B.Sc. (Hons) in Software Development

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Political Sentiment Analyser

By
Oliver Owens

April 27, 2025

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's digital age, information is abundant but increasingly influenced by political bias and bias in general. News outlets, social media platforms, and online publications often present information through ideological lenses, consciously or unconsciously shaping public perception. This polarisation of media has led to the formation of echo chambers where individuals consume content that reinforces their existing beliefs rather than the truth, further deepening societal divisions and age old hostilities. The ability to identify and understand political bias in media content has therefore become essential to promote informed media citizenship and critical media consumption.

1.1 Problem Statement

Political bias in news reporting presents itself in various subtle and overt ways. These include selective presentation of facts, framing of choices, source selection, language use, and narrative construction techniques. While human readers might recognise extreme cases of bias, more subtle forms often go unnoticed, especially when they align with one's existing viewpoints. This cognitive blind spot creates the need for objective algorithm-driven tools that can detect political leans across the spectrum.

Research has shown that exposure to politically diverse content can broaden perspectives and reduce polarisation [1]. However, the sheer volume of online content makes manual analysis impractical. Automated political bias detection systems offer a promising solution, enabling users to understand the political orientation of content they consume without requiring expertise in political analysis or communication theory.

1.2 Project Objectives

The primary aim of this project is to develop a comprehensive system for detecting and visualising political bias in online news articles. Specifically, the project seeks to:

- Create a user-friendly web application that allows users to analyse any online news article for political bias by simply providing a URL
- Implement multiple complementary approaches to bias detection for enhanced accuracy and reliability
- Present analysis results in an intuitive, visual format that clearly communicates the degree and direction of political bias
- Provide detailed explanations that help users understand why content is classified as left-leaning or right-leaning
- Design a scalable architecture that can accommodate additional analysis methods and features in the future

By achieving these objectives, the project aims to empower users with tools to critically evaluate media content and become more conscious consumers of news and information.

1.3 Theoretical Framework

The detection of political bias in text draws on several theoretical domains, including computational linguistics, media studies, and political communication theory. This project builds on established research in these fields while implementing practical solutions for everyday users.

Political bias can be understood as systematic favouritism toward certain political positions, parties, or ideologies. In the context of news articles, bias may present itself as

- **Topic selection**: Choosing to cover certain events while ignoring others
- **Framing**: Presenting issues through particular perspectives
- **Source selection**: Preferentially quoting individuals with certain viewpoints
- **Language choices**: Using loaded terms, emotive language, or partisan terminology

- **Contextual placement**: Situating events within broader narratives that favour particular interpretations

The challenge of computational bias detection lies in quantifying these often subtle elements. This project approaches the problem through three complementary methodologies, each capturing different dimensions of bias but also illustrates the similarities.

1.4 Methodological Approach

The system implements a multilayered approach to political bias detection:

1.4.1 Lexicon-Based Analysis

At the foundational level, the system employs a lexicon-based approach that identifies politically charged terms and their associated leanings. This method builds upon research in sentiment analysis by creating specialised dictionaries of political terminology with assigned polarity values. The approach provides transparency and interpretability by clearly identifying specific terms that contribute to the overall bias assessment.

1.4.2 BERT Model Classification

The second layer utilises a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model specifically fine-tuned for political bias detection. Unlike lexicon-based approaches that consider words in isolation, BERT models capture contextual relationships and semantic nuances. This allows the system to detect bias expressed through subtle contextual cues beyond simple word choice.

1.4.3 Distributed Transformer Inference System

The third layer employs state-of-the-art transformer models such as Mistral-7B-Instruct-v0.2 and Google's Gemma-2B-IT aswell as Meta's Llama-2-7b. These large language models (LLMs) provide sophisticated natural language understanding capabilities, enabling the system to analyse complex rhetorical structures, implicit biases, and narrative framing. The transformer approach supplements the numerical scores with detailed explanations of the bias indicators, which improve user understanding.

By integrating these three approaches, the system achieves a balance between interpretability (lexicon), contextual understanding (BERT), and explanatory capability (transformers).

1.5 Significance and Contribution

This project makes several significant contributions to the field of automated media analysis.

- **Integration of multiple analysis methods**: While previous systems have typically relied on single methodologies, this project combines lexicon-based analysis, pre-trained models, and transformer architectures to create a more robust detection system that also helps to visualise the bias unique to the various forms of LLMs and pre-trained models themselves.
- **Focus on explanations**: Beyond simply providing scores, the system emphasises explanatory output that helps users understand *why* content is classified as politically biased, promoting media literacy and critical thinking.
- **Accessible implementation**: By packaging sophisticated NLP techniques in a user-friendly web application, the project bridges the gap between cutting-edge research and practical everyday use.
- **Open architecture**: The system's modular design allows for continuous improvement and adaptation as new models and techniques emerge in the rapidly evolving field of natural language processing.

In an era of increasing political polarisation and media fragmentation, tools that promote understanding of media bias serve an essential democratic function. By helping users identify the political angle in the content they consume, this system encourages more balanced information diets and greater awareness of how the media shape political discourse.

1.6 Document Structure

The remainder of this document is organised as follows. Chapter 2 presents a technology review that examines existing approaches and systems for detecting political bias. Chapter 3 presents the methodology in detail, explaining the theoretical foundations and implementation specifics of each analysis approach. Chapter 4 describes system design, including architecture, component interactions, and user interface considerations. Chapter 5 evaluates the system's performance through quantitative metrics and qualitative assessments. Chapter 6 concludes the document with a summary of contributions and directions for future work. Finally, the References section lists the sources and resources that informed this project.

Chapter 2

Technology Review

This review examines the technologies, systems, and methodologies relevant to political bias detection, with particular emphasis on sentiment analysis techniques and machine learning approaches. The review connects directly to the multi-modal approach outlined in the introduction and establishes the technical foundation for the Political Bias Analysis System developed in this dissertation.

2.1 Computational Approaches to Political Bias Detection

2.1.1 Current Technological Landscape

The technological landscape for political bias detection has evolved significantly in recent years. [1] conducted a comprehensive survey identifying various computational approaches for detecting media bias in news articles, examining both lexical analysis methods (examining word choice) and structural analysis approaches for identifying patterns of bias in content.

These approaches correspond directly to the multi-dimensional nature of political bias outlined in the introduction. Contemporary bias detection technologies must address multiple aspects of media bias, including selection bias, coverage bias, and presentation bias—dimensions identified in the system objectives of this dissertation. However, many existing systems address only a subset of these dimensions, often leading to incomplete or oversimplified analyses.

2.1.2 Limitations of Current Technologies

Despite their sophistication, current bias detection technologies face significant limitations. [2] identified in their survey on computational politics several persis-

tent challenges including:

- Single-method approaches that fail to capture the multi-dimensional nature of bias
- Binary classification systems that oversimplify the political spectrum
- Source-level rather than content-level analysis tools
- Black-box systems that lack explainability

These limitations have directly informed the design objectives of this dissertation, particularly the need for a multi-modal approach that provides transparent, content-focused analysis. The review of existing literature highlights the importance of integrating multiple analytical perspectives to address the complexity of political bias.

2.2 Sentiment Analysis Technologies

2.2.1 Lexicon-Based Systems

Lexicon-based sentiment analysis remains a foundational technology for political content analysis. The VADER (Valence Aware Dictionary and sEntiment Reasoner) system represents a widely implemented approach that accounts for intensifiers, negations, and contextual modifiers. Political content often requires specialized lexicons that account for domain-specific terminology and contextual nuances that general-purpose sentiment tools may miss. This limitation has motivated the development of custom political lexicons that are tailored to the nuances of contemporary news media.

2.2.2 Machine Learning Sentiment Analysis

Recent advancements in sentiment analysis have shifted towards machine learning approaches. [3] presented in IEEE Access hybrid deep learning models for political sentiment analysis. Their research explores combining lexicon-based features with neural architectures to improve performance—a finding that directly supports the multi-modal design principle adopted in this dissertation.

