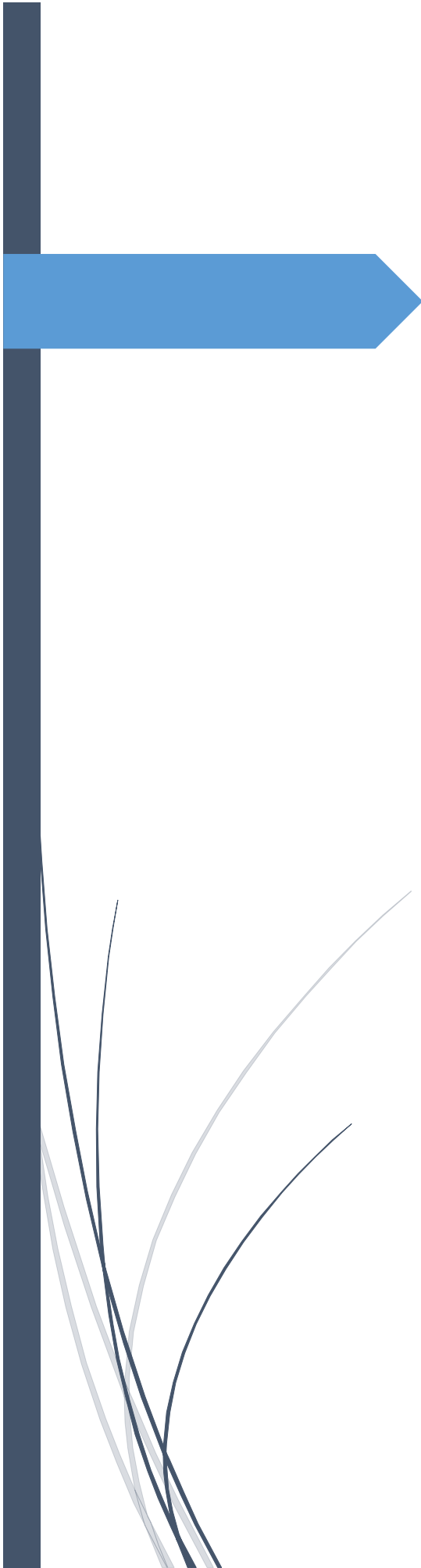




Sentiment Analysis Using Transformer- Based Language Models: A Comprehensive Comparative Study

Student Name: Malik Haroon

Programme: MSc Data and Artificial Intelligence

Module: L7 - Artificial Intelligence

Module code: ACCA7026

Table of Contents

Abstract.....	2
1. Introduction and Problem Statement.....	3
1.1 The Evolution of Sentiment Analysis in the Digital Age	3
1.2 Historical Development of Sentiment Analysis Methodologies	3
1.3 The Transformer Revolution in Natural Language Processing.....	3
1.4 The Interpretability Challenge in Deep Learning Models	3
1.5 Research Questions and Objectives.....	4
2. Literature Review	4
2.1 Foundational Research in Sentiment Analysis	4
2.2 Feature Engineering and Traditional Machine Learning Approaches.....	5
2.3 Deep Learning and Neural Network Approaches	5
2.3 Research Gaps and Contributions.....	6
3. Methodology.....	6
3.1 Dataset Description and Preparation.....	6
3.2 Text Preprocessing Pipeline	7
3.3 Traditional Machine Learning Model Design	8
3.4 Transformer-Based Model Design	8
3.5 Evaluation Framework	8
4. Results and Evaluation	9
4.1 Comparative Performance Analysis.....	9
4.2 Feature Importance Analysis	10
5. Discussion.....	11
6. Conclusion and Recommendations.....	12
6.1 Summary of Findings and Contributions.....	12
6.2 Recommendations for Practice and Future Research	12
References	13
Appendices.....	14

Abstract

This report presents a comparative analysis, compared to the previous models of sentiment classification built on IMDB movie review dataset, both the traditional machine learning models and the present-day state-of-the-art transformers-based models. Two models have been constructed and critically evaluated: TF-IDF vectorization model and Logistic Regression, and a fine-tuned DistilBERT transformer model. The experimental results prove that transformer-based method is highly performing with an accuracy of 90.52 and F1-score of 90.52 slightly higher than the traditional method with an accuracy of 89.09 and F1-score of 89.09. This change in performance, however, comes at a great computational cost, BERT is approximately 1,637 times slower to train (605.85 seconds versus 0.37 seconds). Outside quantitative measures, this work offers important observations on how interpretable a model is, as well as its computational and practical implementation aspects. As demonstrated by SHAP analysis and attention visualisation, a comparison between the two architecture reveals that even though Logistic Regression presents its feature importance in a transparent way, BERT represents subtle context-related relationships at the cost of interpretability. The results are relevant to the existing discussion of the trade-offs between model complexity and practicability in a production setting and this paper provides evidence-based information to practitioners who may choose architectures of sentiment analysis depending on their specific limitations in terms of accuracy requirements, computational resources, and interpretability requirements.

Keywords: Natural Language Processing, BERT, Transformer Models, Logistic Regression, Deep Learning

1. Introduction and Problem Statement

1.1 The Evolution of Sentiment Analysis in the Digital Age

The sudden and drastic increase in user-generated content over the digital platforms has radically changed the way organisations comprehend the opinion of the population, customer satisfaction, and market perception. Sentiment analysis or opinion mining has become an important computational linguistics tool that allows to obtain and categories subjective information automatically in text data (Zucco *et al.*, 2020). It has found a wide range of applications: commercial brand tracking, financial market forecasting, policy monitoring by the government, and feedback analysis in healthcare (Mends Karen, Anokwuru and Igba, 2025).

