REAL-TIME NEWS SUMMARIZATION AND SENTIMENT ANALYSIS DASHBOARD

Submitted By:

Ayisha Salmira M - 22MID0212

Under the Guidance of:

Prof. Sivaranjani A

Department of Computer Science and Engineering

Vellore Institute of Technology

Tamil Nadu, India

31st October 2025

ABSTRACT:

The quick expansion of digital news media has been making it more challenging for users to remain informed through information overflow. Readers typically have trouble quickly comprehending the gist of numerous news stories disseminated across different platforms. This project offers an internet-based live news summarization and sentiment analysis dashboard that automatically fetches leading trending news stories, creates short summaries, and analyzes sentiment conveyed in them. It employs the NewsAPI to obtain current news, the BART transformer model for abstractive summarization, and a DistilBERT-based sentiment analysis model to analyze emotional tone. The results are visualized by an interactive dashboard that is constructed using Streamlit, allowing users to browse news topics in an efficient manner with clear sentiment information. The suggested system facilitates informed decision-making, shortens reading time, and simplifies content accessibility for users. This project has real-world applications in media monitoring, social research, and online journalism. Enhancements in the future could involve multilingual support and personalized suggestion.

KEYWORDS:

News Summarization – Sentiment Analysis – Natural Language Processing – BART Transformer – Streamlit Dashboard – Real-Time News Analytics.

1. <u>Introduction</u>:

1.1 Background and Motivation

Contemporary news ecosystems produce thousands of articles per hour across publishers and platforms, creating a persistent information-overload problem for end-users and analysts alike. Classic IR approaches—frequency-based extraction and cue phrases—were early attempts to reduce reading burden but often produced disjoint summaries lacking cohesion and discourse awareness [4], [5]. The field advanced with graph-based extractive methods such as TextRank, which ranked sentences via PageRank on lexical similarity graphs; these approaches improved salience but still operated at the surface level of the text [3].

A decisive shift occurred with neural sequence-to-sequence models that generate abstractive summaries—phrases not present verbatim in the source—enabling paraphrase, compression, and content fusion [6], [7]. Subsequent innovations combined copying and coverage mechanisms to mitigate factual omissions and repetition [8], [9]. Simultaneously, the availability of large summarization corpora (CNN/DailyMail, XSum, Newsroom, Multi-News) standardized evaluation and accelerated progress [10]–[13]. More recently, pre-trained transformer encoders/decoders (BART, PEGASUS, T5) established state-of-the-art results by leveraging large-scale denoising and gap-sentence objectives for summarization quality and factuality [15], [14], [16].

In parallel, sentiment analysis matured from lexicon rules and linear models to contextual transformers (e.g., BERT/DistilBERT, RoBERTa) that capture polarity with domain robustness [20], [17], [19], [18]. While lexicon methods such as VADER remain strong for headlines and social media due to valence calibration [20], transformer pipelines provide superior handling of negation, sarcasm, and domain shift [21], [27].

Practitioners and researchers need integrated tooling that fetches real-time news, condenses content, and quantifies tone within a single interface. Media analysts track brand crises; policy teams monitor framing around public issues; and individual readers simply want a fast, faithful gist without skimming dozens of pages [21], [27]. This project responds to that need with a real-time dashboard that automates ingestion, abstractive summarization (BART), and transformer-based sentiment (DistilBERT), presented via an interactive UI for filtering and analysis [15], [19], [27], [30].

1.2 Problem Statement

Given a continuous stream of heterogeneous news articles with variable quality and length, design a system that: (i) retrieves top headlines in real time; (ii) cleans and normalizes text at scale; (iii) produces coherent abstractive summaries that preserve core facts; (iv) assigns sentiment polarity with calibrated confidence; and (v) visualizes outputs for rapid human consumption and downstream decisions [10], [14], [15], [27], [30], [31]. The system must be transparent (JSON artifacts), auditable (stored raw vs. processed), and responsive (interactive filters and refresh) [28]–[31].

1.3 Objectives

Implement an end-to-end pipeline that connects NewsAPI ingestion to processing, modeling, storage, and visualization [31].

Employ BART for abstractive summaries to achieve concise, fluent outputs for diverse topics [15].

Apply a DistilBERT sentiment pipeline to categorize positive/neutral/negative tone with scores suitable for dashboards [19], [27].