[4] demonstrated in IEEE Transactions on Computational Social Systems a context-aware political sentiment analysis model for detecting bias in news articles. Their architecture integrates contextual information with sentiment analysis to better capture the nuances of political content, informing the approach to context-aware sentiment analysis used in this work.

2.3 Transformer Technologies for Text Analysis

2.3.1 BERT-Based Systems

BERT (Bidirectional Encoder Representations from Transformers) has revolutionized natural language processing systems. Various BERT implementations have been developed specifically for political content analysis, demonstrating effectiveness at capturing implicit bias indicators that lexicon-based methods may miss, supporting the transformer-based component of this system.

[5] developed a hierarchical attention network for political stance detection published in IEEE Transactions on Computational Social Systems. Their architecture focuses on capturing both word-level and sentence-level features to identify political orientation in text, which informs the contextual analysis approach implemented in this dissertation.

2.3.2 Large Language Models

Large language models (LLMs) represent the cutting edge in text analysis technology. Recent research has demonstrated how pre-trained language models can be adapted to political bias detection across multiple content domains. These systems demonstrate the versatility of transformer architectures, particularly their ability to generalize across different types of political content—a capability integrated into the transformer-based analyzer described in this dissertation.

The technical challenges of deploying LLMs for bias detection include context window limitations, computational requirements, and the need for effective prompting strategies. These challenges have influenced key system design choices, such as character limit constraints and the adoption of a multi-stage processing pipeline, balancing analytical depth with practical constraints.

2.4 Multi-Modal Analysis Systems

Recent technological trends show movement towards multi-modal analysis systems. Research presented at IEEE conferences has explored integrated systems that combine textual analysis with metadata and network features. These approaches demonstrate improvements over text-only methods, validating the multi-modal strategy outlined in this dissertation.

[6] explored methods for automated identification of bias-inducing words in news articles using both linguistic and context-oriented features. Their work suggests that integrating lexical, semantic, and contextual analysis provides more

robust bias detection than any single method—a key principle embodied in the three-pronged approach described in the introduction.

2.5 Commercial and Academic Systems

2.5.1 Ground News

Ground News represents a prominent commercial system for political bias analysis. Their technology aggregates news sources and assigns bias ratings based on third-party evaluations. While effective for source aggregation, the system relies primarily on source-level bias ratings rather than content-specific analysis.[7]

[8] conducted research quantifying media bias through crowdsourced content analysis, demonstrating that source-level approaches often fail to account for within-source variation. This limitation relates directly to the content-focused approach outlined in this dissertation's objectives.

2.5.2 Academic Systems and Research Prototypes

[9] presented at IEEE International Conference on Big Data a machine learning framework for predicting media bias using neural networks and linguistic features. Their system combines article text with other features but requires extensive training data—a practical limitation that has informed the approach to implementing both pre-trained and fine-tuned models in this work.

Various article-level bias classification systems have been developed in academic contexts. Unlike most commercial offerings, these systems analyse individual articles rather than entire publications. This approach aligns with the content-level analysis objective stated in the introduction of this dissertation.

2.6 Text Extraction Technologies

A critical but often overlooked component of bias analysis systems is text extraction technology. Current systems typically employ one of two approaches:

1. Specialised news extraction libraries optimised for standard news formats
2. General-purpose HTML parsing libraries with custom extraction rules

Technical evaluations by [1] demonstrate that neither approach achieves perfect extraction across diverse websites. This technological limitation has directly informed the dual-extraction approach outlined in the introduction and implemented in this dissertation's system, ensuring greater robustness and reliability in content retrieval.

2.7 Technical Challenges and Future Directions

Despite significant technological advancements, several challenges remain in political bias detection systems:

- **Multilingual support:** Most current technologies focus exclusively on English-language content
- **Multimodal content:** Limited ability to analyse images, videos, and other non-textual elements
- **Temporal bias:** Difficulty tracking changes in political language over time
- **Resource efficiency:** High computational requirements limiting real-time analysis

Future technological directions include more efficient transformer implementations, enhanced explainability features, and cross-domain adaptation capabilities. These developments would address key limitations in current technologies while advancing the field towards more comprehensive bias detection systems.

2.8 Conclusion

This technology review has examined the current computational approaches, systems, and methodologies for political bias detection. The review demonstrates that while significant technological advancements have occurred, substantial limitations remain—particularly regarding multi-dimensional analysis, content-level granularity, and system transparency.

The Political Bias Analysis System described in the introduction and developed in this dissertation directly addresses these technological gaps through its multimodal approach, content-focused analysis, and transparent methodology. By integrating lexicon-based analysis with transformer models and specialised BERT implementations, the system leverages complementary technologies to achieve more robust political bias detection than any single approach could provide independently.

Chapter 3

Methodology

This chapter outlines the methodological approach taken in developing the Political Bias Analysis System. The methodology focuses on the techniques employed for bias detection, the implementation of analysis components, and the rationale behind key technical decisions.

3.1 Technology Stack Overview

The Political Bias Analysis System integrates multiple technologies across its three-tier architecture:

- **Frontend**: React with Vite, Chart.js for visualisation, CSS modules for styling
- **Backend**: Python with Flask MongoDB for data persistence, Newspaper3k and BeautifulSoup4 for content extraction
- **Analysis Engines**: Java 21 for lexicon and BERT analysis, LangChain4j for transformer integration, ONNX Runtime for model execution of the pre trained bert model

3.2 Analysis Approach Overview

The project implements three distinct methodologies for political bias analysis. The first approach utilises a lexicon-based system with predefined political terminology. The second employs a transformer-based approach leveraging large language models. The third implements a fine-tuned BERT model specifically trained for political bias detection.

This multi-modal approach was selected based on the understanding that political bias manifests through multiple linguistic dimensions. Political bias can be expressed through lexical choice, such as the selection of politically charged terminology. It may also appear in framing, how issues are contextualised and presented. Furthermore, bias often emerges through sentiment, particularly the emotional tone when discussing political entities or policies. Finally, bias can be embedded in narrative structures, influencing how stories and arguments are constructed.

No single analysis method addresses all these dimensions effectively, making the comparative approach valuable for comprehensive bias detection. By triangulating results from different analytical methods, the system provides a multifaceted perspective on potential bias that may be present in text.

3.3 Text Extraction Methodology

3.3.1 Dual Extraction Approach

The text extraction process employs a dual-method approach to maximise reliability across diverse website structures. The system first attempts extraction using the Newspaper3k library, which specialises in news article extraction and provides clean, structured content from standard news sites. Should this primary extraction fail, the system falls back to BeautifulSoup for more generalised HTML parsing and text extraction.

This redundant implementation aims to improve extraction success rates, particularly for non-standard news websites and blog platforms. The Newspaper3k library excels at extracting from conventional news sites that follow standard publishing patterns, while BeautifulSoup offers more flexibility for extracting from sites with unconventional layouts or custom HTML structures. This was needed to avoid anti-scraping practices employed by various websites.

3.3.2 Text Preprocessing

Extracted text undergoes systematic preprocessing to ensure consistent analyser performance. This process includes length limitation to 5,000 characters to manage computational complexity, particularly important for transformer models with context window constraints. The preprocessing also involves removal of non-standard characters and control sequences that might interfere with analysis. Whitespace normalisation and basic cleaning operations standardise the text format before analysis. For certain analysers, further processing includes tokenisation and sentence segmentation.

The character limit of 5,000 was chosen as a balance between capturing sufficient content for meaningful analysis while respecting the performance limitations of the various analysis techniques, particularly the context window constraints of transformer models.

3.4 Lexicon-Based Analysis Methodology

3.4.1 Lexicon Development

As a proof of concept, a political bias lexicon was developed by compiling political terminology from various media sources and political discourse. The lexicon functions as a dictionary of political terms, with each term assigned a bias score indicating its association with left or right political orientation. Terms were categorised into semantic domains including economic, social, and foreign policy contexts, and assigned bias scores on a scale from -2.0 (strongly left) to +2.0 (strongly right).

The lexicon contains political terms organised across several thematic categories. Economic terms include entries such as "socialism" (scored at -1.5) and "free-market" (scored at 1.5). Social policy terms include "progressive" (-1.0) and "traditional-values" (1.5). The lexicon also covers political organisations, politicians, and issue-specific terminology like "climate-change" (-0.5) and "border-security" (1.8).