1.2 Historical Development of Sentiment Analysis Methodologies

The initial sentiment analysis has been based on dictionaries-based lexicon-oriented programmes based on dictionaries with pre-determined sentiment orientations (Nguyen, Nguyen and Ho, 2021), however dictionaries face context specific sentiment. The paradigm shifts in machine learning brought in statistical methods and the supervised algorithms were better than the lexicon-based methods (Beigi and Moattar, 2021). It was focused on feature engineering, where n-grammes, part-of-speech tags, and syntactic dependencies were used (Alamanda, Pabboju and Gugulothu, 2023). SVMs were useful with high-dimensional text data, whereas Logistic Regression provided probabilistic impressions and feature significance utilization (Kholwal, 2023a). The TF-IDF weighting scheme became a stalemate in the classical text classification.

1.3 The Transformer Revolution in Natural Language Processing

Transformer architectures became a paradigm shift, and attention mechanisms allowed processing concurrently and discovering long-range dependencies (Huang *et al.*, 2024). BERT (Bidirectional Encoder Representations and Transformers) brought a revolution to the field of NLP because it is pre-trained on large corpora and then fine-tuned on individual tasks (Gardazi *et al.*, 2025). As of 2019, fine-tuned BERT models are at the cutting edge of performance on the sentiment analysis benchmarks (Li, 2023), with other models such as DistilBERT providing compressed models that retain 95% of BERT performance through knowledge distillation.

1.4 The Interpretability Challenge in Deep Learning Models

Interpretability has become of importance as deep learning models continue to become more complex. Conventional algorithms such as Logistic Regression provide inbuilt interpretability (feature coefficients) (Li, 2023), whereas deep neural networks are black boxes, making decisions as black as possible, and hard to audit. For regulated industries, explainable decisions are needed to comply and ensure the trust in the industry by the stakeholders (Ihejirika, 2025). Models such

as SHAP offer unified explanations through cooperative game theory, and visualizing attention shows the input tokens that are used to explain transformer model predictions (Eggen, Lysnæs-Larsen and Strümke, 2025).

1.5 Research Questions and Objectives

This research deals with a few basic research questions that lie at the crossroads of the sentiment analysis methodology, performance of models, and deployment issues:

- In what ways do classic TF-IDF-based systems perform against transformer-based models in the classification performance, precision, recall, and F1-score on standard sentiment analysis data sets?
- How do these methods compare in terms of complexity in terms of computational effort, training time, and resource demands?
- What is the varying interpretability of these models and what can then be known of model decision-making by interpretability tools?
- What are the operational suggestions that can be made to companies that choose sentiment analysis architectures, in accordance with their own constraints and needs?

2. Literature Review

2.1 Foundational Research in Sentiment Analysis

The research roots of sentiment analysis can be traced back to the early research related to the attempts of Hearst (1992) to determine the possibility of automatic sentiment identification of textual information. Later studies by Wiebe (2000) defined the language of subjectivity analysis, differentiating between objective statements of facts and subjective utterances of opinion. These pioneer studies formed the theoretical basis on which other research in sentiment analysis would base their research.

(Danyal *et al.*, 2023) have performed seminal research which proved the efficiency of machine learning methods in categorizing sentiments. Their results contrasted several different classifiers such as Naive Bayes, Maximum Entropy, and Support Vector Machines to movie review data, and it was determined that machine learning methods could be used to classify binary sentiment tasks with an accuracy of more than 80%. This study overturned the direction of the subject to statistical methods and the IMDB movie review data set is now considered a standard of sentiment analysis studies.

(Ahanin and Ismail, 2022) proposed a sentiment classification method (an unsupervised pointwise mutual information-based classification) and proved that sentiment orientation could be estimated by the word associations in the absence of label training data. This piece of work led to the further

studies of lexicon-based and semi-supervised methods and widened the range of the methodological tools that sentiment analysis practitioners could use.

2.2 Feature Engineering and Traditional Machine Learning Approaches

Machine learning methods of sentiment analysis heavily rely on feature engineering. (Yang *et al.*, 2023) have provided detailed comparisons of feature representations in text classification and have shown that the simple bag-of-words feature can achieve competitiveness with other more advanced representations in combination with the right classifier. Their study indicated that Logistic Regression with L2 regularization is a useful text classification method in high-dimensional cases.

The TF-IDF weighting scheme that was originally used in the information retrieval was especially useful in text classification. As (Bhavani and Kumar, 2021) showed, SVM classifiers used with TF-IDF features performed at the state-of-the-art on text categorization tasks. Later work by (Abiodun *et al.*, 2021) examined the methods of selecting features to be used in text classification, and showed that with careful selection of these features, performance and efficiency may be enhanced.

In a study, (Kholwal, 2023) developed theoretical underpinnings of Bayesian text classification, showing that Logistic Regression using suitable priors would be effective in dealing with high-dimensional, sparse feature spaces of text data. Their contribution led to the popularization of the Logistic Regression in text classification problems.

2.3 Deep Learning and Neural Network Approaches

Convolutional architectures were the first to be applied to sentiment analysis using neural networks. It was shown by (Umer *et al.*, 2023) that a Convolutional Neural Network (CNN) on word embeddings could be used to perform at state-of-the-art levels on several text classification tasks. This paper showed that word2vec (Johnson, Murty and Navakanth, 2023) word embeddings were effective in semantic relationship representations and enhanced classification results.