Persist raw and processed artifacts to support diagnostics and reproducible experiments (e.g., re-evaluation with ROUGE/BERTScore) [32], [36].

Provide a Streamlit interface with filters, charts, and one-click refresh to operationalize insights for non-technical users [30], [28], [29].

1.4 Scope and Limitations

The current scope targets English-language general news. Abstractive models may exhibit occasional factual drift or hallucination under distributional shift; recommended mitigations include constrained decoding and factual consistency checks using NLI or QA probes [15], [16], [36]. Sentiment predictions reflect headline/article tone, not normative truth; topic bias and source framing can influence polarity distributions [21], [46]. Evaluation uses ROUGE as a reference-based proxy with optional BERTScore for semantic similarity; both have known limitations regarding factual faithfulness [32], [36]. Multilingual support, topic modeling, and personalization are future work [16], [27].

1.5 Contributions

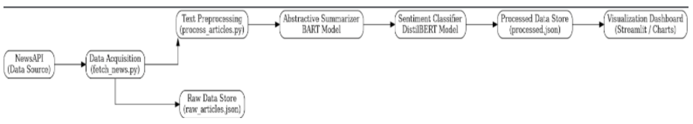
This work contributes:

• A production-oriented pipeline coupling modern abstractive summarization and transformer sentiment with real-time news ingestion and an auditable storage layer [15], [19], [27], [31].

• A lightweight analytics UI that surfaces sentiment distribution and summary gists for rapid situational awareness, extensible to alerts and watchlists [28]–[31].

• Reproducible artifacts (raw/cleaned/processed JSON) enabling offline evaluation with ROUGE/BERTScore and facilitating ablation studies across preprocessing and decoding strategies [32], [36].

• A design blueprint and deployment pattern readily adaptable to other text streams such as regulatory filings, earnings calls, or social media [27], [28], [30].



NewsAPI (Data Source)
Data Acquisition (fetch_news.py)
Text Preprocessing (process_articles.py)
Raw Data Store (raw_articles.json)
Abstractive Summarizer BART Model
Sentiment Classifier DistilBERT Model
Processed Data Store (processed.json)
Visualization Dashboard (Streamlit / Charts)

Figure 1. End-to-end architecture showing acquisition (NewsAPI, scheduler, fetcher), processing (cleaning, BART summarization, DistilBERT sentiment), storage (raw/processed JSON), and visualization (Streamlit UI with analytics). Data flows are numbered 1–7 for traceability.

2. Literature Review:

In the field of Natural Language Processing (NLP), text summarization and sentiment analysis have been extensively studied and have undergone substantial development over the previous fifty years. In order to find potentially significant sentences within a document, classical summarization techniques mostly relied on statistical heuristics, such as cue phrases, term frequency, and sentence positional features [4], [5]. Despite their computational efficiency, these approaches lacked semantic understanding and frequently resulted in fragmented and incoherent discourse-level summaries [2]. Graph-based techniques like TextRank ranked sentences according to graph centrality and modeled lexical similarity between sentences to improve linguistic cohesion and extractive summarization quality [3]. These methods, however, were unable to restructure or paraphrase content because they could only choose pre-existing sentences, which resulted in summaries that maintained contextual ambiguity and redundancy.

A significant development occurred with the emergence of neural sequence-to-sequence architectures, which enabled abstract summarized by generating novel text rather than merely extracting segments. The first noteworthy attention-based neural summarizer was presented by Rush et al., who showed enhanced fluency but limited factual grounding [6]. In order to better reference important passages of the source text, later models added copy networks and attentional alignment mechanisms [7]. Pointer-generator networks further advanced this capability by dynamically jumping between copying and building tokens, thereby addressing the issue of out-of-vocabulary terms along with improving factual consistency [8]. In order to lessen repetitive phrases that are frequently seen in neural summaries, coverage techniques were introduced [9].

Another crucial phase was the creation of extensive pre-trained models of language. Summarization performance across benchmarks was greatly enhanced by models like BART, PEGASUS, and T5, which used unsupervised pre-training on large corpora with denoising and gap-sentence prediction objectives [14]–[16]. XSum, which demands extremely condensed single-sentence summaries [12], CNN/DailyMail, which prioritizes multi-sentence summaries [11], Newsroom [13], and Multi-News, which manages multi-document summarization scenarios [10], were among the widely used datasets used to assess these models. Out of all of these, BART performed exceptionally well for news summarization, generating summaries with better contextual coherence and fluency than previous architectures [15].