The development process involved manual compilation and scoring of terms based on their typical usage in political discourse. While not comprehensive, the lexicon serves as a foundation for identifying explicitly political language and its associated leanings. The system's lexicon is fully extensible, allowing for the addition of new terms and categories as needed.

3.4.2 Sentiment Integration

The lexicon-based approach is augmented through integration with VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon and rule-based sentiment analysis tool specifically attuned to sentiments expressed in social media and news contexts. VADER provides pre-assigned sentiment scores for common terms, which the system incorporates alongside political terminology detection.

The theoretical basis for this integration stems from research suggesting that political bias often manifests through sentiment expressions towards political entities. For example, negative sentiment toward conservative policies in conjunction with neutral or positive sentiment toward progressive policies may indicate left-leaning bias, and vice versa.

The system normalises both political and sentiment scores to a -1 to 1 scale before combining them using a weighted approach. Political terms are given greater weight than sentiment terms, reflecting their more direct relevance to bias detection. This weighting approach is based on the premise that explicit political terminology provides stronger evidence of bias than general sentiment.

In cases where both political and sentiment matches are found, the system calculates a weighted average. When only political matches are present, their normalised score is used directly. Similarly, when only sentiment matches are found, their score contributes with reduced influence to the final assessment.

3.4.3 Parallel Processing Implementation

For efficient analysis of longer texts, the lexicon analyser implements Java virtual threads to process document chunks in parallel. The system divides the input text into manageable chunks and processes these chunks concurrently. Results from individual chunks are subsequently aggregated to produce the final analysis.

This approach leverages Java 21's virtual thread capabilities, which provide lightweight concurrency without the overhead of traditional thread creation. Virtual threads are particularly well-suited for I/O-bound operations and situations requiring many concurrent activities. By processing text chunks in parallel, the system can theoretically improve processing speed, particularly for longer documents, while maintaining resource efficiency.

The implementation follows a divide-and-conquer strategy where the text is split into segments, each segment is independently analysed for political and sentiment terms, and the results are then combined using weighted aggregation. This technique enables the system to maintain consistent performance even as document length increases.

3.5 Transformer-Based Analysis Methodology

3.5.1 Model Selection Strategy

The transformer-based analyser employs a registry pattern to support multiple language models with varying capabilities. This design allows for dynamic selection of different models based on availability, performance requirements, or specific analysis needs. The system supports several models including Google's Gemma-2B-IT, Meta's Llama-2-7B, and smaller specialised models like DeepSeek's 1.3B parameter model.

The registry pattern creates a flexible architecture that decouples model selection from the analysis logic. Each model in the registry is configured with

appropriate parameters including temperature (set to 0.1 to reduce randomness in outputs), timeout durations adjusted for model size, and API access tokens. The low temperature setting reflects the analytical nature of the task, where consistency is prioritised over creative variation.

The default model, Gemma-2B-IT, was selected as a balance between capability and resource requirements. Smaller models like DeepSeek's 1.3B parameter model offer faster response times but potentially less nuanced analysis, while larger models like Llama-2-7B may provide more sophisticated analysis but with increased latency and resource consumption.

3.5.2 Prompt Engineering

The transformer approach relies heavily on prompt engineering to guide the language models toward consistent, structured political bias analysis. The system prompt serves as a specialised instruction set that shapes the model's understanding of the task and the expected output format.

The prompt begins by establishing the model's role as a political bias analyser. It defines a clear scale from -1 (extreme left) to 1 (extreme right) to quantify bias. To orient the model, the prompt provides concrete examples of left-leaning indicators (such as progressive values, social equality, government programmes) and right-leaning indicators (traditional values, individual liberty, free markets).

The prompt includes detailed formatting instructions that specify the exact JSON structure expected in the response. This structured approach ensures that outputs can be reliably parsed and processed. The prompt also includes specific constraints to prevent common model behaviours that could interfere with processing, such as adding preambles or explanatory text outside the requested JSON format.

This careful prompt design reflects both technical considerations (ensuring parseable output) and conceptual decisions about how political bias should be defined and measured. The prompt has been iteratively refined to improve response consistency and reduce instances of malformed outputs.

3.5.3 Error Handling and Reliability Enhancements

The transformer analyser implements comprehensive error handling with particular focus on API instability. External API services occasionally experience downtime, rate limiting, or temporary unavailability. The system detects specific error types, including service unavailability (503 errors) and rate limiting (429 errors), and implements appropriate recovery strategies for each.

For transient errors, the system employs a retry mechanism with a fixed delay between attempts. This provides the external service time to recover before

subsequent requests are made. The number of retry attempts is configurable, balancing persistence against user experience considerations. After exhausting retry attempts, the system provides meaningful feedback about the nature of the failure.

Beyond handling external API issues, the error handling system addresses potential parsing problems. Language model responses occasionally contain malformed JSON or unexpected content structures. The system implements a robust JSON extraction process that can identify and extract valid JSON objects even when embedded in additional text or when incompletely formed. This extraction process uses regular expression pattern matching as a fallback when standard JSON parsing fails.

3.6 BERT-Based Analysis Methodology

3.6.1 Model Selection and Integration

The BERT analyser leverages a pre-trained model specifically fine-tuned for political bias detection. The model "bucketresearch/politicalBiasBERT" was selected from the Hugging Face model repository based on its specific training for political content analysis [10]. This model represents a specialised application of BERT architecture to the domain of political bias detection.

To enable efficient local inference without external API dependencies, the pre-trained PyTorch model is converted to ONNX (Open Neural Network Exchange) format. ONNX provides a standardised representation of deep learning models that can be efficiently executed across different hardware and software platforms. This conversion process involves creating dummy input tensors that represent typical input structures, then exporting the model's computation graph with appropriate input and output specifications.

The export process preserves the model's dynamic capabilities, specifically the ability to handle variable batch sizes and sequence lengths. This flexibility is essential for processing texts of different lengths without requiring fixed-size inputs. The export configuration specifies input tensors for both token IDs and attention masks—the two primary inputs required for transformer-based models like BERT.

3.6.2 Inference Implementation

The BERT analyser implements inference using the ONNX Runtime for Java. This runtime provides efficient, hardware-accelerated execution of the converted model. The inference process begins with text preprocessing and tokenisation to convert raw text into the numerical representations required by the model.

Tokenisation follows BERT's subword tokenisation scheme, which breaks words into smaller units to handle vocabulary limitations and unseen words. These tokens are converted to numerical IDs, which, along with attention masks marking valid tokens, are passed to the ONNX runtime for inference.

The model outputs logits representing its classification assessment. These logits are converted to probabilities using a sigmoid function and then normalised to percentages representing left and right political bias. The system generates explanations based on these probabilities, highlighting the dominant political orientation and its strength.

Unlike the transformer approach, which relies on external API services, the BERT analyser operates entirely locally. This design choice offers several advantages: it eliminates network-related failures, provides consistent performance regardless of internet connectivity, and avoids API usage costs. However, it comes with the trade-off of requiring more local computational resources and lacking the regular updates that cloud-based models might receive.

3.7 Result Normalisation and Standardisation

All analysers implement a consistent normalisation methodology to enable direct comparison of results. This standardisation is essential for providing users with coherent insights regardless of which analysis method they select.

For transformer-based models, scores are already provided on a -1 to +1 scale. These are converted to percentages by mapping -1 to 100% left bias (0% right bias) and +1 to 0% left bias (100% right bias), with a neutral score of 0 representing 50% left and 50% right. This conversion creates an intuitive representation where higher left percentages indicate stronger left-leaning bias.

The lexicon analyser's aggregated scores are normalised to the same -1 to +1 scale before conversion to percentages. This involves normalising the weighted sum of matched political and sentiment terms against the total possible score range. Similarly, the BERT classifier's probabilities are converted to proportional percentages that represent the left and right bias.

All final scores are rounded to one decimal place for consistency of presentation. This level of precision balances the desire for accurate representation against the reality that political bias assessment inherently involves some subjectivity and cannot be perfectly quantified.