The challenge of modelling the sequential dependencies of text in the text was solved with recurrent neural networks (RNNs) and long short-term memory (LSTM) networks. The authors proposed tree-structured LSTMs to sentiment analysis (Tai, Socher and Manning 2015), showing that the recursive architecture was able to learn syntactic structure and achieve better results on cranial tasks of sentiment classification.

Attention mechanism, which has been proposed by (Bilianos and Mikros, 2023) in the context of machine translation, quickly received applications in sentiment analysis. This is due to the fact that hierarchical attention networks were suggested by (Galassi, Lippi and Torroni, 2020) to

classify documents and proved that the attention process could recognise significant words and sentences enhancing the performance and comprehensibility.

2.3 Research Gaps and Contributions

Although there is a lot of research on sentiment analysis methodologies, there still are some significant questions that are not sufficiently developed. To begin with, although many studies have been conducted comparing the traditional and deep learning methods, few studies have been done with detailed analysis including computational efficiency, interpretability, and practical implementation considerations. Second, limited attention has been paid to the trade-offs between model complexity and interpretability, especially on the implications of such systems on organisations using sentiment analysis. Third, the tools of interpretability have become quite developed, but they are not used to compare model structures with each other.

This report fills these research gaps by conducting a comprehensive comparative study that involves performance indicators, computational performance, interpretability analysis and practical implications.

3. Methodology

3.1 Dataset Description and Preparation

The dataset on which the study was conducted is the IMDB movie review (Maas *et al.*, 2011) and includes 50,000 movie reviews that are evenly distributed in terms of negative sentiment and positive sentiment. Each review is associated with binary sentiment (positive/negative), and this gives a well-organized supervised learning exercise. The data set is perfectly balanced in terms of classes and there are 25,000 positive and 25,000 negative reviews hence no class imbalance correction methods are necessary.

Figure 1 shows the first exploratory data analysis indicated some significant characteristics of the data. The length of the review differs significantly, and the average length of the review is 1,176 characters (almost 235 words). The length distribution of the reviews is not different between the sentiment classes with positive reviews having 1186-character average and negative reviews having 1167-character average. This is the same thing, which indicates that the length of the review by itself is not a true predictor of the sentiment, as it is essential to focus on the linguistic content, rather than on the document length.

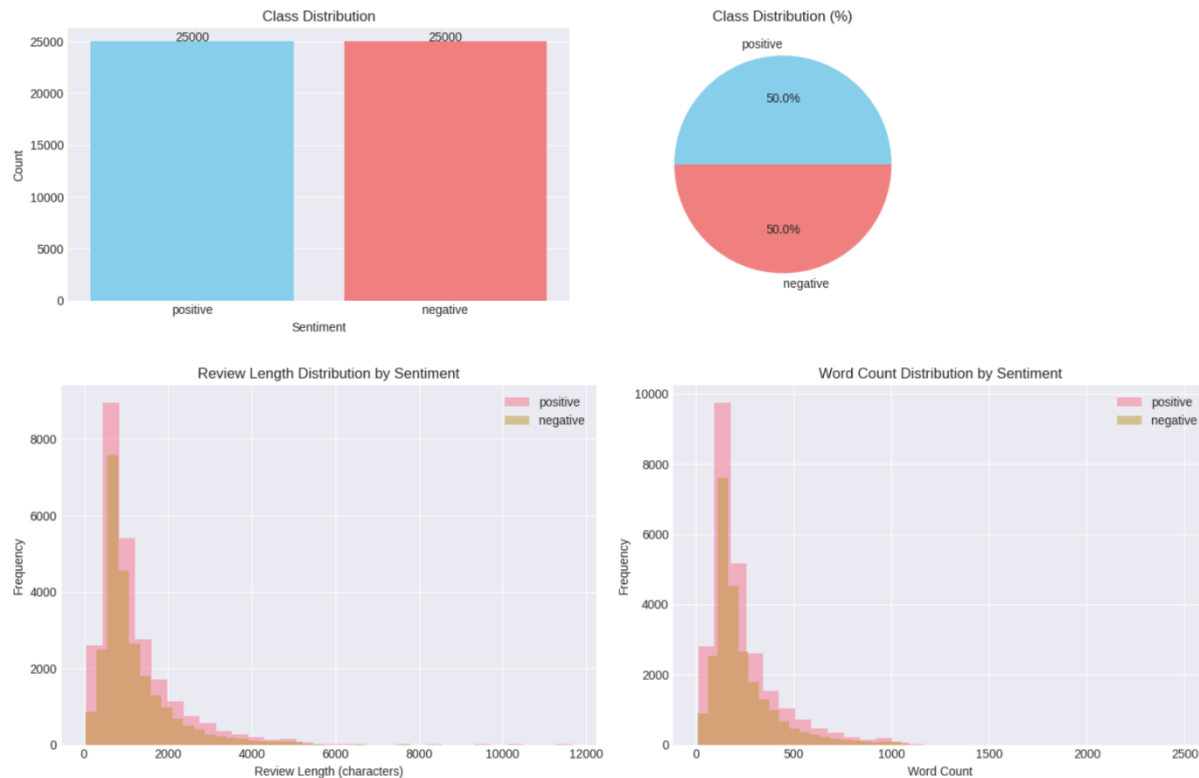


Figure 1: IMDB movie Exploratory Data Analysis

3.2 Text Preprocessing Pipeline

Text preprocessing is one of the most important stages of unstructured text data processing to be used in machine learning. We use our preprocessing pipeline, which adheres to several transformations that aim to normalize text and not to lose information, which is sentiment-relevant:

1. **Conversion to lowercase:** All the text is changed to lowercase so that there is uniform representation as well as so the model did not consider the same word as different because of the use of capitalization.
2. **HTML Tag Removal:** Movie reviews have sometimes had HTML formatting tags which are stripped away to deem all unnecessary markup that may be misleading to the model.
3. **Removal of Special Character and Digit:** Non-alphabetic characters are dropped since punctuation and digits do not add much signal in the classification of sentiment and dimensionality of features is also increased.
4. **Tokenization:** The Natural language toolkit (NLTK) tokenizer splits text into distinct tokens (words) so that it can be processed further.

5. **Stopwords Removal:** Typical English stopwords (e.g., the, is, at, etc.) are eliminated, since they are common in all documents and do not give much discriminative data on the sentiment classification.
6. **Lemmatization:** Since a word can have morphological variations (e.g. running, ran, run) desired results are that morphological variants of the word be treated by WordNet lemmatization as the same feature.

3.3 Traditional Machine Learning Model Design

The classical method uses TF-IDF-vectorization and Logistic Regression as the classification method. This combination is an established starting point of text classification tasks that provides interpretable features importance and probabilistic predictors.

1. **TF-IDF Vectorization:** TF-IDF weighting scheme has the name frequency-inverse document frequency, which is used to turn raw text into numerical feature vectors that can be used in machine learning algorithms.
2. **Logistic regression:** It is a supervised machine learning model designed to do binary classification, i.e. predict the likelihood of a categorical event occurring (e.g. yes/no 0/1) given independent factors.

3.4 Transformer-Based Model Design

Transformer approach Transformer-based approach uses the DistilBERT, which is a distilled version of BERT that still achieves 95% of the performance of BERT at a 40% smaller model (Areshey and Mathkour, 2024). The use of DistilBERT was preferred to full BERT to trade off the performance and computational efficiency, and thus the model is more feasible to use in actual deployment.

1. **Tokenization and Input Representation:** DistilBERT uses WordPiece algorithm to split text into subword units which are what it uses. The tokenizer encodes text to input IDs and produces attention masks of what the tokens are supposed to be attended to. A maximum length of sequence was set to 256 tokens, which is a compromise between the memory of information and computational capabilities.
2. **Model Architecture:** DistilBERT has 6 layers of transformer (as compared to BERT which has 12), 768 hidden dimensions per layer, and 12 attention heads. It was fine-tuned on binary sentiment classification by replacing its final classification layer with a task-specific output layer on the pre-trained model.

3.5 Evaluation Framework

Evaluation Metrics: There were several complementary metrics used to evaluate performance of the model:

1. Accuracy: The proportion of right predictions of all predictions.
2. Precision: This is a percentage of the positive predictions that are correct.
3. Recall: Percent of true positives that have been correctly picked.
4. F1-Score Precision and recall Molten mean.
5. Confusion Matrix: Detailed statistics of correct and wrong predictions in the classes.

Interpretability Analysis: SHAP (SHapley Additive explanations) was applied to calculate feature scores of importance on the Logistic Regression model. The model attention in the case of BERT was determined by attention visualisation through the number of tokens that were predicted. These interpretability tools may be utilised in comparison of decision-making mechanism in various architectures.

4. Results and Evaluation

4.1 Comparative Performance Analysis

Table 1: Comparative Performance Metrics

Metric	Logistic Regression	DistilBERT	Improvement
Accuracy	0.8909	0.9052	+1.61%
F1-Score	0.8909	0.9052	+1.61%
Precision	0.8913	0.9052	+1.56%
Recall	0.8909	0.9052	+1.61%
Training Time	0.37s	605.85s	+163,700%

This small 1.61% accuracy increase is at a high computational cost and the DistilBERT model would need a full 1,637 times longer to train. The implications of this trade-off on deployment scenarios include situations in which computational resources are limited, and situations in which training time is limited.