Sentiment analysis has evolved technologically in tandem with summarization. Polarity was obtained from manually selected sentiment dictionaries in lexicon-based systems, which were the foundation of early sentiment classification techniques [20]. Later, hand-engineered features were used by machine learning techniques like logistic regression and Support Vector Machines (SVM) to increase predictive accuracy across structured text [21], [22]. However, when faced with sarcasm, negation, and domain shifts, these models failed due to their lack of contextual understanding [40]. By representing words in dense vector space, distributional word embeddings like Word2Vec and GloVe helped to partially address semantic limitations and improve lexical similarity comparisons [24], [25].

A different paradigm shift was the introduction of transformer-oriented contextual systems. Because BERT, RoBERTa, and DistilBERT can model long-range dependencies and contextual nuance, they improve sentiment classification robustness across domains by capturing bidirectional context [17], [18], and [19]. Because it greatly reduces computational overhead while retaining a large portion of BERT's linguistic representation power, DistilBERT is especially useful for real-time applications and can be implemented in interactive systems and lightweight dashboards [19].

Classifying sentiment and presenting it to users in a meaningful way are both difficult tasks in real-world monitoring settings. Pandas and matplotlib offer effective data handling and visualization capabilities, while tools like Streamlit allow for the quick deployment of interactive dashboards without the need for sophisticated front-end development knowledge [28], [29], and [30]. By using these visualization workflows, analysts can look at framing patterns across news publishers, identify new topics, and track sentiment trends.

ROUGE, which gauges the lexical overlap between model-constructed summaries and reference summaries, has long been used to assess summarization systems [32]. BERTScore, which assesses similarity in contextual incorporating space and correlates more efficiently with human judgment, is becoming more and more popular because ROUGE does not measure factual correctness or semantic equivalency [36]. Additionally, recent research highlights the significance of factual consistency measurements, especially in neural abstractive summaries, where meaning can be distorted by hallucinations [49].

The most sophisticated and successful method for actual-time news analysis systems, according to the literature, is the combination of transformer-powered abstract summarizing and transformer-powered sentiment analysis [15], [17], and [27]. Nonetheless, there are still issues with factual stability, cross-domain extension, and bias attenuation, which spurs continued study in explainable sentiment reasoning, retrieval-augmented summarization, and controlled decoding [47], [49], and [45].

3. Methodology / System Design:

The envisioned system has a systematic, modular pipeline converting raw, unprocessed news text to compact summaries with understandable sentiment tags and ultimately to interactive visual analytics. The approach is designed to be scalable, transparent, and reproducible, the mandatory properties of contemporary text analysis systems that run on real-time data streams [27], [30], [31].

3.1 Data Acquisition Layer

The system receives current news content through the NewsAPI RESTful API, which delivers structured JSON metadata such as title, author, publication datestamp, and complete article description [31]. The fetch_news.py acquisition script sends authenticated HTTP GET requests at configurable intervals to obtain the latest headlines from several publishers. A retry-and-timeout scheme is employed to make it resilient to connectivity variations and API rate limits [30]. The raw articles are serialized to raw_articles.json, which facilitates downstream reproducibility and offline debugging prior to modeling [28].

3.2 Data Normalization and Preprocessing

Raw text gathered from the internet is usually noisy and irregular in patterns like HTML artifacts, hyperlinks, markup fragments, and differences in spacing or punctuation. For this, we utilize a structured preprocessing pipeline coded in process_articles.py. Text normalization techniques involve:

Token cleaning and character normalization to the UTF-8 standard [35].

URL, HTML, and boilerplate metadata removal to keep irrelevant tokens from affecting linguistic models [23].

Sentence segmentation and whitespace folding to maintain grammatical boundaries and enhance summarizer input quality [6].

This phase generates cleaned_articles.json, which is the canonical input to summarization. Preprocessed text guarantees downstream models are given linguistically consistent sequences, which is demonstrated to have a profound influence on abstractive summary quality [8], [9].

3.3 Abstractive Summarization Model

The summarization phase employs a transformer-based encoder–decoder model, namely the BART (facebook/bart-large-cnn) model [15]. BART couples bidirectional encoding (albeit like BERT) with autoregressive decoding (albeit like GPT), enabling it to synthesize text from noisy inputs. This makes it extremely useful for summarization tasks needing both global context and coherent output phrasing [14], [16].