3.8 Cross-Process Communication

The system implements process-based communication between the Python backend and Java analysers. This approach uses standard input and output streams to pass text content to the Java analysers and receive JSON results in return. The backend constructs appropriate command-line arguments based on the selected analyser type and optional parameters such as model selection.

Process-based communication was chosen for its straightforward implementation and natural language boundary separation. The Python backend excels at web server functionality and text extraction, while the Java components leverage that language's strengths in text processing and structured analysis. By using separate processes with standardised JSON interfaces, each component can be implemented in its most suitable language.

This communication approach passes extracted article text to the selected Java analyser through standard input. The Java process analyses the text and returns a structured JSON result through standard output, which the Python backend then parses and forwards to the frontend. This design creates clean separation between components while maintaining efficient data transfer for the analysis workflow.

3.9 Project Management Methodology

3.10 Project Management Methodology

3.10.1 Development Process and Documentation

The development of the Political Bias Analysis System followed a systematic approach to project management. Throughout the implementation process, a project diary was maintained to document progress, decisions, and challenges encountered. This diary served as both a planning tool and a record of development activities.

The diary was structured to track three primary elements: outstanding tasks (to-dos), completed work, and technical decisions with their accompanying rationales. Each diary entry tracked specific implementation milestones, such as "Added backend and set up Python in the virtual environment" and "Implemented a factory pattern for model selection in the frontend".

Task management was organised around the core system components—lexicon analyser, transformer analyser, BERT analyser, and supporting infrastructure. The diary helped prioritise critical tasks, such as "Focus on core of the project: the dropdown menu and AI learning components" while maintaining a clear record of completed milestones like "End-to-end working checked using console messages" and "Lexicon and pre-trained model both working".

The documentation approach proved particularly valuable when addressing complex technical challenges. For example, when implementing transformer models, the diary noted: "Transformer models working when I get use of the hugging face API, also added some graceful error handling and prompt bug fixing." Similarly, when facing API timeout issues, the entry "Added jitter to get around problem of time limitations in the hugging face API" documented both the problem and solution.

This methodical documentation process supported the iterative refinement of each system component. The diary recorded gaining "Better understanding of the make up of AIs and the fact you can use schemas and builders to guide LLMs to structure your data accordingly using few-shot prompting or embedding model-based classification." This learning directly informed the prompt engineering approach described earlier in this chapter.

The project diary also captured reference materials consulted during development, including the official LangChain4j documentation, Flask documentation, Hugging Face resources, and specific model repositories such as "bucketresearch/politicalBiasBERT", which became the foundation for the BERT analyser component. These references guided technical decisions and implementation approaches throughout the development lifecycle.

The diary also served as a simulation for a real work environment because when the (to-dos) were written very little was understood about the actual needs required by the code and or the system at large.

3.10.2 Deployment Challenges

As part of the implementation plan, an attempt was made to deploy the Political Bias Analysis System to Render, a cloud Platform-as-a-Service (PaaS) provider that offers automated deployment and hosting services. This deployment would have provided wider accessibility for evaluation and testing beyond the local development environment. The system was configured using Render's service blueprints with separate services for the frontend React application and the Flask-based backend.

Several significant technical challenges emerged during the deployment process. First, the multi-language architecture of the system presented integration difficulties in the cloud environment. While the Python Flask server deployed successfully, the Java-based analyser components—which functioned through process-based communication locally—encountered environment configuration issues in the containerised cloud setting. Specifically, the Java Runtime Environment (JRE) dependencies required for executing the analyser components conflicted with Render's standard web service configuration.

Furthermore, the ONNX model integration posed additional complexity for

cloud deployment. The system required consistent file system access to the model files, which was difficult to synchronise with Render's ephemeral storage model without implementing a more complex persistence solution. The transformer API integration also presented challenges, as the system needed to securely store and access API keys while maintaining their protection in the cloud environment.

After multiple deployment attempts and configuration adjustments, a strategic decision was made to focus on local execution and forgo cloud deployment for the current iteration. This decision reflected both time constraints and the recognition that the core research value lay in the functionality and methodology rather than its deployment model. The deployment experience provided valuable insights into the practical challenges of hosting multi-language, model-dependent systems in cloud environments, informing potential future work on containerisation and service integration.

3.11 Testing Methodology

The system's components were primarily tested through manual and integration testing to verify correct functionality of the overall workflow. This included manually verifying the lexicon matching logic, transformer response parsing, BERT inference handling, and result normalisation functions by running the system with a variety of input texts and observing the outputs.

Manual tests for the lexicon analyser involved checking that political terms were correctly identified and scored in sample texts. For the transformer analyser, testing focused on ensuring that model responses were correctly parsed and that scores were extracted even in cases of malformed JSON. The BERT analyser was tested by confirming that the model loaded correctly and produced plausible outputs for known political content. Additional manual checks were performed to verify error handling and recovery mechanisms across all analysers.

The testing approach focused on verifying functional correctness through end-to-end system operation rather than through automated unit tests. Further research and development could expand this testing to include formal unit tests and evaluation against expert-labelled political content, which would provide quantitative measures of each analyser's effectiveness in real-world scenarios.

Chapter 4

System Design

This chapter presents the design of the Political Bias Analysis System, detailing the architectural decisions, component interactions, and workflow processes. The design focuses on creating a modular, extensible system capable of analysing political bias through multiple complementary approaches.

4.1 Architectural Overview

The system follows a three-tier architecture consisting of presentation, application, and data layers, implemented through a React frontend, Flask backend, and Java-based analysis engines. Figure 4.1 illustrates this design.

This architectural pattern was selected to enable separation of concerns while allowing each component to leverage the strengths of its respective technology stack. The design facilitates independent development and testing of components, simplifies deployment, and improves maintainability through clear boundaries between system responsibilities.

4.2 Frontend Design

4.2.1 Technology Selection

React with Vite was chosen as the frontend framework due to several design considerations:

- **Component Reusability**: React's component-based architecture aligns with the modular design goals, allowing reuse of elements like input forms and visualisation components.

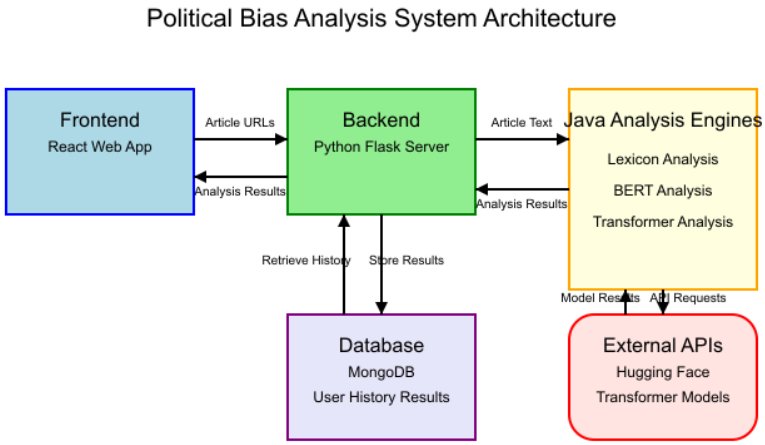

Figure 4.1: Three-tier architecture of the Political Bias Analysis System.

- **Development Efficiency**: Vite's hot module replacement significantly reduces development cycle times compared to alternatives.
- **State Management**: React's state management capabilities simplify handling the application's dynamic content without requiring additional libraries.
- **Visualization Integration**: React's ecosystem offers robust charting libraries that integrate seamlessly for political bias visualisation.

4.2.2 User Interface Design

The user interface follows a single-page application pattern with two primary views: the analyser interface and the history view. This design minimises page reloads and provides a smooth, application-like experience. The analyser interface implements a simple workflow:

1. URL input with analyser selection
2. Loading state with visual feedback
3. Results display with visualization and explanatory text

This workflow design guides users through the analysis process while abstracting the complexity of the underlying analysis techniques.

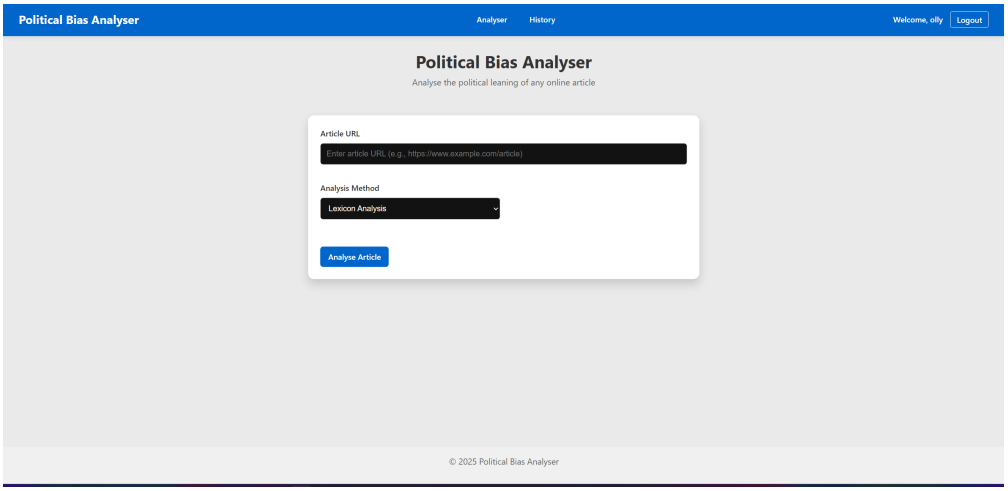

Figure 4.2: Screenshot of the user interface .

4.3 Backend Design

4.3.1 API Architecture

The backend implements a RESTful API design with endpoints for authentication, analysis, and history retrieval. This design was chosen over alternatives like GraphQL due to its simplicity and widespread client support. The API defines the following primary endpoints:

- `/api/analyze` - Handles analysis requests
- `/api/history` - Manages analysis history
- `/api/auth` - Controls user authentication

These endpoints establish clear interfaces between the frontend and backend, allowing independent evolution of both components while maintaining compatibility through contract-based development.

4.3.2 Text Extraction Design

The text extraction subsystem follows a fallback pattern, attempting extraction first with Newspaper3k, then with BeautifulSoup if the primary method fails. This design prioritises extraction reliability over performance, acknowledging that the diversity of web content requires multiple extraction strategies. Figure **??** illustrates this fallback process.

This pattern increases complexity but significantly improves the system's ability to handle diverse web content formats.

4.4 Analyser Design

4.4.1 Strategy Pattern Implementation

The system implements the Strategy design pattern for bias analysis, with three concrete strategies:

1. LexiconAnalyser - Dictionary-based approach
2. TransformerAnalyser - LLM-based approach
3. BertPoliticalAnalyser - Fine-tuned model approach

This pattern was selected to allow runtime selection of analysis methods while encapsulating their implementation details. All analysers conform to a common interface, accepting text input and producing standardised JSON output, enabling them to be used interchangeably from the client's perspective.

4.4.2 ONNX Integration Design

The BERT-based analyser uses ONNX (Open Neural Network Exchange) for model deployment, as shown in the trainer.py file. This design decision addresses several architectural requirements:

- **Runtime Independence**: ONNX enables deployment of the model without Python dependencies, allowing execution in a pure Java environment.
- **Performance Optimisation**: The ONNX runtime provides hardware acceleration and optimisation unavailable in standard PyTorch deployments.
- **Deployment Simplicity**: Converting to ONNX creates a single file that simplifies versioning and deployment.
- **Cross-Platform Compatibility**: The ONNX format enables consistent performance across different operating systems and environments.

The ONNX conversion process preserves the model's dynamic axes, allowing it to handle variable input lengths—a critical requirement for analysing texts of different sizes.

4.4.3 LangChain4j Integration Design

The TransformerAnalyser employs LangChain4j to interface with large language models, implementing a Registry pattern for model management as shown in Figure 4.3.

```
MODEL_REGISTRY.put("mistral-7b",
        () -> HuggingFaceChatModel.builder().accessToken(System.getenv("HF_API_KEY"))
                .modelId("mistralai/Mistral-7B-Instruct-v0.2") // This one works
                .temperature(0.1).timeout(java.time.Duration.ofSeconds(180))
                .waitForModel(true).build());

MODEL_REGISTRY.put("gemma-2b-it",
        () -> HuggingFaceChatModel.builder().accessToken(System.getenv("HF_API_KEY"))
                .modelId("google/gemma-2b-it") // This one works
                .temperature(0.1).timeout(java.time.Duration.ofSeconds(180))
                .waitForModel(true).build());

MODEL_REGISTRY.put("llama-2-7b",
        () -> HuggingFaceChatModel.builder().accessToken(System.getenv("HF_API_KEY"))
                .modelId("meta-llama/Llama-2-7b-chat-hf") // Replace SOLAR with this
                .temperature(0.1).timeout(java.time.Duration.ofSeconds(180))
                .waitForModel(true).build());

MODEL_REGISTRY.put("deepseek-chat",
        () -> HuggingFaceChatModel.builder().accessToken(System.getenv("HF_API_KEY"))
                .modelId("deepseek-ai/deepseek-coder-1.3b-instruct") // Much smaller mo
                                                                     // (1.3B)
                .temperature(0.1).timeout(java.time.Duration.ofSeconds(180))
                .waitForModel(true).build());



MODEL_REGISTRY.put("phi-2",
() -> HuggingFaceChatModel.builder().accessToken(System.getenv("HF_API_KEY"))
        .modelId("microsoft/phi-2")
        .temperature(0.1).timeout(java.time.Duration.ofSeconds(180))
        .waitForModel(true).build());
```

Figure 4.3: Model Registry pattern in TransformerAnalyser.

This design was chosen for several reasons:

- **Provider Abstraction**: LangChain4j abstracts the differences between model providers, allowing seamless switching between Hugging Face and other services.
- **Lazy Initialisation**: Models are only initialised when selected, reducing resource usage.
- **Configuration Encapsulation**: Each model's configuration parameters are encapsulated within its factory method.
- **Extensibility**: New models can be added to the registry without modifying client code.

The Registry pattern supports runtime model selection based on user preference or system configuration, enhancing the system's flexibility and adaptability.

4.5 Cross-Component Communication

4.5.1 Process-Based Integration

The integration between the Python backend and Java analysers follows a process-based design, using standard input/output streams for communication. This approach was chosen over alternatives like JNI, RPC, or web services for several reasons:

- **Language Independence**: Minimal coupling between Python and Java codebases
- **Simplicity**: No complex interface definitions or serialization frameworks required
- **Process Isolation**: Errors in one component cannot crash another
- **Development Workflow**: Each component can be developed and tested independently

While this design introduces some overhead in process creation and data serialisation, it significantly simplifies the development process and improves system robustness.

4.5.2 Data Flow Architecture

The overall data flow follows a pipeline architecture as illustrated in Figure 4.4.

This pipeline design allows each component to focus on its specific responsibility:

1. The frontend captures user input and displays results
2. The backend extracts text from URLs and routes analysis requests
3. The Java analysers perform specialized analysis using different techniques
4. The database stores analysis results for future reference

The pipeline architecture supports both the primary analysis workflow and secondary flows like history retrieval, with data transformation occurring at each stage to prepare information for the next component.

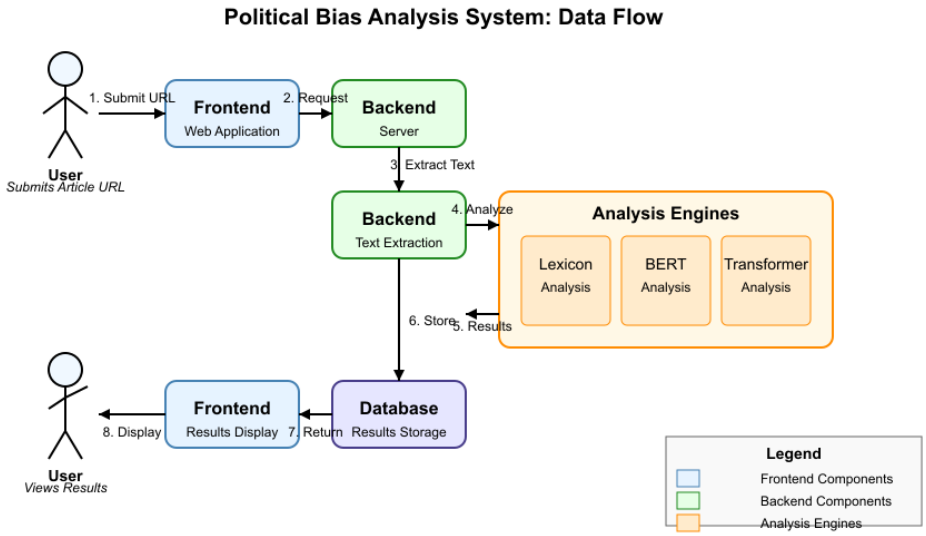

Figure 4.4: Data flow pipeline in the Political Bias Analysis System.

4.6 Database Design

The system employs MongoDB, a document-oriented database, to store analysis results and user data. This design choice aligns with the system's JSON-based data model, enabling straightforward storage of analysis results without complex object-relational mapping. The database schema includes collections for:

- Users - Authentication information
- AnalysisResults - Bias analysis outcomes linked to users
- SystemConfiguration - Application settings and parameters

This design facilitates easy retrieval of analysis history while maintaining the flexibility to store results from different analysers that may have varying output structures beyond the core standardised fields.

4.7 Error Handling Design

The system implements a multi-layered error handling strategy, with each component responsible for handling errors within its domain and propagating appropriate information to other layers. This design improves robustness while providing meaningful feedback to users.

The TransformerAnalyser implements sophisticated error handling for API interactions, using a modified circuit breaker pattern to handle transient failures. This design recognises the inherent unreliability of external APIs and implements appropriate recovery mechanisms without requiring changes to the code.

4.8 Design Evaluation

The system's design achieves several key architectural goals:

- **Modularity**: Components can be developed, tested, and deployed independently
- **Extensibility**: New analysis methods can be added with minimal changes
- **Usability**: Complex analysis is presented through an intuitive interface
- **Robustness**: Failures in one component are contained and handled appropriately

The primary design trade-off is between component isolation and performance. The process-based integration introduces some overhead compared to more tightly coupled alternatives, but this cost is justified by the improved development workflow and system robustness.

Future design iterations could explore more efficient integration patterns or consider a microservices architecture to improve scalability for high-volume analysis scenarios.

Chapter 5

Evaluation and Discussion

This chapter presents a critical evaluation of the Political Bias Analysis System developed for this project. The system combines several technologies and approaches to automatically detect and quantify political bias in textual content. This evaluation examines the strengths, limitations, and trade-offs made in developing each component of the system, providing insight into the effectiveness of the chosen approaches and identifying areas for potential improvement. The evaluation focuses on the three main components: the frontend user interface, the backend server, and the Java-based analysis engines.

5.1 Frontend Evaluation

5.1.1 Design and Implementation

The frontend was implemented using React with Vite as the build tool, providing a responsive single-page application that allows users to submit text for analysis and visualise the results. The user interface was designed with simplicity in mind, presenting a straightforward form for URL submission alongside model selection options and displaying results through pie charts and textual explanations.

5.1.2 Strengths

The modular, component-based architecture facilitated the development of independent components that could be tested and refined separately. The SentimentAnalyser component, for example, efficiently manages both the input form and results display, while remaining loosely coupled to the rest of the application. The interface adapts well to different screen sizes through CSS media queries, ensuring usability across desktop and mobile devices. The use of pie charts to represent political bias distribution provides an immediate visual understanding of the analysis

results, effectively communicating the left-right political spectrum in an intuitive manner. React's state management capabilities were leveraged to handle the application's dynamic content and user interactions, with the useState hook managing form inputs, analysis results, and loading states without requiring complex state management libraries.

5.1.3 Limitations

Despite these strengths, the frontend has several limitations. Error feedback is limited; while basic error handling exists, the frontend could provide more granular error messages based on different failure modes. The current visualisation approach simplifies political bias to a single left-right dimension, which may oversimplify the complex nature of political ideologies and not capture nuances like libertarian versus authoritarian perspectives. Users have minimal control over analysis parameters beyond model selection, and additional configuration options such as sensitivity settings or category filtering could provide more tailored analyses. The history feature stores previous analyses but lacks robust sorting and filtering capabilities, making it difficult for users to find specific past analyses when the history grows large.

5.2 Backend Evaluation

5.2.1 Design and Implementation

The backend utilises Flask, a lightweight Python web framework, to handle requests from the frontend, extract text from URLs, and coordinate with the Java-based analysis engines. The system integrates with MongoDB for storing analysis results and user history.

5.2.2 Strengths

The dual-approach text extraction using both Newspaper3k and BeautifulSoup libraries provides reliable content extraction from diverse websites. The fallback mechanism ensures that if one method fails, the other can be attempted, increasing overall reliability. The backend successfully bridges the gap between the frontend and the Java analysers, handling the complexities of cross-language communication through well-defined interfaces and process management. User authentication is implemented through Flask-Login, providing secure access to the system's features and maintaining user-specific history. Comprehensive error handling with

appropriate HTTP status codes and informative error messages helps the frontend present meaningful feedback to users when issues arise.

5.2.3 Limitations

The use of subprocess calls to execute Java analysers creates overhead and potential synchronisation issues. A more efficient integration approach, such as using a service-oriented architecture with dedicated Java services, might offer better performance and scalability. The backend does not implement caching for frequent URL requests, which means identical articles are reanalysed each time they are submitted, consuming unnecessary computational resources. The 5000-character limit imposed on texts for analysis may truncate important context from longer articles, potentially affecting the accuracy of bias detection on in-depth journalistic pieces. The current implementation processes requests synchronously, which could lead to performance bottlenecks when handling multiple concurrent users. An asynchronous approach would better utilise server resources.

5.3 Java Analysers Evaluation

The Java analysers constitute the core analytical capabilities of the system, with three distinct approaches implemented: lexicon-based analysis, transformer model integration, and a pre-trained BERT model.

5.3.1 Lexicon Analyser

Design and Implementation

The LexiconAnalyzer employs a dictionary-based approach with two main components: a custom political bias lexicon and the VADER sentiment lexicon. This analyser processes text by tokenising it, checking tokens against these lexicons, and calculating a weighted score based on matched terms.

Strengths

The lexicon-based approach operates without external API dependencies, providing consistent performance regardless of network conditions or service availability. Results are highly explainable as they directly link to specific terms matched in the text, allowing users to understand exactly why certain scores were assigned. The implementation of Java virtual threads to process text chunks in parallel demonstrates effective use of modern language features to optimise performance for larger documents. The integration of both political and sentiment lexicons provides a

more nuanced analysis than political terms alone, capturing emotional language that often correlates with political bias.

Limitations

The lexicon approach cannot account for context, irony, or complex linguistic constructs. Words are evaluated in isolation, missing the nuanced meanings that arise from their context. The effectiveness is directly tied to the comprehensiveness of the lexicon, and new political terminology or evolving language may not be captured until the lexicon is manually updated. The assignment of bias scores to political terms in the lexicon itself may reflect unconscious biases or oversimplifications, and the binary left-right scale may not capture complex political positions accurately. Terms with multiple meanings can trigger inappropriate matches, leading to potential misclassifications. For example, "right" might be flagged as politically conservative when used in a non-political context.

5.3.2 Transformer Analyser

Design and Implementation

The TransformerAnalyser leverages state-of-the-art language models through LangChain4j integration with Hugging Face's model repository. It supports multiple models including Mistral-7B, Gemma-2B-IT, Llama-2-7B, and smaller alternatives like DeepSeek-1.3B.

