
King Saud University

College of Sciences

# Statistics Is All You Need:

## LLMs vs Traditional Statistical Models

By Abdulrahman Alkurayshan

Student Number: 443102096

Supervised by: Dr. Aaid Algahtani

Committee Member: Prof. Abdulhamid Alzaid

Committee Member: Dr. Mohamed Abdelkader

2025

# Acknowledgements

أحبُّ أن أؤكّد عِظَمَ استشعارِ حديثِ النبيّ ﷺ: «مَن لا يشكرُ الناسَ لا يشكرُ الله»؛ إذ يضعُ الشكرَ في موضعه الحقيقي، ولهذا أحمدُ اللهَ أولًا وآخرًا على نعمه التي لا تُحصى، وما توفيقي إلا به سبحانه جلّ وعلا.

ويَسبق الجميعَ في الشكر والامتنان والديّ العزيزان؛ رحمَ اللهُ أمي وأسكنها فسيح جناته، وأسألُ الله أن يجعل هذا العمل صدقةً جاريةً لها، فهي الحاضرُ الغائبُ في كل إنجازٍ أخطّه. كما أشكرُ والدي على دعمه اللامحدود وعطائه الصادق، الذي كان – بعد عون الله – سببًا في استمراري وتقدّمي؛ فلم يبخل عليّ يومًا.

ولا يفوتني أن أثني على إخوتي الأحبة: **حمود، وعبدالله، وعبدالعزيز**؛ فقد كانوا خيرَ السند والرفقة في هذه المسيرة، وكان لتشجيعهم أثرٌ كبيرٌ في نفسي.

ثم أتوجَّه بخالص التقدير والعرفان لسعادة الدكتور **عايض القحطاني**، لحرصه واهتمامه بالمشروع، وتوجيهاته العميقة، ودعمه المعنوي المستمر، الذي كان له فضلٌ كبيرٌ في ظهور هذا العمل بأجمل صورة.

كما أتقدم بجزيل الشكر لأساتذة قسم الإحصاء الذين لم يبخلوا بعلمهم وتوجيهاتهم طوال سنوات الدراسة؛ فقد تعلّمتُ منهم ما يبقى أثرُه طويلًا، فجزاهم الله عني خيرَ الجزاء.

وأخصُّ بالشكر السادةَ مناقشي المشروع على وقتهم الثمين، وملاحظاتهم الجوهرية التي أثرت العمل ووجّهته نحو الأفضل:

- الأستاذ الدكتور **عبدالحميد الزيد**
- الدكتور **محمد عبدالقادر**

كما أرفعُ خالص امتناني لكل من كان له دورٌ، صغيرًا كان أو كبيرًا، في دعمي وتشجيعي من الأساتذة والزملاء والأصدقاء، فوجودهم كان عونًا لي بعد الله في إتمام هذا العمل.

كما أتوجَّه بالشكر للدكتور **عبدالله آل ثنيان.**

كما لا أنسى أن أشكر صديقي العزيز **صالح المحمود**، رفيق الدرب في هذا التخصص، وشريك الطريق في العلم والتحديات؛ فله مني كل الامتنان والتقدير.

وفي الختام، أسألُ الله أن يجعل هذا العمل خالصًا لوجهه الكريم، نافعًا لوطني ولمن يطّلع عليه، وأن يكون بدايةً لما هو أعظم. فإن أصبتُ فمن الله وحده، وإن أخطأتُ فمن نفسي والشيطان. وأسأله سبحانه أن يكتب لهذا الجهد القبول، ولمن ساهم في دعمي الأجر والثواب.

# Contents

# List of Figures

# List of Tables

**Abstract**

This thesis investigates whether Large Language Models (LLMs) can outperform traditional statistical models in Arabic text classification and sentiment analysis. Using a real-world case study on Saudi bank tweets, we compare a fine-tuned BERT model with a TF–IDF Logistic Regression baseline under a controlled experimental framework. Both models are analyzed through a unified statistical lens—mapping input features to logits and probabilities via softmax—yet they fundamentally differ in their representation of text: sparse, manually engineered features versus context-rich embeddings learned from large corpora.

The results show that BERT achieves superior performance, particularly on minority classes such as *Neutral* and *Positive* tweets, where contextual understanding is essential. Logistic Regression, however, remains competitive, computationally efficient, and interpretable—highlighting the continued relevance of classical models in practical settings. These findings validate the hypothesis that LLMs provide meaningful gains over traditional approaches, while also demonstrating that statistical thinking remains central to understanding model behavior. The work concludes that modern NLP is not a replacement for statistics—but rather an extension of it.

# Overview

Text classification has developed from classical statistical methods into modern language modeling approaches capable of representing meaning and context. Early techniques such as Logistic Regression, Support Vector Machines (SVM), and Naïve Bayes were widely used with sparse text representations, including bag-of-words and term frequency–inverse document frequency (TF–IDF) [1]. These models were computationally efficient and transparent, allowing analysts to interpret the role of each variable through coefficients and weights. However, they treated text as unordered word counts, overlooking sentence structure and contextual relationships.

The introduction of distributed word representations brought an important improvement. Models such as Word2Vec [2] and GloVe [3] represented words as continuous numerical vectors, enabling the measurement of semantic similarity between terms. These embeddings reduced data sparsity and allowed statistical models to capture patterns that simple frequency-based methods could not. Nevertheless, the embeddings were fixed for each word and unable to reflect meaning changes across different contexts.

Recurrent neural networks (RNNs) and long short-term memory (LSTM) models [5] further advanced sequential text analysis by incorporating word order and contextual dependencies. Yet, these methods were computationally demanding and often struggled with long sentences. The development of the Transformer architecture by Vaswani et al. [6] provided a more efficient alternative through its self-attention mechanism, which allows the model to consider relationships between all words in a sequence simultaneously.

Large pre-trained models based on the Transformer framework—such as BERT [7] and GPT [8]—extended these ideas by learning language patterns from extensive text corpora before being fine-tuned for specific applications. This approach led to substantial performance gains in many natural language processing tasks, including text classification and sentiment analysis [11]. In

Arabic text analysis, specialized models such as AraBERT have shown notable improvements over traditional classifiers on benchmark datasets [9, 10].

Despite these advancements, key considerations remain. Classical statistical models continue to offer clarity, simplicity, and ease of interpretation—properties that are essential in applied research and policy analysis. In contrast, large neural models, while powerful, can be complex to interpret and computationally intensive to apply [12].

In this study, we build on both perspectives by comparing Logistic Regression, representing the traditional statistical approach, with a fine-tuned BERT model, representing the contemporary language modeling approach. The analysis focuses on Arabic sentiment data to examine how each method handles linguistic variability, contextual understanding, and classification accuracy. This comparison aims to highlight the strengths and trade-offs between interpretable statistical modeling and modern context-based representations within the same analytical framework.

# 1 Introduction

Text classification is a fundamental task in natural language processing with applications ranging from sentiment analysis to spam filtering. Traditionally, text classification has been approached with statistical models that rely on handcrafted features such as word frequency or $n$-grams. These models, particularly **logistic regression**, are relatively simple and interpretable but often consider only shallow lexical patterns. Recent advances in deep learning have introduced *Large Language Models (LLMs)*, which learn language representations from very large corpora and can capture nuanced context and semantics. This thesis explores whether LLMs can leverage their rich representations to outperform traditional methods in classifying text data.

Despite the success of statistical models in many settings, they are limited by their reliance on sparse features and an inability to fully capture context (for example, bag-of-words models lose word order and context). LLMs, by contrast, use architectures like Transformers that consider word sequences as a whole, enabling them to interpret meaning in context. This capability suggests that LLMs may significantly improve classification accuracy and generalization, especially for complex language understanding tasks.

The central question of this research is: *Do large language models outperform traditional statistical models in text classification, and if so, why?* We hypothesize that the deep contextual learning of LLMs leads to better performance than the surface-level pattern recognition of older models. To investigate this, we conduct a comparative experiment using an Arabic-language sentiment analysis dataset as a case study. We explicitly state our hypotheses here for clarity:

- **Research Hypothesis:** Large Language Models (LLMs) outperform traditional statistical models in text classification tasks due to their ability to capture contextual and semantic representations beyond surface-level statistical patterns.

- **Null Hypothesis ($H_0$):** There is no significant difference in classification performance between LLMs and traditional statistical models.
- **Alternative Hypothesis ($H_1$):** LLMs achieve significantly higher classification accuracy and generalization performance compared to traditional statistical models.

In order to test these hypotheses, we fine-tune a Transformer-based LLM (a BERT model) and train a single conventional baseline, **Logistic Regression**, on the same text dataset. We then evaluate their performances using multiple metrics and statistical significance tests. A positive finding in favor of the LLM would support the view that its capacity for contextual understanding drives improved outcomes. Conversely, if no substantial difference is found, it would suggest that traditional linear baselines remain competitive for this task.

The remainder of this thesis is organized as follows. Chapter 2 reviews relevant background and literature, covering the development and capabilities of LLMs as well as the principles of traditional text classification models. Chapter 3 describes the methodology of our experiment, including the dataset, preprocessing steps, model configurations, and evaluation protocol. Chapter 4 presents the results of the comparative experiments and provides an in-depth analysis of the LLM's behavior (through an examination of embedding and attention mechanisms) alongside error analysis. Finally, Chapter 5 concludes the thesis by summarizing the findings, discussing their implications, and suggesting directions for future work.

# 2 Large Language Models (LLMs)

### Definition and Significance

Large Language Models are advanced machine learning architectures designed to process, understand, and generate natural language by learning complex patterns from extensive text corpora.

Formally, an LLM models the probabilistic structure of language by estimating distributions over sequences of tokens (words or subword units). This enables LLMs to perform a wide range of natural language processing (NLP) tasks, including text classification, generation, translation, and summarization. Crucially, LLMs implicitly learn semantic and syntactic relationships from raw text, making them indispensable in modern NLP—especially for applications requiring nuanced contextual understanding that goes beyond the capabilities of traditional approaches.

**Historical Development**

The field of language modeling has evolved from simple count-based methods to sophisticated neural architectures. Early models, such as $n$**-gram models**, estimated the probability of a word given its preceding context using frequency counts. Under the Markov assumption, the probability of the next word $w_t$ given the previous $n-1$ words can be approximated as:

$$P(w_t \mid w_{t-n+1}, \ldots, w_{t-1}) \approx \frac{\text{Count}(w_{t-n+1}, \ldots, w_t)}{\text{Count}(w_{t-n+1}, \ldots, w_{t-1})},$$

where occurrences of word sequences are counted in a corpus. While effective for short contexts, $n$-gram models suffer from data sparsity and use fixed-size context windows, limiting their ability to capture long-range dependencies.

The advent of deep learning brought **recurrent neural networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks, which improved sequence modeling by maintaining hidden state information over arbitrary-length sequences. These models could, in principle, preserve long-range information better than $n$-grams. However, RNNs have difficulty with very long contexts and cannot be easily parallelized during training, hindering their scalability.

The introduction of the **Transformer** architecture by [6] was a breakthrough that addressed these issues. Transformers use self-attention mechanisms to learn long-range dependencies efficiently, allowing for parallel computation across sequence elements. Transformer-based models

such as **GPT** (Generative Pre-trained Transformer; [8]) and **BERT** (Bidirectional Encoder Representations from Transformers; [7]) have since become the foundation of state-of-the-art LLMs, revolutionizing NLP tasks with unprecedented performance.

### Architectural Foundations

The Transformer architecture centers on the multi-head self-attention mechanism, which computes context-aware representations of each input token by relating it to every other token in the sequence. A key component is the **scaled dot-product attention**, defined by:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)\, V\,,$$

where $Q$, $K$, and $V$ denote the query, key, and value matrices (respectively) derived from the input. In practice:

- $Q \in \mathbb{R}^{n \times d_k}$ contains $d_k$-dimensional query vectors for each of the $n$ input positions.
- $K \in \mathbb{R}^{n \times d_k}$ contains key vectors for each input position.
- $V \in \mathbb{R}^{n \times d_v}$ contains value vectors for each input position.

The similarity of a pair of tokens $i$ and $j$ is measured by the dot product of $Q_i$ and $K_j$. The softmax function then weights each value $V_j$ according to these normalized similarities, producing an output for each token that emphasizes the tokens most relevant to it.

A Transformer layer consists of multiple attention "heads" that allow the model to attend to different types of linguistic information in parallel. Each head performs attention with independent learned projections of $Q$, $K$, and $V$ (often $d_k = d_v = 64$ for 12-headed models), and their outputs are concatenated and linearly transformed. The architecture also includes **feed-forward** layers applied to each position and residual connections with layer normalization that help stabilize training.

This design enables Transformers to capture complex, long-range patterns in text more effectively than earlier RNN-based approaches, and with greater computational efficiency due to parallelization.

## 2.1 Embeddings: How It Works (with an Illustrative Example)

Embeddings map discrete tokens to dense continuous vectors that encode semantic and syntactic regularities [2, 3]. In encoder-based LLMs (e.g., BERT), the input representation for token $i$ is the sum of token, position, and (optionally) segment embeddings [7]:

$$\mathbf{x}_i = \mathbf{T}_i + \mathbf{P}_i + \mathbf{S}_i \,. \tag{2.1}$$

**Illustrative example** Consider the sentence "I love programming." After WordPiece tokenization [**?**], the sequence is

$$[\text{[CLS]},\ \text{I},\ \text{love},\ \text{programming},\ \text{[SEP]}].$$

For illustration, suppose a reduced embedding dimension $d = 4$. The learned components and resulting input vectors $\mathbf{x}_i$ (via (2.1)) might be:

Table 1: Toy embedding components (dimension $d = 4$) and their sum $\mathbf{x}_i = \mathbf{T}_i + \mathbf{P}_i + \mathbf{S}_i$. Values are illustrative.

| Token | $\mathbf{T}_i$ | $\mathbf{P}_i$ | $\mathbf{S}_i$ | $\mathbf{x}_i$ |
|---|---|---|---|---|
| [CLS] | $[0.10, 0.20, 0.05, 0.00]$ | $[0.00, 0.01, 0.02, 0.03]$ | $[0, 0, 0, 0]$ | $[0.10, 0.21, 0.07, 0.03]$ |
| I | $[0.30, 0.60, 0.50, 0.10]$ | $[0.01, 0.02, 0.03, 0.04]$ | $[0, 0, 0, 0]$ | $[0.31, 0.62, 0.53, 0.14]$ |
| love | $[0.40, 0.80, 0.60, 0.20]$ | $[0.02, 0.03, 0.04, 0.05]$ | $[0, 0, 0, 0]$ | $[0.42, 0.83, 0.64, 0.25]$ |
| programming | $[0.50, 0.75, 0.65, 0.30]$ | $[0.03, 0.04, 0.05, 0.06]$ | $[0, 0, 0, 0]$ | $[0.53, 0.79, 0.70, 0.36]$ |
| [SEP] | $[0.05, 0.10, 0.05, 0.00]$ | $[0.04, 0.05, 0.06, 0.07]$ | $[0, 0, 0, 0]$ | $[0.09, 0.15, 0.11, 0.07]$ |

Stacking the rows of Table 1 yields the input matrix $X \in \mathbb{R}^{5\times 4}$ that feeds the first Transformer layer.

## 2.2 Self-Attention Mechanism

Self-attention contextualizes each token by aggregating information from all positions [6]. Given query, key, and value matrices $Q, K, V \in \mathbb{R}^{n\times d_k}$ for a sequence of length $n$, a single head computes:

$$\text{Attention}(Q, K, V) \;=\; \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)\, V\,. \tag{2.2}$$

**Numerical toy example (continuing "I love programming.").** Let $d_k = 3$ and (for one head) define learned projections $W_Q, W_K, W_V \in \mathbb{R}^{4\times 3}$. Using the $\mathbf{x}_i$ from Table 1, form $Q = XW_Q$, $K = XW_K$, and $V = XW_V$. Suppose, for the token *I*, the scaled dot products with keys are (illustrative):

$$\big[\, e_{\text{I, [CLS]}} \;\; e_{\text{I, I}} \;\; e_{\text{I, love}} \;\; e_{\text{I, programming}} \;\; e_{\text{I, [SEP]}} \,\big] = [\,0.20,\; 1.04,\; 1.21,\; 0.87,\; 0.15\,].$$

Applying softmax over the five positions yields attention weights

$$\boldsymbol{\alpha}_{\text{I}} = [\,0.12,\; 0.24,\; 0.27,\; 0.23,\; 0.14\,],$$

indicating "I" attends most to "love" and "programming". The contextualized vector is the weighted sum

$$\mathbf{z}_{\text{I}} \;=\; \sum_j \alpha_{\text{I},j}\, \mathbf{v}_j\,,$$

with $\mathbf{v}_j$ being the corresponding row of $V$.

**Multi-head aggregation.** BERT uses multiple heads in parallel; head outputs are concatenated and linearly projected, followed by residual connection and layer normalization, then a position-wise feed-forward network [6]. This allows different heads to specialize (e.g., subject–verb, modifier–head, long-range discourse cues).

Table 2: Illustrative attention weights (one head) for three content tokens. Rows: query token; columns: attended token.

| Query ↓ / Key → | [CLS] | I | love | programming | [SEP] |
|---|---|---|---|---|---|
| I | 0.12 | 0.24 | **0.27** | 0.23 | 0.14 |
| love | 0.10 | 0.18 | 0.16 | **0.41** | 0.15 |
| programming | 0.08 | 0.14 | **0.49** | 0.19 | 0.10 |

## 2.3 From Attention to Classification

For sequence classification, the final hidden state of the special token `[CLS]` (denoted $\mathbf{h}_{[\text{CLS}]} \in \mathbb{R}^d$) serves as a learned summary of the input [7]. A linear classifier maps it to class logits, then softmax yields probabilities:

$$\mathbf{z} \;=\; W\,\mathbf{h}_{[\text{CLS}]} + \mathbf{b}\,, \qquad \mathbf{p} \;=\; \text{softmax}(\mathbf{z})\,, \tag{2.3}$$

with $W \in \mathbb{R}^{C\times d}$, $\mathbf{b} \in \mathbb{R}^C$, and $C$ classes. Training minimizes cross-entropy:

$$\mathcal{L} \;=\; -\sum_{c=1}^{C} y_c \log p_c\,. \tag{2.4}$$

**Numeric illustration (binary sentiment).** Assume $\mathbf{h}_{[\text{CLS}]} = [0.40,\, 0.70,\, -0.20,\, 0.60]^\top$, two classes (*Positive*, *Negative*), and

$$W = \begin{bmatrix} 1.2 & 0.8 & -0.5 & 1.0 \\ -0.9 & -0.6 & 0.4 & -1.2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}.$$

Then

$$\mathbf{z} = \begin{bmatrix} 2.06 \\ -2.04 \end{bmatrix}, \qquad \mathbf{p} = \text{softmax}(\mathbf{z}) \approx \begin{bmatrix} 0.984 \\ 0.016 \end{bmatrix},$$

thus predicting *Positive* with high confidence for "I love programming."

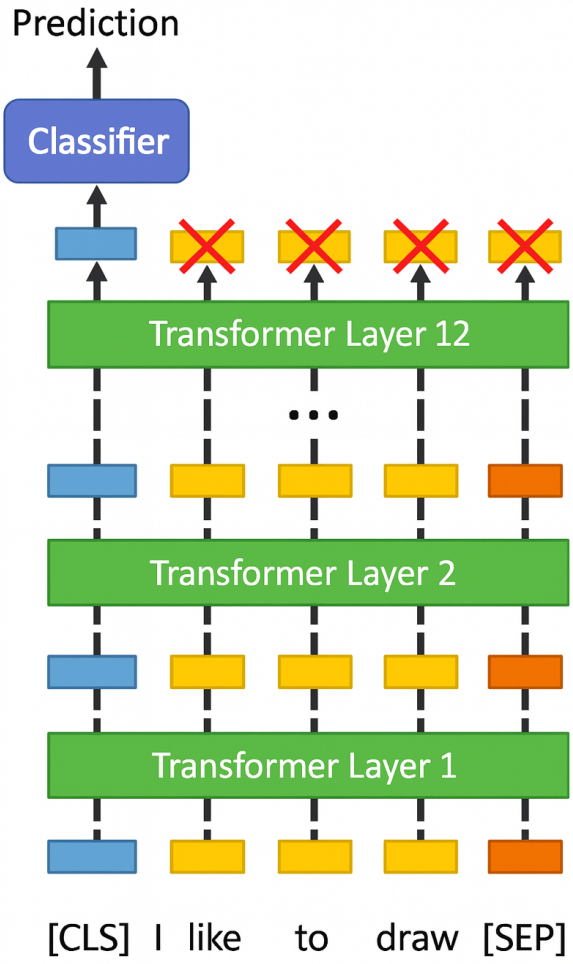
**Interpretability note.** While attention offers some transparency into token interactions, richer interpretability techniques (e.g., probing, gradients, attention rollout) are often used to analyze how $\mathbf{h}_{[\text{CLS}]}$ captures task-relevant information [**?**].

### Applications and Impact

Transformer-based LLMs have achieved remarkable success across diverse NLP tasks. They have set new performance records in text classification, question answering, machine translation, and many other benchmarks. LLMs can capture linguistic information across multiple levels of granularity (from character-level morphology to long-range discourse) □[**?**]□. Their broad knowledge acquired from pre-training on massive corpora allows them to generalize well, even to tasks or domains not explicitly seen during training.

However, the rise of LLMs also comes with challenges. These models often operate as black boxes due to their enormous number of parameters and layered, nonlinear architecture [12]. This lack of interpretability raises concerns in sensitive applications where understanding model decisions is important. Additionally, LLMs demand substantial computational resources for training and inference; training an LLM can require specialized hardware (GPUs/TPUs) and energy consumption far exceeding that of simpler models. There are also ethical considerations: LLMs can inadvertently capture biases present in their training data and may produce inappropriate or incorrect outputs if not carefully controlled.

In summary, LLMs represent a leap in NLP capability, offering superior performance by leveraging context and meaning in ways that traditional models cannot. This thesis focuses on one particular advantage of LLMs—their performance in text classification—by directly comparing an LLM against classic algorithms on the same task.




# 3 Traditional Statistical Models for Text Classification

*Motivation.* Before the emergence of Transformer-based LLMs, text classification was dominated by compact, interpretable statistical models trained on manually engineered features (often sparse representations such as bag-of-words and TF–IDF). These approaches—most notably Naïve Bayes, Logistic Regression, and Support Vector Machines (SVM)—remain valuable baselines and, in

many practical settings, still offer competitive accuracy at far lower computational cost than LLMs. They also provide greater transparency, as their learned parameters can often be directly interpreted.

*Core Idea.* Traditional models convert text into numeric feature vectors (for example, token occurrence counts or TF–IDF scores). Learning then reduces to estimating a relatively simple decision function in this feature space. For instance, Naïve Bayes uses word occurrence probabilities under an independence assumption; Logistic Regression finds a linear decision boundary that best separates classes in the feature space; and SVM seeks the maximal margin hyperplane between classes. We outline these models and their properties below.

*Strengths.* Traditional models train quickly, scale well to large datasets, and are robust when labeled data are limited. They are easy to regularize (to avoid overfitting), straightforward to deploy in production, and yield coefficients or feature importance scores that support human interpretation and debugging. This interpretability is useful for auditing which words drive the predictions, an important feature for certain domains (e.g., legal or healthcare) where transparency is required.

*Limitations.* Because these models operate on mostly context-agnostic features, they cannot fully capture phenomena like word order, polysemy (multiple meanings of the same word), or long-range dependencies in text. As task complexity increases (for example, understanding nuanced sentiment or domain-specific language use), LLMs typically outperform traditional models. Nonetheless, the simplicity and clarity of statistical classifiers ensure they remain indispensable both as transparent baselines and as components in hybrid NLP systems.

**Overview of Approaches.** This section first describes the typical feature engineering pipeline for statistical text classification (tokenization, normalization, $n$-gram extraction, and weighting schemes like TF–IDF). It then presents the three classic classifiers (Naïve Bayes, Logistic Regression, and SVM), including their statistical formulation and key characteristics, and provides simple examples to illustrate how they work. Finally, we will highlight how these models will be used as

baselines in our comparative experiments.

## Feature Engineering for Text Classification

Traditional statistical models cannot directly process raw text; instead, the text must be transformed into structured numeric representations through feature engineering. This preprocessing step is critical because it defines the feature space in which the model will attempt to separate the classes.

**Tokenization and Normalization.** As a first step, raw text is broken into tokens (e.g., words or subword units). The text is typically normalized to reduce irrelevant differences: for example, all text may be lowercased to treat "Bank" and "bank" the same, and punctuation or excessive whitespace may be removed. In some cases, removing common stop words (like "the" or "and") can help reduce noise, and applying stemming or lemmatization can consolidate different forms of a word (e.g., *"running"* $\rightarrow$ *"run"*) into a single base form. The goal of these steps is to standardize the text and reduce sparsity in the representation.

**Bag-of-Words Representation.** One of the simplest and most commonly used representations is the bag-of-words (BoW) model, which converts a document into a vector of token occurrence counts:

$$\mathbf{x}_d = \big(\text{Count}(w_1, d),\ \text{Count}(w_2, d),\ \ldots,\ \text{Count}(w_V, d)\big)\,,$$

where $w_i$ denotes the $i$-th word in the vocabulary of size $V$, and $\text{Count}(w_i, d)$ is the number of times that word appears in document $d$. This yields a high-dimensional, sparse vector (most entries are zero for any given document, since each document contains only a small subset of the entire vocabulary).

**Term Frequency–Inverse Document Frequency (TF–IDF).** To mitigate the influence of extremely frequent words which may carry little semantic content (e.g., "the", "and"), BoW counts are often reweighted using TF–IDF. The TF–IDF score for word $w$ in document $d$ is:

$$\text{tfidf}(w, d) = \text{tf}(w, d) \cdot \log \frac{N}{\text{df}(w)},$$

where $\text{tf}(w, d)$ is the raw frequency of $w$ in $d$, $\text{df}(w)$ is the number of documents in the corpus that contain $w$, and $N$ is the total number of documents. Intuitively, TF–IDF increases with the word's frequency in the document but decreases with its frequency across the corpus, highlighting words that are important for that document but not common overall.

**Feature Selection and Regularization.** The dimensionality $V$ of text feature vectors can be extremely large (tens or hundreds of thousands of unique terms). To manage this, one can apply feature selection or regularization:

- Use statistical measures (e.g., chi-square or mutual information) to select a subset of tokens that are most indicative of the class labels, discarding rare or irrelevant terms.
- Apply regularization techniques (such as L1 or L2 penalties) during model training to discourage reliance on any single feature and to effectively ignore less useful features. For example, L1-regularized logistic regression (Lasso) can drive many feature weights to zero, performing implicit feature selection.

## 3.1 Term Frequency–Inverse Document Frequency (TF–IDF)

### 3.1.1 Definition

When dealing with text data, machines cannot directly understand words—they require numerical representations that describe how important each word is. **Term Frequency–Inverse Document**

**Frequency (TF–IDF)** is a numerical statistic that reflects how relevant a word is to a specific document within a larger collection of documents.

It combines two main concepts:

- **Term Frequency (TF):** measures how often a word appears in a single document.
- **Inverse Document Frequency (IDF):** measures how rare that word is across the entire corpus.

By combining both, TF–IDF highlights words that are frequent within a particular text but not common across all texts—helping models focus on the most meaningful terms for prediction.

### 3.1.2 Mathematical Formulation

The TF–IDF value for a term $t$ in a document $d$ from a collection of documents $D$ is calculated as:

$$\text{TF–IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Where:

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

and

$$\text{IDF}(t, D) = \log\left(\frac{N}{n_t}\right)$$

Here:

- $f_{t,d}$ is the number of times the term $t$ appears in document $d$.
- $N$ is the total number of documents in the corpus.
- $n_t$ is the number of documents that contain the term $t$.

The logarithm in the IDF reduces the influence of very common words that appear in almost every document, ensuring that such frequent words do not dominate the analysis.

### 3.1.3 Purpose and Intuition

The main goal of TF–IDF is to assign meaningful importance to words in a way that improves how models interpret text.

- Words that occur **often in one document** but **rarely in others** receive **high TF–IDF scores**, indicating that they are informative.
- Words that appear **frequently across all documents** receive **low scores**, as they do not help distinguish one document from another.

In this project, TF–IDF is used to transform Arabic tweets about Saudi banks into numerical features for the **Logistic Regression** model. These weighted values allow the model to identify patterns—such as recognizing that words like (excellent)ممتاز are associated with positive sentiment, while words like تأخير (delay) may indicate negative sentiment.

TF–IDF therefore acts as a bridge between language and mathematics, enabling statistical models to capture meaningful differences in text data.

### 3.1.4 Illustrative Example Application

**Step-by-step Calculation Example.** To demonstrate how the TF–IDF value is computed, consider the term `fast` in the document `doc2: ``Hail is growing fast and smart.''`

1. **Compute Term Frequency (TF):** The document `doc2` contains four unique words after cleaning: {`hail`, `growing`, `fast`, `smart`}. Each word appears once.

$$\text{TF}(\texttt{fast}, \texttt{doc2}) = \frac{f_{\texttt{fast,doc2}}}{\text{total terms in doc2}} = \frac{1}{4} = 0.25$$

2. **Compute Document Frequency (DF):** The term `fast` appears only in one document (`doc2`), so:

$$\mathrm{DF}(\texttt{fast}) = 1$$

3. **Compute Inverse Document Frequency (IDF):** The total number of documents in the corpus is $N = 3$. Using the natural logarithm:

$$\mathrm{IDF}(\texttt{fast}) = \ln\left(\frac{N}{\mathrm{DF}(\texttt{fast})}\right) = \ln\left(\frac{3}{1}\right) = 1.0986$$

4. **Compute TF–IDF:**

$$\mathrm{TF}-\mathrm{IDF}(\texttt{fast}, \texttt{doc2}) = \mathrm{TF} \times \mathrm{IDF} = 0.25 \times 1.0986 = 0.2747$$

Therefore, the word `fast` in `doc2` receives a TF–IDF score of **0.2747**, indicating it is an important term that uniquely characterizes that document.

**Final TF–IDF Results.** Repeating the same procedure for every term in all documents yields the following results.

Table 3: Manual TF–IDF calculation results for the three-document corpus.

| Doc | Term | term_count | doc_total | TF | IDF | TF–IDF |
|---|---|---|---|---|---|---|
| doc1 | data | 1 | 6 | 0.1667 | 1.0986 | 0.1831 |
| doc1 | innovation | 1 | 6 | 0.1667 | 1.0986 | 0.1831 |
| doc1 | king | 1 | 6 | 0.1667 | 1.0986 | 0.1831 |
| doc1 | leads | 1 | 6 | 0.1667 | 1.0986 | 0.1831 |
| doc1 | saud | 1 | 6 | 0.1667 | 1.0986 | 0.1831 |
| doc1 | statistics | 1 | 6 | 0.1667 | 1.0986 | 0.1831 |
| doc2 | fast | 1 | 4 | 0.2500 | 1.0986 | 0.2747 |
| doc2 | growing | 1 | 4 | 0.2500 | 1.0986 | 0.2747 |
| doc2 | smart | 1 | 4 | 0.2500 | 1.0986 | 0.2747 |
| doc2 | hail | 1 | 4 | 0.2500 | 0.4055 | 0.1014 |
| doc3 | arabia | 1 | 7 | 0.1429 | 1.0986 | 0.1570 |
| doc3 | global | 1 | 7 | 0.1429 | 1.0986 | 0.1570 |
| doc3 | growth | 1 | 7 | 0.1429 | 1.0986 | 0.1570 |
| doc3 | inspires | 1 | 7 | 0.1429 | 1.0986 | 0.1570 |
| doc3 | ksu | 1 | 7 | 0.1429 | 1.0986 | 0.1570 |
| doc3 | saudi | 1 | 7 | 0.1429 | 1.0986 | 0.1570 |
| doc3 | hail | 1 | 7 | 0.1429 | 0.4055 | 0.0579 |

Table 4: Wide TF–IDF matrix (Part 1) for the three-document corpus.

| doc_id | data | innovation | king | leads | saud | statistics | fast | growing |
|---|---|---|---|---|---|---|---|---|
| doc1 | 0.183 | 0.183 | 0.183 | 0.183 | 0.183 | 0.183 | 0.000 | 0.000 |
| doc2 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.275 | 0.275 |
| doc3 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

Table 5: Wide TF–IDF matrix (Part 2) for the three-document corpus.

| doc_id | smart | hail | arabia | global | growth | inspires | ksu | saudi |
|---|---|---|---|---|---|---|---|---|
| doc1 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| doc2 | 0.275 | 0.101 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| doc3 | 0.000 | 0.0579 | 0.157 | 0.157 | 0.157 | 0.157 | 0.157 | 0.157 |

**Interpretation.** Words such as `fast`, `growing`, and `smart` have the highest TF–IDF scores (around 0.27), showing that they are distinctive in `doc2`. In contrast, `hail` appears in multiple documents and therefore receives a lower TF–IDF weight, indicating it carries less discriminative power. This weighting mechanism allows the Logistic Regression model to focus on the most informative words when classifying or analyzing text.

## 3.2 Logistic Regression

Logistic Regression (LR) is a discriminative model that directly estimates class probabilities using a linear function of the input features. Unlike Naïve Bayes, LR does not assume feature independence; it can learn weights that account for correlations between features implicitly.

**Multinomial Logistic Regression Equation**

In a multi-class classification problem with $K$ classes, the logistic regression model computes a separate score $z_k$ for each class $k \in \{1, 2, \ldots, K\}$ using:

$$z_k = \mathbf{w}_k^\top \mathbf{x} + b_k$$

Where:

- $\mathbf{x} \in \mathbb{R}^n$ is the feature vector representing the input (e.g., a movie review),
- $\mathbf{w}_k \in \mathbb{R}^n$ is the weight vector for class $k$,
- $b_k \in \mathbb{R}$ is the bias term for class $k$,
- $z_k$ is the unnormalized score for class $k$.

These scores are then passed through the **softmax function** to obtain probabilities:

$$P(y = k \mid \mathbf{x}) = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

This ensures all class probabilities are between 0 and 1 and sum to 1. The class with the highest probability is chosen as the model's prediction.

**Component Explanations**

**1. Feature Vector (x).** The feature vector **x** encodes information about the input instance. In natural language processing, features are often extracted using:

- Bag-of-Words or TF–IDF representations,
- Word counts (e.g., number of positive/negative words),
- Binary indicators (e.g., presence of negation or punctuation).

**2. Weights ($\mathbf{w}_k$).** Each class $k$ has an associated weight vector $\mathbf{w}_k$. Each component $w_{k,i}$ indicates how much the $i$-th feature contributes to the prediction of class $k$:

- A large positive $w_{k,i}$ increases the probability of class $k$,
- A large negative $w_{k,i}$ decreases the probability of class $k$,
- A near-zero $w_{k,i}$ means the feature has little influence on class $k$.

For example, if class "Positive" has high weights for the features "excellent" and "amazing," these terms increase the likelihood that an input is classified as positive.

**3. Bias ($b_k$).** The bias term $b_k$ allows the model to account for class imbalances or inherent tendencies. It shifts the score $z_k$ up or down regardless of the input features. For instance, if most reviews are neutral, the model may learn a higher $b_{\text{neutral}}$ to reflect this.

**4. Linear Score ($z_k$).** Each class score is computed as:

$$z_k = \sum_{i=1}^{n} w_{k,i} \cdot x_i + b_k = \mathbf{w}_k^\top \mathbf{x} + b_k$$

These are raw confidence scores before normalization. The higher $z_k$, the more confident the model is in class $k$.

**5. Softmax Activation.** The softmax function converts raw scores into a valid probability distribution:

$$P(y = k \mid \mathbf{x}) = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$$

This guarantees that:

- $P(y = k \mid \mathbf{x}) \in [0, 1]$ for each class $k$,
- $\sum_{k=1}^{K} P(y = k \mid \mathbf{x}) = 1$.

The predicted label is the class with the highest probability.

**Advantages.**

Logistic Regression makes no explicit assumptions about feature distribution or independence. It often performs better than Naïve Bayes when features are correlated or when there is sufficient training data to estimate a reliable weight for each feature. The learned weights **w** are straightforward to interpret: a positive $w_j$ means the presence of the $j$-th feature increases the log-odds of the positive class, while a negative $w_j$ means it decreases the log-odds. This interpretability is useful for understanding which words are most indicative of each class. LR also extends gracefully to multiple classes.

**Limitations.**

Being a linear model, LR can only carve linear decision boundaries in the feature space. If the true decision boundary is highly nonlinear (in the original text feature space), LR may struggle unless

features are enriched (for example, using higher-order $n$-grams or embedding features). Another limitation is that LR's performance relies heavily on the chosen features; if important contextual information is not encoded in the features, LR cannot infer it. However, with a rich feature set (like unigrams + bigrams with TF–IDF weighting) and regularization, logistic regression can be a very competitive text classifier.

## 3.3 Illustrative Example: Using Logistic Regression for Movie Review Sentiment Classification

**Task Overview.**

In this section, we demonstrate how a logistic regression model performs sentiment classification on short movie reviews. The task is to determine whether a review expresses a **Positive** or **Negative** sentiment based on the textual content. We use a simple, interpretable pipeline where each sentence is transformed into TF–IDF features, multiplied by learned model weights $\mathbf{w}$, combined with the intercept $b$, and passed through the sigmoid function $\sigma(z)$ to produce a probability of belonging to the Positive class.

- **Input:** "The movie was fantastic and thrilling." $\Rightarrow$ **Positive**
- **Input:** "The plot was dull and uninteresting." $\Rightarrow$ **Negative**

This example illustrates the complete transformation sequence:

$$\text{TF–IDF features } (\mathbf{x}) \;\rightarrow\; \text{linear score } (z = \mathbf{w}^\top \mathbf{x} + b) \;\rightarrow\; \text{sigmoid } (\sigma(z)) \;\rightarrow\; \text{final prediction.}$$

**Training Data.**

We trained the model on the following labeled mini–corpus (14 examples):

| | text | label |
|---|---|---|
| 1 | The movie was fantastic and thrilling | Positive |
| 2 | The plot was dull and uninteresting | Negative |
| 3 | Absolutely wonderful film with great acting | Positive |
| 4 | Terrible and boring storyline | Negative |
| 5 | I loved the cinematography and the music | Positive |

```
6            I hated the pacing and the acting Negative
7                 What a fantastic experience Positive
8                       This was a dull movie Negative
9              A wonderful and enjoyable film Positive
10             A disappointing and slow movie Negative
11   An excellent and heartwarming experience Positive
12                    A bad and terrible plot Negative
13          Amazing acting and great direction Positive
14           Awful script and weak performance Negative
```

**Trained Parameters.**

After fitting a TF–IDF + L2–regularized logistic regression, we obtained an intercept $b = 0.000000$ and the weight vector **w** (shown here for the terms that actually appear in the two evaluation sentences):

Table 6: Model weights **w** (subset for terms used in the two sentences)

| term | weight |
|---|---|
| thrilling | 0.4756 |
| uninteresting | -0.3811 |
| dull | -0.4943 |
| fantastic | 0.4507 |
| plot | -0.4820 |
| movie | -0.2280 |
| was | -0.2036 |
| the | 0.0076 |
| and | -0.4385 |

**Step 1: Feature Extraction (TF–IDF)**

Using the trained vocabulary and IDF, we obtain sparse TF–IDF feature vectors **x** for each sentence (non–zero entries only).

Table 7: Non-zero TF–IDF entries for Sentence 1

| term | x |
|---|---|
| movie | 0.3218 |
| fantastic | 0.3090 |
| the | 0.2732 |
| thrilling | 0.2732 |
| and | 0.2069 |
| was | 0.1951 |

Table 8: Non-zero TF–IDF entries for Sentence 2

| term | x |
|---|---|
| uninteresting | 0.4513 |
| dull | 0.3466 |
| plot | 0.3466 |
| was | 0.2891 |
| the | 0.2507 |
| and | 0.1368 |

**Step 2–3: Linear Score**

For each sentence we compute the linear score

$$z = \mathbf{w}^\top \mathbf{x} + b.$$

Using the entries above and the learned $\mathbf{w}$ and $b$ we obtain:

**Sentence 1 ("The movie was fantastic and thrilling.").**

$$z_1 = \underbrace{\begin{bmatrix} 0.4756 & -0.3811 & -0.4943 & 0.4507 & -0.4820 & -0.2280 & -0.2036 & 0.0076 & -0.4385 \end{bmatrix}}_{\mathbf{w}^\top} \times \underbrace{\begin{bmatrix} 0.2732 \\ 0 \\ 0 \\ 0.3090 \\ 0 \\ 0.3218 \\ 0.1951 \\ 0.2732 \\ 0.2069 \end{bmatrix}}_{\mathbf{x}_1} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{b} = 0.223163$$

**Sentence 2 ("The plot was dull and uninteresting.").**

$$z_2 = \underbrace{\begin{bmatrix} 0.4756 & -0.3811 & -0.4943 & 0.4507 & -0.4820 & -0.2280 & -0.2036 & 0.0076 & -0.4385 \end{bmatrix}}_{\mathbf{w}^\top} \times \underbrace{\begin{bmatrix} 0 \\ 0.4513 \\ 0.3466 \\ 0 \\ 0.3466 \\ 0 \\ 0.2891 \\ 0.2507 \\ 0.1368 \end{bmatrix}}_{\mathbf{x}_2} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{b} = 0.356137$$

**Step 4: Sigmoid and Probability**

The logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$ maps the score into a valid probability:

$$P(\text{Positive} \mid \mathbf{x}_1) = \sigma(z_1) = 0.555560, \qquad P(\text{Positive} \mid \mathbf{x}_2) = \sigma(z_2) = 0.356137.$$

**Step 5: Final Prediction**

We predict **Positive** when $P \geq 0.5$ and **Negative** otherwise. