, returning processed tokens ready for further analysis.

3.5 TextRank Summarization

TextRank builds a sentence similarity graph using Jaccard similarity between preprocessed sentence tokens. PageRank algorithm (implemented via networkx) ranks sentences by importance. Sentences are filtered to remove very short ones, timestamps, and commentary fragments. Summaries are generated as 10% of original text length with a minimum threshold of 7-8% to ensure quality. The filter_and_score_sentences() function improves readability by preferring longer, well-formed sentences.

3.6 Topic Modeling

TF-IDF vectorization uses sklearn's TfidfVectorizer with parameters: min_df=3 (word must appear in at least 3 articles), max_df=0.7 (exclude words in more than 70% of articles), max_features=1000 (limit vocabulary size). PCA reduces dimensionality before clustering for stability. K-Means clustering uses sklearn with n_init=50, max_iter=1000, random_state=42, and algorithm="elkan" for reproducibility and stability. During experimentation, several values of k were evaluated on validation data; for the final semester model we fixed k=7 topics to obtain a small number of interpretable, broad themes while keeping the clusters reasonably separated. Keywords are extracted from cluster centers in the original TF-IDF space by identifying terms with highest TF-IDF weights for each cluster.

> The extracted keywords help identify the content and theme of each topic. Figure 1 shows the top keywords for each topic, while Figure 2 displays a heatmap showing keyword intensity across topics.

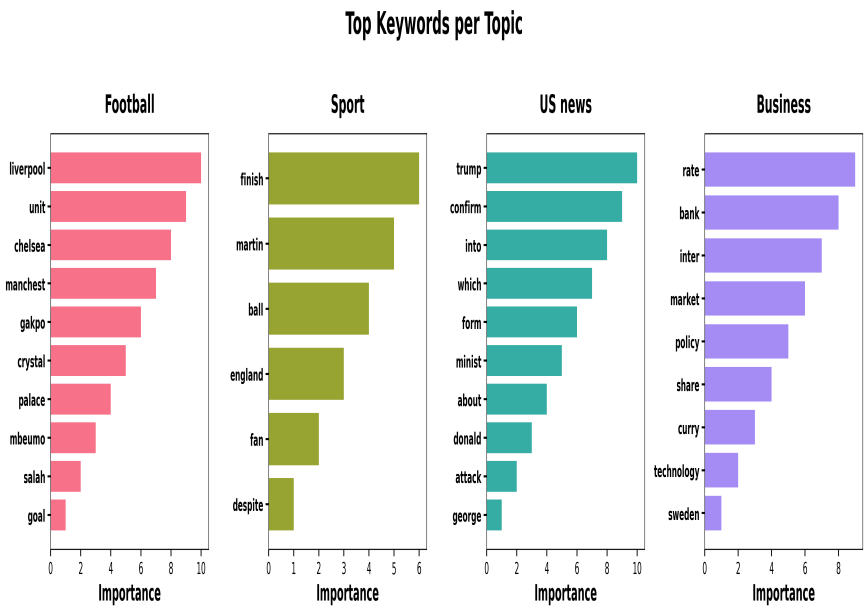

Figure 1: Top keywords for each topic

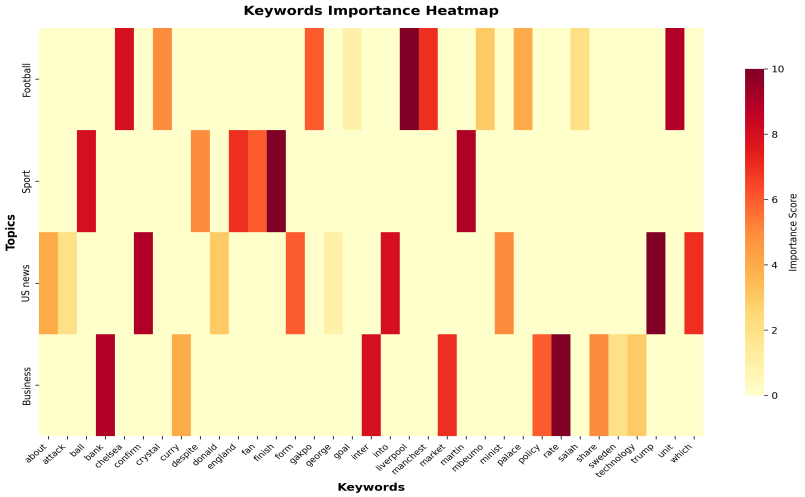

Figure 2: Keywords heatmap across topics

4 Results

The system was tested on 24 articles from the Guardian API. 4 thematic clusters were identified with an average silhouette score of 0.0968. The silhouette value is relatively low, which is expected for real-world news data where topics naturally overlap and many articles mention multiple themes in parallel. In this context the silhouette score is used mainly as a sanity check rather than an absolute quality target, and the final number of topics was chosen based on qualitative inspection of keyword lists and dashboards. TextRank generates summaries approximately 10% of the original text length, providing concise but informative extracts that allow the user to skim large volumes of articles more efficiently.