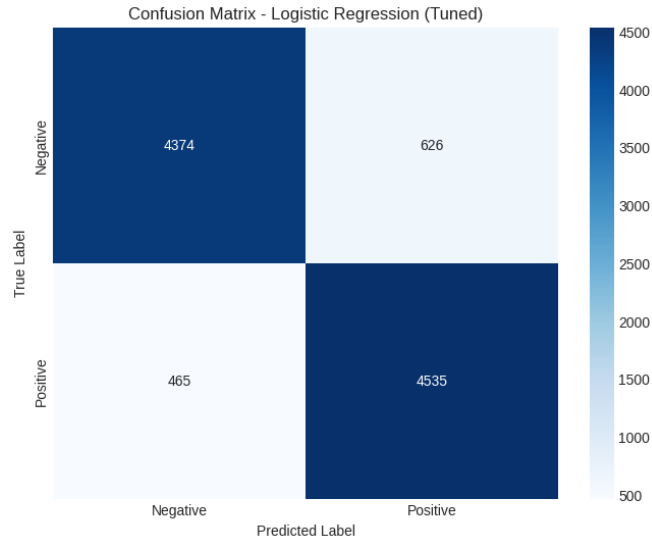


Figure 2: Logistic Regression Confusion Matix

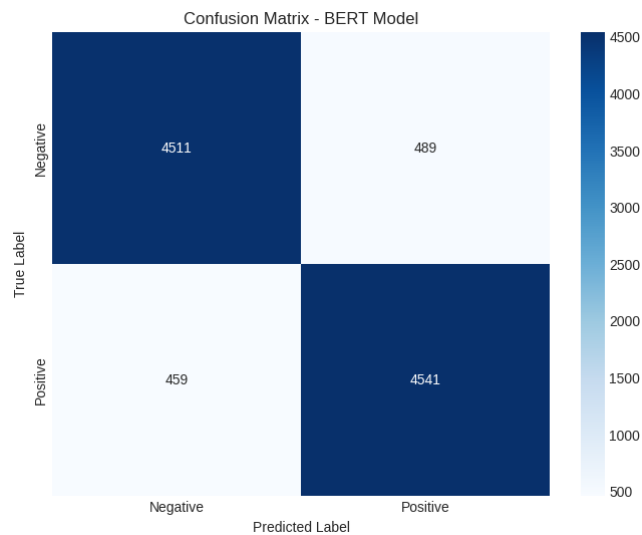


Figure 3: BERT Model Confusion Matrix

4.2 Feature Importance Analysis

The SHAP analysis of the Logistic Regression model demonstrates that there are unique linguistic patterns that are related to positive and negative sentiment. Figure 4 shows the strong association words with each sentiment category.

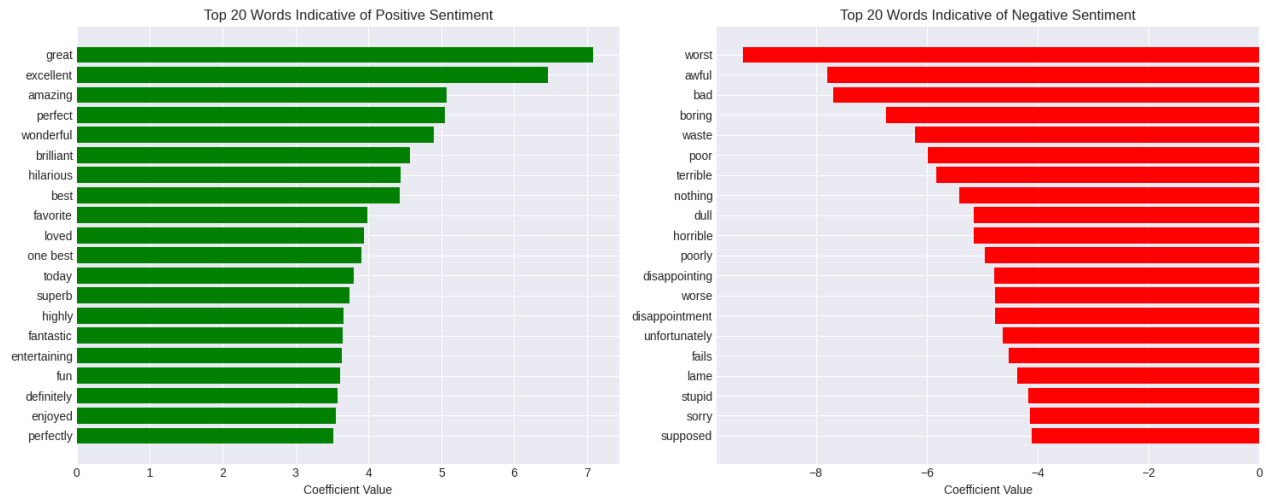


Figure 4: Feature Importance Analysis

These coefficients show some intuitive trends: positive sentiment words are used to express enthusiasm and quality (great, excellent, amazing), whereas negative sentiment ones are used to express criticism and disappointment (worst, awful, boring). The size of coefficients indicates the intensity of association, and normally negative terms tend to have larger absolute coefficients, and this implies that a negative sentiment can be conveyed using more unique vocabulary.

5. Discussion

The trade-off between model complexity and practical utility in sentiment analysis is identified in the comparative analysis as the relatively small difference in accuracy between transformer-based models (0.61%) is traded off against (1,637) times longer training times, (1,637) times slower inference times, and reduced interpretability compared to more traditional models (Logistic Regression). Although transformers are incredibly effective at modelling subtle contextual dependencies using self-attention and attention is a key feature where their performance is superior to other models in language-related tasks, their performance improvement on binary movie review classification indicates that more basic lexical cues can easily resolve well-defined sentiment problems. This brings up critical theoretical and practical concerns: Logistic Regression can give clear, SHAP-explainable feature importance that is vital to regulated industries, but transformer black boxes depend on attention visualisation that can give little understanding of the way they make decisions. The efficiency and explainability of Logistic Regression may be tolerable compared to the marginal accuracy benefits of transformers in resource-constrained applications, real-time applications, or even explainability-critical applications, although hybrid architectures or newer, efficient versions of transformers can provide a balance tradeoff between

accuracy and less prohibitive constraints of computational resources and environmental impact (Strubell et al., 2019).

6. Conclusion and Recommendations

6.1 Summary of Findings and Contributions

This general comparison reveals that, despite the fact that DistilBERT is a little more effective (90.52% accuracy) than the performance of the Logistic Regression (89.09% accuracy), the difference (1.61% difference) is at a very high cost an increase in the training time by a 1,637-fold factor and extremely slow inference. The research provides rigid empirical data defining performance standards of both approaches and elicits trade-offs underlying between the complexity and utility of models. SHAP and attention visualisation enable us to establish that Logistic Regression enables interpretable features of importance to be found with a level of understanding that is needed in regulated industries, and that transformer models are black boxes, which are hard to debug and validate. These results introduce a methodological implementation that comprises of combination of performance measures, computational analysis and interpretability evaluation that can even be replicated in a later study under other domains and tasks.