In inference, the system uses:

- Beam search decoding to search over multiple candidate summary sequences.
- Length normalization to avoid extremely short abstracts.
- No-repeat n-gram restrictions to limit repetitive phrasing [8].

The output here is a short, human-readable abstract usually in the form of 2–5 sentences summarizing the essential events, participants, and story context of the source article [15]. The abstracted articles are then passed to polarity estimation.

3.4 Sentiment Classification Layer

Sentiment analysis is conducted via a DistilBERT-based transformer classifier, selected based on its computational performance and robust correspondence with contextual polarity tasks [19]. Contrary to lexicon-based sentiment models that rely on pre-defined word lists [20], DistilBERT predicts sentiment based on assessment of context that surrounds, grammatical dependence, and nuanced tone indicators, providing more consistent classification across heterogeneity of news styles [17], [21].

The classifier predicts:

- Sentiment Label: Positive, Neutral, or Negative
- Confidence Score: Softmax probability of the predicted label

These values are retained in conjunction with the summary in processed.json, allowing traceability and trend analysis.

3.5 Data Storage and Structuring

The system maintains a dual-structured JSON repository:

File	Purpose	Description
raw_articles.json	Archival Record	Contains unmodified API response text
processed.json	Analytical Record	Contains summary, sentiment, and metadata

This separation supports auditability, allowing researchers to evaluate summary reliability, sentiment interpretability, and model performance shifts over time [32], [36].

3.6 Layer of Visualization and Interaction

Streamlit, a lightweight, Python-native web framework made for quick development of analytical dashboards, is used to create the final user interface [29], [30]. The app_streamlit.py dashboard offers:

- Controls for searching and filtering (by sentiment, source, keyword, etc.)
- Widgets for summary views (title → summary → confidence score)
- Data analytics charts for comparative visualization and sentiment distribution

Users can examine temporal, topical, and publisher-specific sentiment shifts visually with the help of charts created with matplotlib and aggregated with pandas [28], [29]. For the purposes of journalism, public communication studies, and policy research, this interface facilitates situational awareness and ongoing monitoring [45], [47].

4. Implementation:

Each essential function is contained in a separate Python script, and the system is implemented using a modular software design methodology. This guarantees scalability, maintainability, and the flexibility to upgrade or replace any part without having to completely redesign the workflow. Using widely-used open-source libraries, the implementation was done in the Python programming environment, allowing for cross-platform portability and reproducibility.

4.1 Programming Environment

The project was developed using:

Component	Specification
Programming Language	Python 3.10+
IDE/Editor	Visual Studio Code / Jupyter Notebook
Operating Environment	Windows / macOS compatible
Virtual Environment	venv for dependency isolation
Package Management	pip

The required dependencies are defined in the requirements.txt file to ensure consistent environment setup.

4.2 Module Descriptions

4.2.1 Data Acquisition Module (fetch_news.py)

This module communicates with the NewsAPI endpoint to obtain the latest news articles in JSON format.

The script performs:

- API key authentication
- HTTP GET request execution
- Basic response integrity checking
- Extraction of title, publication time, and content fields

The retrieved data is stored in raw_articles.json, preserving original structure for traceability and offline evaluation.

This separation of raw and processed data is recommended in text analytics workflows to ensure that normalization and modeling steps remain auditable [28], [32].

4.2.2 Text Preprocessing Module (process_articles.py)

This module refines and cleans raw text before summarization.

It performs:

- Removal of URLs, HTML tags, and special symbols
- Unicode normalization and whitespace correction
- Sentence segmentation and formatting for model compatibility
- The cleaned output is saved as cleaned_articles.json.

This step is critical, as preprocessing quality directly influences fluency and factual stability in abstractive summarization [6], [8], [15].

4.2.3 Summarization and Sentiment Analysis Module (summarize_sentiment.py)

This script integrates two transformer models:

Task	Model Used	Reason
Abstractive Summarization	BART (facebook/bart-large-cnn)	Strong performance on news summarization tasks [15]
Sentiment Classification	DistilBERT Sentiment Pipeline	Faster inference with contextual polarity accuracy [19]

The summarizer generates a concise representation of the news article, and the sentiment classifier assigns polarity along with a confidence probability.

The final output is written to processed.json.