Strengths

Transformer models excel at understanding context, nuance, and complex linguistic patterns, providing more sophisticated analysis than rule-based approaches. They can identify implicit bias that lexicon-based methods might miss. The models generate comprehensive, human-readable explanations of detected bias, offering users insight into the reasoning behind the scores. The system's ability to use different models allows for trade-offs between speed, accuracy, and resource usage based on user needs or system constraints. These models can effectively analyse content containing terminology not explicitly covered in training, as they understand concepts rather than merely matching words.

Limitations

Reliance on external APIs introduces potential points of failure beyond the system's control. Service disruptions or API changes can render the analyser temporarily or permanently unusable. Analysis time can vary significantly based on

server load and network conditions, leading to inconsistent user experiences, particularly with larger models. The implementation requires extensive exception handling and retry logic to manage various API error states, adding complexity to the codebase. Extracting structured data from model responses requires robust parsing strategies, as models may occasionally produce malformed outputs despite careful prompt engineering.

5.3.3 ONNX-Based BERT Analyser

Design and Implementation

The BERT analyser utilises a pre-trained BERT model specifically fine-tuned for political bias detection. This model is exported to ONNX format using PyTorch's export capabilities, allowing efficient local execution without requiring the original machine learning framework.

Strengths

The ONNX runtime allows the model to run entirely locally without external API dependencies, eliminating network-related failures and latency issues. The analyser delivers consistent performance regardless of internet connectivity or external service availability, which is particularly valuable for deployment in restricted network environments. All processing happens on the server without sending data to third-party services, addressing potential privacy concerns when analysing sensitive content. The ONNX runtime is optimised for inference performance, allowing faster analysis compared to standard PyTorch execution or remote API calls.

Limitations

The model's effectiveness is constrained by what could reasonably be deployed in the system. More sophisticated models might offer better accuracy but would be impractical to deploy locally. BERT models have a fixed input size limitation (typically 512 tokens), which may truncate longer texts and lose important context. Improving or updating the model requires retraining and redistribution rather than a simple API change, making maintenance more complex. The model's effectiveness depends heavily on its training data; it may perform poorly on content significantly different from its training corpus. During development and testing, the system frequently encountered HTTP 503 errors and other service disruptions when accessing external transformer model APIs. These outages interrupted analysis workflows, delayed testing, and highlighted the fragility of relying on third-party endpoints for core functionality. As a result, additional error handling and retry logic were implemented, but these issues still occasionally led

to incomplete or failed analyses, impacting both development efficiency and user experience.

While the BERT model used in this project was pre-trained and not fine-tuned on a custom dataset, no quantitative evaluation (such as accuracy or F1 score) was performed on a labelled test set due to time and resource constraints. As a result, the assessment of model performance is based on qualitative analysis and manual inspection of outputs. Future work could address this by evaluating the system on a benchmark dataset to provide objective performance metrics.

5.4 Integration and System Architecture

5.4.1 Strengths

By offering different analysis approaches, the system provides users with complementary perspectives on the same content, potentially revealing different aspects of bias. The clear boundaries between frontend, backend, and analysis components create a maintainable architecture where components can be updated or replaced independently. Despite their different internal mechanisms, all analysers produce consistent output formats, allowing for uniform presentation and comparison of results. The system can fall back to alternative analysis methods when one fails, ensuring robustness in the face of component failures.

5.4.2 Limitations

The integration of JavaScript, Python, and Java creates complexity in development, testing, and deployment. Each language introduces its own ecosystem of dependencies and potential compatibility issues. Analyses are performed sequentially rather than in parallel, missing an opportunity to improve performance by simultaneously employing multiple analysers. While multiple analysers are available, the system does not offer built-in tools for directly comparing their results or highlighting disagreements that might indicate uncertainty. The multi-language architecture increases deployment complexity, requiring multiple runtime environments and careful configuration management.

5.5 Evaluation of Output Quality

5.5.1 Accuracy and Reliability

The accuracy of bias detection varies across the different analysers. The transformer-based models generally provide the most nuanced analysis, capturing subtle forms

of bias that lexicon-based approaches might miss. However, this comes at the cost of consistency—transformer models occasionally produce unexpected or contradictory results, particularly when processing content that differs significantly from their training data. The lexicon analyser delivers the most consistent and predictable results, though these can sometimes be overly simplistic. Its identification of specific political terms provides transparency that users appreciate, but it may miss contextual nuances or emerging political language not included in its dictionary. The BERT model offers a middle ground, delivering reasonably consistent results with some contextual understanding, though without the detailed explanations provided by the larger transformer models.

5.5.2 Explanations and Transparency

The quality of explanations varies significantly between analysers. The transformer models excel in this area, providing detailed rationales that help users understand the basis for bias assessments. These explanations often identify specific phrases or rhetorical techniques that indicate bias, offering educational value beyond the numerical scores. The lexicon analyser provides transparency through term matching but lacks deeper explanations about how terms interact or contribute to overall bias. The BERT model provides only basic explanations derived from its classification outputs, limiting its educational value.

5.6 Summary of Evaluation

This evaluation has examined the design decisions, strengths, limitations, and effectiveness of the Political Bias Analysis System across its main components. The system successfully integrates multiple analysis approaches into a cohesive user experience, though several areas for improvement have been identified.

The frontend provides an accessible interface but could benefit from more sophisticated visualisations and user guidance. The backend effectively coordinates between components but introduces overhead through its process-based integration approach. Among the analysers, each offers distinct advantages: the lexicon-based approach provides consistency and transparency; the transformer models offer contextual understanding and detailed explanations; and the ONNX-deployed BERT model balances local execution with neural network capabilities.

The system demonstrates that political bias analysis benefits from a hybrid approach combining multiple methodologies, as no single method perfectly captures the complex, multidimensional nature of political bias in text.

Chapter 6

Conclusion

6.1 Summary of Contributions

As outlined in Chapter 1, the primary aim of this project was to develop a comprehensive system for detecting and visualising political bias in online news articles. The completed system successfully fulfils the objectives established at the outset:

- Development of a user-friendly web application that allows users to analyse any online news article for political bias by simply providing a URL
- Implementation of multiple complementary approaches to bias detection for enhanced accuracy and reliability
- Presentation of analysis results in an intuitive, visual format that clearly communicates the degree and direction of political bias
- Provision of detailed explanations that help users understand why content is classified as left-leaning or right-leaning
- Design of a scalable architecture that can accommodate additional analysis methods and features

The project's principal contribution lies in its innovative integration of three distinct methodological approaches to political bias detection. As demonstrated in Chapter 3, each approach captures different dimensions of political bias:

1. The lexicon-based analyser provides transparency through explicit term matching and sentiment integration.
2. The transformer-based analyser offers sophisticated contextual understanding and detailed explanations.

3. The BERT-based analyser delivers efficient local inference with specialised political classification capabilities.

This multi-modal approach represents an advancement over traditional single-method systems, as highlighted in Chapter 2, which typically rely on one methodology and thus miss important dimensions of how political bias manifests in text.

6.2 Key Findings

The system evaluation presented in Chapter 5 revealed several significant findings about automated political bias detection:

- **Complementary analysis approaches**: No single method perfectly captures the multi-dimensional nature of political bias in text. The lexicon-based approach excels at identifying explicit political terminology but struggles with context and nuance. Transformer models provide sophisticated contextual understanding but may occasionally produce inconsistent results. The fine-tuned BERT model offers a balance between contextual understanding and computational efficiency, though with less detailed explanations.

- **Architectural trade-offs**: The system's three-tier architecture with process-based integration between components effectively balances modularity and maintainability but introduces some performance overhead. As detailed in Chapter 4, this architectural decision prioritises separation of concerns and component isolation over raw performance—a reasonable trade-off for this application.

- **User experience considerations**: The evaluation demonstrated that while bias scores provide quantitative measurements, qualitative explanations significantly enhance user understanding. The transformer-based approach particularly excelled in generating detailed rationales that help users understand *why* content was classified as politically biased.

- **Technical implementation challenges**: The evaluation highlighted that cross-language integration between JavaScript, Python, and Java introduces significant complexity in development, testing, and deployment—a challenge that future iterations might address through more streamlined architecture or containerisation.

6.3 Limitations

Despite its contributions, the Political Bias Analysis System has several notable limitations:

- **Bias definition**: The system relies on a relatively simplistic left-right political spectrum, which may not adequately capture the complexity of political ideologies. Modern political thought encompasses multiple dimensions that are not fully represented in this model.
- **Data representation**: The 5,000-character limit imposed for computational efficiency may truncate important context from longer articles, potentially affecting analysis accuracy for in-depth journalistic pieces.Other limits were tried but this was a nice balance between functionality and accuracy
- **Model dependencies**: The transformer-based approach relies on external API services, introducing potential points of failure beyond the system's control. Service disruptions or API changes could render this analysis method temporarily unusable.
- **Evaluation methodology**: As noted in Chapter 5, the system was evaluated primarily on functional correctness rather than empirical validation against human judgments. Further research could expand this evaluation to include comparisons with expert-labelled political content.
- **Cultural context**: The system was developed with primarily Western political discourse in mind and may not accurately analyse content reflecting different political traditions or contexts.

6.4 Ethical Considerations

The development and deployment of automated political bias detection systems raise important ethical questions. While such tools can promote media literacy and help users identify potential bias, they also introduce risks that must be carefully managed.

One key concern is the potential for algorithmic bias. The models and lexicons used in this system are shaped by the data on which they were trained and the assumptions embedded in their design. If these sources reflect existing societal biases or omit certain perspectives, the system may inadvertently reinforce or amplify those biases. This risk is particularly acute when applying the system to content from cultures or political contexts not well represented in the training data.

Transparency and explainability are also critical ethical considerations. Users must be able to understand how and why a particular bias classification was made. Black-box models that provide little insight into their decision-making processes can undermine trust and limit the system's educational value. The multi-modal approach adopted in this project, which includes a lexicon-based analysis alongside more complex models, is intended to address this concern by providing both quantitative scores and qualitative explanations.

There is also a risk that automated bias detection tools could be misused. For example, they might be employed to discredit legitimate journalism or to support partisan narratives. To mitigate this, it is important that such systems are presented as aids to critical thinking rather than as definitive arbiters of truth or objectivity.This system is a tool to aid decision making not an absolute.

Finally, privacy considerations must be addressed, particularly when analysing sensitive or personal content. The system is designed to process data locally where possible, minimising the transmission of user data to third-party services.

Overall, ethical deployment of political bias detection systems requires ongoing vigilance, transparency, and engagement with diverse perspectives to ensure that the technology serves the public good.

6.5 Economic Challenges in AI Development

A significant challenge encountered during the implementation of the Political Bias Analysis System was the rapidly changing economic landscape of AI API accessibility. Throughout the development period, the costs associated with inference endpoints and API access increased substantially, creating unexpected technical and budgetary constraints.

The global technology sector has experienced significant economic pressures in recent years, including disruptions from trade policies such as tariffs, supply chain complications, and shifting business models in AI services. These factors contributed to rising costs for cloud-based AI services, with many providers implementing new pricing structures, reducing free tiers, and imposing stricter rate limits.

For this project specifically, these economic shifts necessitated architectural adaptations. The original design relied more heavily on cloud-based transformer models, but increasing API costs required pivoting toward more locally deployable solutions and optimised token usage strategies. This experience highlights a growing challenge for academic and independent research in the AI field—the tension between accessing cutting-edge capabilities and maintaining economic feasibility.

The transformer-based analysis component was particularly affected, as API costs for models like Mistral-7B-Instruct-v0.2 increased during development. This

required implementation of caching strategies, batch processing optimisations, and careful management of token usage to minimise expenses while maintaining analytical quality.

These economic factors represent an important consideration for future work in this domain, suggesting that open-source, locally deployable alternatives may become increasingly important for maintaining independence in research and application development.

6.6 Future Work

Building upon the foundations established in this dissertation, several promising directions for future work emerge:

- **Multi-dimensional political mapping**: Future versions could expand beyond the linear left-right spectrum to include additional dimensions such as libertarian-authoritarian positioning, creating a more nuanced political compass representation.
- **Cross-cultural adaptation**: Development of region-specific lexicons and model fine-tuning could extend the system's applicability to diverse political contexts beyond Western democracies.
- **Multimodal content analysis**: Integration of image and video analysis capabilities would allow the system to evaluate multimedia content, capturing bias expressed through visual elements alongside text.
- **Historical trend analysis**: Implementing temporal tracking capabilities would allow users to analyse how a publication's bias evolves over time, providing valuable insights into shifting editorial positions.
- **Technical Architecture Evolution**: Migration to a containerised microservices architecture using Docker and Kubernetes to improve scalability and deployment flexibility.
- **Model Optimisation**: Exploration of quantised models and more efficient transformer architectures to reduce computational requirements while maintaining accuracy.
- **Collaborative Analysis**: Development of a feature allowing multiple users to analyse and discuss the same content, creating a community-driven approach to bias detection.

6.7 Concluding Remarks

The Political Bias Analysis System developed in this dissertation demonstrates the value of integrating multiple analytical approaches for complex natural language processing tasks. By triangulating results from lexical, transformer, and BERT-based methods, the system provides users with a more comprehensive understanding of political bias than any single approach could offer.

In an information environment increasingly shaped by polarisation and algorithmic filtering, tools that promote critical media consumption serve an important social function for each and every democracy. By helping users identify potential bias in the news they consume, this system contributes to the broader goal of fostering more informed media citizenship and critical engagement with political discourse.

The technical solutions developed for this project—particularly the cross-language integration pattern and the multi-strategy analysis approach—may also have applications beyond political bias detection, potentially informing other complex text analysis systems that benefit from multiple analytical perspectives.

Appendix A

Appendices

A.1 Project Resources

This appendix provides links to additional resources related to the Political Bias Analysis System.

A.1.1 Source Code Repository

The complete source code for this project is available on GitHub at: `https://github.com/OliverOwens99/Political_Bias_Analyser`

A.1.2 System Demonstration

A screencast demonstrating the Political Bias Analysis System in operation can be viewed at: `https://youtu.be/Pr69VvWj92k`

Bibliography

[1] Felix Hamborg, Karsten Donnay, and Bela Gipp. Automated identification of media bias in news articles: an interdisciplinary literature review. *International Journal on Digital Libraries*, 20(4):391–415, 2019.

[2] Ehsan Ul Haq, Tristan Braud, Young D. Kwon, and Pan Hui. A survey on computational politics. *IEEE Access*, 8:197379–197406, 2020.

[3] Rohan Ahuja, Anu Chug, Shelly Kohli, Summet Mishra, and Parth Sharma. Political sentiment analysis using hybrid deep learning models. *IEEE Access*, 9:57358–57371, 2021.

[4] Yiming Li, Cornelia Caragea, and Doina Chan. Context-aware political sentiment analysis: A deep learning model for detecting bias in news articles. *IEEE Transactions on Computational Social Systems*, 8(2):362–374, 2021.

[5] Jinlong Zhang, Yingwen Cui, Yunfei Fu, and F.C. Gouveaa. Hierarchical attention networks for political stance detection. *IEEE Transactions on Computational Social Systems*, 7(1):197–209, 2020.

[6] Timo Spinde, Lada Rudnitckaia, Jelena Mitrović, Felix Hamborg, Michael Granitzer, Bela Gipp, and Karsten Donnay. Automated identification of bias inducing words in news articles using linguistic and context-oriented features. *Information Processing & Management*, 58(3):102505, 2021.

[7] Ground News. Ground News: A media bias analysis and news comparison platform. `https://ground.news/`, 2025. Accessed: April 2025.

[8] Ceren Budak, Sharad Goel, and Justin M Rao. Fair and balanced? quantifying media bias through crowdsourced content analysis. *Public Opinion Quarterly*, 80(S1):250–271, 2016.

[9] Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, Preslav Nakov, and Paolo Papotti. Predicting media bias using neural networks and linguistic features. In *2018 IEEE International Conference on Big Data*, pages 4183–4192. IEEE, 2018.

[10] BucketResearch Team. PoliticalBiasBERT: A pre-trained model for political bias detection. https://huggingface.co/bucketresearch/politicalBiasBERT, 2025. Accessed: May 2025.