6.2 Recommendations for Practice and Future Research

The organisations are recommended to apply the Logistic Regression in live, memory-intensive, or explainable models where the marginal accuracy improvements are not significant, and the algorithm performance (89% accuracy) and the explainability are of greater concern. The latter recommends the use of DistilBERT where accuracy is critical and when the CPU resources and faster inference are available. The hybrid architectures featuring balanced solutions are offered through transformer fallback, where Logistic Regression is used as a fast baseline on the cases of ambiguity.

In the future work Add both of the traditional and transformer baselines just to give some context on what is suggested, summary of computational needs with accuracy measures and add interpretability analysis to learn what is going on within the model. The research directions of criticism are domain adaptation to a wide range of text types, multilingual sentiment analysis in which transformer pre-training can be more beneficial, better explainability methods of neural models, efficient architectures where model compression can be more practical and systematic study of demographic bias in sentiment classification. The studies will explore the theoretical foundation of sentiment analysis applications as well as responsible application of the technology to ensure its fair use.

References

- Abiodun, E.O. *et al.* (2021) "A systematic review of emerging feature selection optimization methods for optimal text classification: the present state and prospective opportunities," *Neural Computing and Applications*, 33(22), pp. 15091–15118. Available at: <https://doi.org/10.1007/s00521-021-06406-8>.
- Ahanin, Z. and Ismail, M.A. (2022) "A multi-label emoji classification method using balanced pointwise mutual information-based feature selection," *Computer Speech & Language*, 73, p. 101330.
- Alamanda, S., Pabboju, S. and Gugulothu, N. (2023) "Authorship Attribution using Sequential Part-of-Speech Pattern Mining," *Revue d'Intelligence Artificielle*, 37(1). Available at: <https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnI=0992499X&AN=163153535&h=LVjMDfKxqZlTetcBxhaj3JMdKGBNXf5DvT%2BLhEqI577NEDWgQQz%2B1EZ3H03EGgQo1XtpXKLC4Y8Ku3oO%2BT8aCg%3D%3D&crl=c> (Accessed: March 15, 2026).
- Areshey, A. and Mathkour, H. (2024) "Exploring transformer models for sentiment classification: A comparison of BERT , ROBERTA , ALBERT , DISTILBERT , and XLNET," *Expert Systems*, 41(11), p. e13701. Available at: <https://doi.org/10.1111/exsy.13701>.
- Beigi, O.M. and Moattar, M.H. (2021) "Automatic construction of domain-specific sentiment lexicon for unsupervised domain adaptation and sentiment classification," *Knowledge-Based Systems*, 213, p. 106423.
- Bhavani, A. and Kumar, B.S. (2021) "A review of state art of text classification algorithms," *2021 5th international conference on computing methodologies and communication (ICCMC)*. IEEE, pp. 1484–1490. Available at: <https://ieeexplore.ieee.org/abstract/document/9418262/> (Accessed: March 15, 2026).
- Bilianos, D. and Mikros, G. (2023) "Sentiment analysis in cross-linguistic context: How can machine translation influence sentiment classification?," *Digital Scholarship in the Humanities*, 38(1), pp. 23–33.
- Danyal, M. *et al.* (2023) "Sentiment analysis based on performance of linear support vector machine and multinomial naïve bayes using movie reviews with baseline techniques," *Journal on Big Data*, 5, p. 1.
- Eggen, M., Lysnæs-Larsen, J. and Strümke, I. (2025) "Integrating attention into explanation frameworks for language and vision transformers." arXiv. Available at: <https://doi.org/10.48550/arXiv.2508.08966>.
- Galassi, A., Lippi, M. and Torroni, P. (2020) "Attention in natural language processing," *IEEE transactions on neural networks and learning systems*, 32(10), pp. 4291–4308.
- Gardazi, N.M. *et al.* (2025) "BERT applications in natural language processing: a review," *Artificial Intelligence Review*, 58(6), p. 166. Available at: <https://doi.org/10.1007/s10462-025-11162-5>.
- Huang, Y. *et al.* (2024) "Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive Survey." arXiv. Available at: <https://doi.org/10.48550/arXiv.2311.12351>.
- Ihejirika, C.J. (2025) "Explainable artificial intelligence in deep learning models for transparent decision-making in high-stakes applications," *Int J Res Publ Rev*, 6, pp. 1924–40.

Johnson, S.J., Murty, M.R. and Navakanth, I. (2023) "A detailed review on word embedding techniques with emphasis on word2vec," *Multimedia Tools and Applications*, 83(13), pp. 37979–38007. Available at: <https://doi.org/10.1007/s11042-023-17007-z>.

Kholwal, R. (2023a) "Text-Classify: A comprehensive comparative study of logistic regression, random forest, and knn models for enhanced text classification performance," *International Journal of Advances in Engineering & Technology*, 16(5), pp. 415–433.

Kholwal, R. (2023b) "Text-Classify: A comprehensive comparative study of logistic regression, random forest, and knn models for enhanced text classification performance," *International Journal of Advances in Engineering & Technology*, 16(5), pp. 415–433.

Li, J. (2023) "Fine-grained sentiment analysis with a fine-tuned BERT and an improved pre-training BERT," *2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA)*. IEEE, pp. 1031–1034. Available at: <https://ieeexplore.ieee.org/abstract/document/10257673/> (Accessed: March 15, 2026).

Maas, A.L. et al. (2011) "Learning Word Vectors for Sentiment Analysis," in D. Lin, Y. Matsumoto, and R. Mihalcea (eds.) *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. ACL-HLT 2011*, Portland, Oregon, USA: Association for Computational Linguistics, pp. 142–150. Available at: <https://aclanthology.org/P11-1015/> (Accessed: March 15, 2026).

Mends Karen, Y.O., Anokwuru, E.A. and Igba, E. (2025) "Integrating Real World Data and Real World Evidence into Commercial Product Strategy: A Predictive Framework for Pharmaceutical Market Forecasting." Available at: https://www.researchgate.net/profile/Karen-Mends/publication/398737980_Integrating_Real_World_Data_and_Real_World_Evidence_into_Commercial_Product_Strategy_A_Predictive_Framework_for_Pharmaceutical_Market_Forecasting/links/695bf2057e61d05b53172e59/Integrating-Real-World-Data-and-Real-World-Evidence-into-Commercial-Product-Strategy-A-Predictive-Framework-for-Pharmaceutical-Market-Forecasting.pdf (Accessed: March 15, 2026).

Nguyen, B., Nguyen, V.-H. and Ho, T. (2021) "Sentiment analysis of customer feedbacks in online food ordering services," *Business Systems Research: International Journal of the Society for Advancing Innovation and Research in Economy*, 12(2), pp. 46–59.

Umer, M. et al. (2023) "Impact of convolutional neural network and FastText embedding on text classification," *Multimedia Tools and Applications*, 82(4), pp. 5569–5585. Available at: <https://doi.org/10.1007/s11042-022-13459-x>.

Yang, M. et al. (2023) "Deep neural networks with L1 and L2 regularization for high dimensional corporate credit risk prediction," *Expert Systems with Applications*, 213, p. 118873.

Zucco, C. et al. (2020) "Sentiment analysis for mining texts and social networks data: Methods and tools," *WIREs Data Mining and Knowledge Discovery*, 10(1), p. e1333. Available at: <https://doi.org/10.1002/widm.1333>.

Appendices

```
# Install required packages
```

```

!pip install kagglehub transformers datasets torch scikit-learn
matplotlib seaborn pandas numpy wordcloud shap
!pip install transformers[torch]
!pip install accelerate -U

# Import libraries
import kagglehub
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
import warnings
warnings.filterwarnings('ignore')

# Download NLTK data
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('punkt_tab')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

# Set style for plots
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")

print("Dataset Columns:")
print(df.columns.tolist())
print("\nFirst few rows of the dataset:")
print(df.head())
print("\nDataset Info:")
print(df.info())

# Check class distribution
print("\nClass Distribution:")
sentiment_counts = df['sentiment'].value_counts()
print(sentiment_counts)

# Visualize class distribution
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

```

```

# Bar plot
axes[0].bar(sentiment_counts.index, sentiment_counts.values,
color=['skyblue', 'lightcoral'])
axes[0].set_title('Class Distribution')
axes[0].set_xlabel('Sentiment')
axes[0].set_ylabel('Count')
for i, v in enumerate(sentiment_counts.values):
    axes[0].text(i, v + 10, str(v), ha='center')

# Pie chart
axes[1].pie(sentiment_counts.values, labels=sentiment_counts.index,
autopct='%1.1f%%',
        colors=['skyblue', 'lightcoral'])
axes[1].set_title('Class Distribution (%)')

plt.tight_layout()
plt.show()

# Basic text statistics
df['review_length'] = df['review'].str.len()
df['word_count'] = df['review'].str.split().str.len()

# Plot text length distribution by sentiment
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Review length distribution
for sentiment in df['sentiment'].unique():
    subset = df[df['sentiment'] == sentiment]
    axes[0].hist(subset['review_length'], alpha=0.5, label=sentiment,
bins=30)
axes[0].set_xlabel('Review Length (characters)')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Review Length Distribution by Sentiment')
axes[0].legend()

# Word count distribution
for sentiment in df['sentiment'].unique():
    subset = df[df['sentiment'] == sentiment]
    axes[1].hist(subset['word_count'], alpha=0.5, label=sentiment,
bins=30)
axes[1].set_xlabel('Word Count')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Word Count Distribution by Sentiment')
axes[1].legend()

```

```

plt.tight_layout()
plt.show()

print(f"\nAverage review length: {df['review_length'].mean():.2f}
characters")
print(f"Average word count: {df['word_count'].mean():.2f} words")
print(f"Min word count: {df['word_count'].min()}")
print(f"Max word count: {df['word_count'].max()}")

# Additional statistics
print(f"\nReview length statistics:")
print(df.groupby('sentiment')['review_length'].describe())

print(f"\nWord count statistics:")
print(df.groupby('sentiment')['word_count'].describe())

# Create TF-IDF features
print("Creating TF-IDF features...")
tfidf_vectorizer = TfidfVectorizer(max_features=10000, ngram_range=(1,
2), min_df=5, max_df=0.7)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

print(f"TF-IDF training matrix shape: {X_train_tfidf.shape}")
print(f"TF-IDF test matrix shape: {X_test_tfidf.shape}")
print(f"Number of features:
{len(tfidf_vectorizer.get_feature_names_out())}")

# Train Logistic Regression model
print("\nTraining Logistic Regression model...")
start_time = time.time()
lr_model = LogisticRegression(max_iter=1000, random_state=42, C=1.0,
class_weight='balanced')
lr_model.fit(X_train_tfidf, y_train)
lr_train_time = time.time() - start_time

# Make predictions
lr_train_pred = lr_model.predict(X_train_tfidf)
lr_test_pred = lr_model.predict(X_test_tfidf)

# Calculate metrics
lr_train_accuracy = accuracy_score(y_train, lr_train_pred)
lr_test_accuracy = accuracy_score(y_test, lr_test_pred)
lr_f1 = f1_score(y_test, lr_test_pred, average='weighted')

```

```

lr_precision = precision_score(y_test, lr_test_pred, average='weighted')
lr_recall = recall_score(y_test, lr_test_pred, average='weighted')

print(f"\nLogistic Regression Results:")
print(f"Training time: {lr_train_time:.2f} seconds")
print(f"Training Accuracy: {lr_train_accuracy:.4f}")
print(f"Test Accuracy: {lr_test_accuracy:.4f}")
print(f"Test F1-Score: {lr_f1:.4f}")
print(f"Test Precision: {lr_precision:.4f}")
print(f"Test Recall: {lr_recall:.4f}")

# Detailed the classification report, I mean which model has better
performance
print("\nClassification Report:")
print(classification_report(y_test, lr_test_pred,
target_names=['Negative', 'Positive']))

# Train the model
print("Starting BERT training...")
start_time = time.time()

# Train
trainer.train()

bert_train_time = time.time() - start_time
print(f"\nBERT Training completed in {bert_train_time:.2f} seconds
({bert_train_time/60:.2f} minutes)")

# Evaluate
print("\nEvaluating BERT model...")
eval_results = trainer.evaluate()
print(f"\nBERT Evaluation Results:")
for key, value in eval_results.items():
    print(f"{key}: {value:.4f}")

# Save the model
model.save_pretrained('./bert_sentiment_model')
tokenizer.save_pretrained('./bert_sentiment_model')
print("\nModel saved to './bert_sentiment_model'")

# Get predictions
print("Making predictions with BERT...")
predictions = trainer.predict(tokenized_test)

```

```

y_pred_bert = np.argmax(predictions.predictions, axis=1)

# Calculate metrics
bert_accuracy = accuracy_score(y_test[:len(y_pred_bert)], y_pred_bert)
bert_f1 = f1_score(y_test[:len(y_pred_bert)], y_pred_bert,
average='weighted')
bert_precision = precision_score(y_test[:len(y_pred_bert)], y_pred_bert,
average='weighted')
bert_recall = recall_score(y_test[:len(y_pred_bert)], y_pred_bert,
average='weighted')

print(f"\nBERT Test Results:")
print(f"Accuracy: {bert_accuracy:.4f}")
print(f"F1-Score: {bert_f1:.4f}")
print(f"Precision: {bert_precision:.4f}")
print(f"Recall: {bert_recall:.4f}")

# Classification report
print("\nBERT Classification Report:")
print(classification_report(y_test[:len(y_pred_bert)], y_pred_bert,
target_names=['Negative', 'Positive']))

# Confusion matrix for BERT
plot_confusion_matrix(y_test[:len(y_pred_bert)], y_pred_bert, 'BERT
Model')

# Create comparison dataframe
comparison_data = {
    'Model': ['Logistic Regression (TF-IDF)', 'BERT (DistilBERT)'],
    'Accuracy': [lr_tuned_accuracy, bert_accuracy],
    'F1-Score': [lr_tuned_f1, bert_f1],
    'Precision': [lr_tuned_precision, bert_precision],
    'Recall': [lr_tuned_recall, bert_recall],
    'Training Time (s)': [lr_train_time, bert_train_time]
}

comparison_df = pd.DataFrame(comparison_data)
print("="*80)
print("MODEL PERFORMANCE COMPARISON")
print("="*80)
print(comparison_df.to_string(index=False))

# Visualize comparison
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

```

```

# Metrics comparison
metrics = ['Accuracy', 'F1-Score', 'Precision', 'Recall']
x = np.arange(len(metrics))
width = 0.35

axes[0].bar(x - width/2, comparison_df.iloc[0][1:5], width,
label='Logistic Regression', color='skyblue')
axes[0].bar(x + width/2, comparison_df.iloc[1][1:5], width,
label='BERT', color='lightcoral')
axes[0].set_xlabel('Metrics')
axes[0].set_ylabel('Score')
axes[0].set_title('Model Performance Comparison')
axes[0].set_xticks(x)
axes[0].set_xticklabels(metrics)
axes[0].legend()
axes[0].set_ylim(0, 1)

# Add value labels on bars
for i, v in enumerate(comparison_df.iloc[0][1:5]):
    axes[0].text(i - width/2, v + 0.01, f'{v:.3f}', ha='center',
va='bottom', fontsize=9)
for i, v in enumerate(comparison_df.iloc[1][1:5]):
    axes[0].text(i + width/2, v + 0.01, f'{v:.3f}', ha='center',
va='bottom', fontsize=9)

# Training time comparison
axes[1].bar(['Logistic Regression', 'BERT'],
            [lr_train_time, bert_train_time],
            color=['skyblue', 'lightcoral'])
axes[1].set_ylabel('Training Time (seconds)')
axes[1].set_title('Training Time Comparison')

# Add value labels
axes[1].text(0, lr_train_time + 5, f'{lr_train_time:.1f}s', ha='center',
va='bottom')
axes[1].text(1, bert_train_time + 5, f'{bert_train_time:.1f}s',
ha='center', va='bottom')

# Sample size comparison
sample_sizes = [len(X_train), len(X_test)]
axes[2].bar(['Training Set', 'Test Set'], sample_sizes,
color=['lightgreen', 'lightblue'])
axes[2].set_ylabel('Number of Samples')
axes[2].set_title('Dataset Split')

```

```
for i, v in enumerate(sample_sizes):  
    axes[2].text(i, v + 100, str(v), ha='center', va='bottom')  
  
plt.tight_layout()  
plt.show()
```