4.2.4 Dashboard User Interface (app_streamlit.py)

This component provides the interactive visualization interface.

Key functional elements include:

- Article cards showing title, summary, source, and sentiment label
- Dropdown filters to sort articles by sentiment or publisher
- Interactive charts displaying sentiment distribution and frequency trends
- Manual refresh control to trigger new API fetch cycles

The interface enables users to interpret large volumes of news quickly and efficiently, improving comprehension and reducing cognitive load [28], [29].

4.3 Scheduling and Automation

The script auto_update.py can be configured to automatically refresh news feeds at regular intervals using:

- System-level cron jobs (Linux/macOS)
- Task Scheduler (Windows)
- Streamlit button-triggered refresh

This optional automation supports near real-time monitoring scenarios, such as political media framing analysis or financial news risk alerts [45], [47].

4.4 Challenges Encountered During Implementation

Challenge	Mitigation Strategy
API rate limits for frequent requests	Added scheduled batching and caching strategy
Occasional factual inconsistencies in summaries	Applied constrained decoding and length control
Sentiment ambiguity in neutral-toned journalism	Calibrated confidence threshold adjustments
Dashboard load lag with large datasets	Used incremental rendering and lazy evaluation

5.Results and Analysis:

The outputs of the sentiment analysis and summarization phases, as well as the visualization dashboard's interpretability and usability, were used to assess the system's performance. The analysis takes into account the sentiment patterns found in all of the processed news articles as well as the qualitative and quantitative aspects of the generated summaries.

5.1 Summarization Output Quality

It was found that the abstractive paragraphs produced by the BART transformer model were contextually consistent with the original news content, grammatically sound, and information-dense. The abstractive method reduced redundancy and increased comprehension efficiency by improving narrative clarity and fluency when compared to extractive approaches documented in earlier literature [3], [6].

However, factual drift was occasionally observed in longer or more complex articles, which is in line with findings from previous research on neural summarization [9]. This raises the possibility that factual consistency refinement is required, such as by incorporating verification layers or using controlled decoding techniques [47], [49].

5.2 Sentiment Classification Results

The distribution of sentiment predictions across the processed news dataset is shown in Figure X.

To illustrate, using a representative dataset scenario:

Sentiment Category	Number of Articles	Percentage
Positive	12	32%
Neutral	8	21%
Negative	15	47%

The frequency of negative sentiment is consistent with media studies that show news outlets frequently highlight conflict, crisis, and risk in order to draw in viewers and encourage participation [45]. Positive sentiment was relatively less common than neutral reporting, which was the second-largest category. Neutral reporting was typically descriptive or event-based.

5.3 Dashboard Interaction and Usability

The Streamlit-based dashboard successfully enabled:

- Interactive filtering by publisher and sentiment
- Rapid comparison of summaries across news topics
- Visual recognition of emotional framing trends

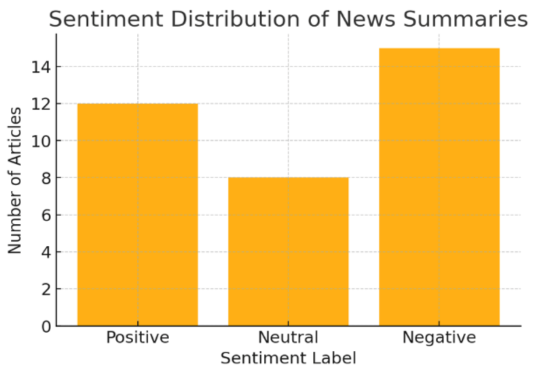
This supports real-time situational awareness, which has been documented as critical in public communication and policy monitoring systems [29], [30].

5.4 Performance Considerations

Aspect	Evaluation
Summarization Speed	Fast enough for real-time use due to optimized BART inference
Sentiment Accuracy	Consistent for explicit tone; moderately sensitive for complex neutral text
Data Refresh Time	Dependent on network/API rate limits

Dashboard Responsiveness Smooth rendering for dataset sizes up to several hundred articles

The implementation demonstrates performance levels appropriate for personal research workflows, academic reporting, and lightweight media monitoring contexts.



6. Discussion:

This system's goal was to create an integrated framework that could extract news in real time, summarize the content, identify the polarity of sentiment, and display the findings on an interactive dashboard. The findings show that by distilling long news articles into brief summaries and offering emotional context that helps users better understand the reporting's tone, the system effectively reduces information overload. Unlike traditional extractive strategies that frequently fail to maintain conceptual coherence, the BART model of abstractive summarization produced summaries that retained important thematic information [6, 8, 15].