Figure 3 shows the distribution of articles across identified topics. The distribution reveals that some topics contain more articles than others, which is typical for news datasets where certain themes dominate depending on current events.

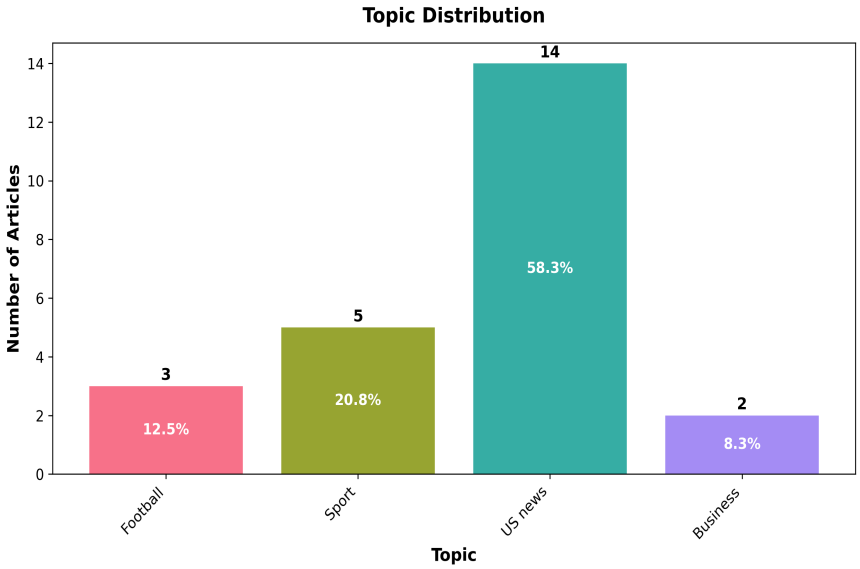

Figure 3: Distribution of articles across topics

The identified topics include: US news (14 articles, keywords: trump, confirm, into, which), Sport (5 articles, keywords: finish, martin, ball, england), and several specialized topics with fewer articles. Figure 4 visualizes these topics in a reduced two-dimensional space using PCA, showing how articles are distributed across topic clusters.

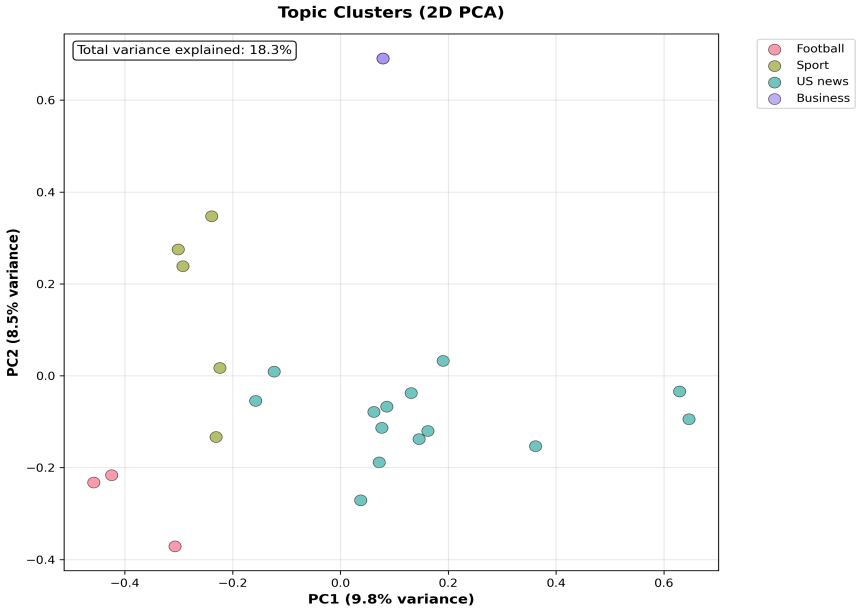

Figure 4: Topic visualization in reduced space (PCA)

Wordclouds provide a visual representation of the most frequent words in each topic. Below are wordclouds for the two largest topics, showing the dominant terms and themes.

5 Conclusion

7.1 Achieved Goals

All project goals were fulfilled. The system successfully downloads articles from the Guardian API, performs comprehensive text preprocessing, generates summaries using TextRank, groups articles into thematic clusters using TF-IDF and K-Means, and provides an interactive Streamlit dashboard for visualization and exploration. The application enables users to quickly get an overview of news articles and identify main topics efficiently.

References

[1] Mihalcea, R., Tarau, P. TextRank: Bringing Order into Text. Proceedings of EMNLP 2004.

[2] Pedregosa, F., et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011) 2825-2830.

[3] Guardian Open Platform API Documentation. https://open-platform.theguardian.com/ [Accessed: 2024]

[4] Streamlit Documentation. https://docs.streamlit.io/ [Accessed: 2024]

[5] Explosion AI. spaCy 3: Industrial-Strength Natural Language Processing in Python. https://spacy.io/usage [Accessed: 2024]