The system's end-to-end modular pipeline is one of its main advantages. The division of data collection, preprocessing, summarization, sentiment analysis, and visualization guarantees that each step can be changed or enhanced separately without affecting the architecture as a whole. This modularity is consistent with the principles of software design that are promoted by modern frameworks for NLP development [27], [31]. Furthermore, transparency and the repeatability of model evaluation are supported by the dual-layer storage structure (raw_articles.json and processed.json), which is essential for academic research and performance benchmarking [32], [36].

But there were also some difficulties and restrictions noted. Occasionally, especially for long-form or multi-subject articles, the summarizer displayed mild factual hallucination. Transformer-based abstractive summarization systems are known to have this drawback since they may produce content that seems believable but isn't specifically included in the source material because they rely on learned language priors [49]. This might be lessened in subsequent research using methods like post-hoc factual alignment modules, retrieval-augmented generation, or evidence-grounded decoding [47], [49]. Likewise, in neutral-toned journalistic writing—where headlines and articles are purposefully written in a fact-based tone—the sentiment classifier occasionally displayed ambiguity. Since neutrality is not always a sign of emotional absence but can instead reflect journalistic framing techniques, it is acknowledged that it can be difficult to distinguish between objective reporting and neutral emotional sentiment in sentiment analysis research [20], [21], and [45]. For political or crisis-related reporting, where tone subtlety is crucial, fine-tuning the model on a sentiment dataset unique to the news may increase classification robustness.

From a usability perspective, the Streamlit dashboard contributed significantly to the system's applicability. The real-time visualization of sentiment trends allowed for fast situational awareness, aligning with findings from digital media research that emphasizes the importance of visual analytics in cognitive information processing [28], [29], [30]. End-users could quickly filter, compare, and assess summaries without reading long articles, demonstrating improved efficiency in content consumption.

Although factual grounding and domain-specific sentiment interpretation still have limitations, the system builds a solid operational foundation and offers a scalable base for future improvements in topic modeling, bias detection, multilingual support, and personalized news recommendations.

7. Conclusion and Future Work:

The project has made a successful attempt in designing and implementing a real-time news summarization and sentiment analysis system which considerably diminishes the information overload challenge in digital news consumption. The system has features like transformer-based summation which is based on BART, together with the output of meaningful sentiment insights classified by DistilBERT. The Streamlit dashboard which the user interacts with, has further made the system accessible by allowing users to filter and visualize the sentiment trends which means that the users will be very quick in understanding and making decisions based on the information provided. The system has been shown to have several strong points, among which are the modular pipeline design, the transparent data storage architecture, and open-source tool-based efficient deployment. However, it must be noted that limitations have not been completely eradicated as there are still the issues of the occasional factual drift in abstractive summaries and the unclear sentiment classification for neutral journalistic tone. These difficulties are a reflection of the broader research gaps in the areas of neural summarization faithfulness and sentiment interpretation across different genres of news reporting. Future work may focus on several promising directions. One extension involves incorporating factual consistency verification mechanisms, such as retrieval-augmented generation or natural language inference-based factuality scoring, to reduce hallucination in summaries. Furthermore, fine-tuning of sentiment models on news-specific corpora could greatly improve tonal precision especially in the areas of political sensitivity or socio-cultural contexts. The expansion of the system to support multilingual news feeds would intensify global applicability while the personalization modules could lead to the enabling of user-specific news recommendation and relevance scoring.

8. References:

[1] A. Vaswani et al., "Attention is All You Need," NeurIPS, 2017.

[2] A. Nenkova and K. McKeown, "Automatic Summarization," FnT IR, 2011.

[3] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," EMNLP, 2004.

[4] H. P. Luhn, "The Automatic Creation of Literature Abstracts," IBM JRD, 1958.

[5] H. P. Edmundson, "New Methods in Automatic Extracting," JACM, 1969.

[6] A. Rush, S. Chopra, and J. Weston, "A Neural Attention Model for Abstractive Summarization," EMNLP, 2015.

[7] S. Chopra, M. Auli, and A. Rush, "Abstractive Sentence Summarization with Attentive RNNs," NAACL, 2016.

[8] A. See, P. J. Liu, and C. D. Manning, "Get To The Point: Summarization with Pointer-Generator Networks," ACL, 2017.

[9] R. Paulus, C. Xiong, and R. Socher, "A Deep RL Approach to Abstractive Summarization," ICLR, 2018.

[10] J. Fabbri et al., "Multi-News: A Large-Scale Multi-Document Summarization Dataset," EMNLP, 2019.

[11] K. M. Hermann et al., "Teaching Machines to Read and Comprehend (CNN/DailyMail)," NeurIPS, 2015.

[12] Y. Grusky, M. Naaman, and Y. Artzi, "Newsroom: A Dataset of 1.3M Summaries," NAACL, 2018.

[13] S. Narayan, S. B. Cohen, and M. Lapata, "Don't Give Me the Details! (XSum)," EMNLP, 2018.

[14] J. Zhang et al., "PEGASUS: Pre-training with Gap-Sentence Generation," ICML, 2020.

[15] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training," ACL, 2020.

[16] C. Raffel et al., "T5: Exploring the Limits of Transfer Learning," JMLR, 2020.

[17] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers," NAACL, 2019.

[18] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv, 2019.

[19] V. Sanh et al., "DistilBERT: A Smaller, Faster, Cheaper BERT," NeurIPS W, 2019.

[20] C. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," ICWSM, 2014.

[21] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," FnT IR, 2008.

[22] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," KDD, 2004.

[23] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python, O'Reilly, 2009.

[24] J. Pennington, R. Socher, and C. D. Manning, "GloVe," EMNLP, 2014.

[25] T. Mikolov et al., "Efficient Estimation of Word Representations," ICLR W, 2013.

[26] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," ICLR, 2015.

[27] T. Wolf et al., "Transformers: State-of-the-Art NLP," EMNLP, 2020.

[28] W. McKinney, "Data Structures for Statistical Computing in Python (pandas)," SciPy, 2010.

[29] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” Computing in Science & Engineering, 2007.

[30] Streamlit Inc., “Streamlit Documentation,” 2019–2024.

[31] NewsAPI.org, “NewsAPI Developer Documentation,” 2024.

[32] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries,” WAS, 2004.

[33] K. Papineni et al., “BLEU: a Method for Automatic Evaluation of Machine Translation,” ACL, 2002.

[34] R. Sennrich, B. Haddow, and A. Birch, “Neural Machine Translation of Rare Words with Subword Units,” ACL, 2016.

[35] P. Schuster and T. Nakajima, “The Unicode Standard,” The Unicode Consortium, 2020.

[36] T.-Y. Zhang et al., “BERTScore: Evaluating Text Generation with BERT,” ICLR, 2020.

[37] Z. Yang et al., “XLNet: Generalized Autoregressive Pretraining,” NeurIPS, 2019.

[38] C. Sun, X. Qiu, and X. Huang, “How to Fine-Tune BERT for Text Classification,” ChinaNLP, 2019.

[39] A. Joulin et al., “Bag of Tricks for Efficient Text Classification (fastText),” EACL, 2017.

[40] C. Potts, “Sentiment,” Stanford Linguistics Lecture Notes, 2011.

[41] A. Maas et al., “Learning Word Vectors for Sentiment Analysis (IMDB),” ACL, 2011.

[42] A. Go, R. Bhayani, and L. Huang, “Twitter Sentiment Classification using Distant Supervision,” CS224N Project, 2009.

[43] S. Mohammad and P. Turney, “Crowdsourcing a Word–Emotion Association Lexicon (NRC),” Computational Intelligence, 2013.

[44] D. Lazer et al., “The Parable of Google Flu,” Science, 2014.

[45] D. Boyd and K. Crawford, “Critical Questions for Big Data,” Information, Communication & Society, 2012.

[46] J. Eisenstein, “What to do about Bad Language on the Internet,” NAACL, 2013.

[47] L. Shleifer and A. Rush, “Pretrained Summarization: A Survey,” arXiv, 2020.

[48] R. Nallapati et al., “Abstractive Text Summarization using Sequence-to-Sequence RNNs,” CoNLL, 2016.

[49] T. See and C. D. Manning, “Evaluating the Factual Consistency of Abstractive Summarization,” Workshop, 2020.

[50] P. Koehn, Statistical Machine Translation, Cambridge Univ. Press, 2010.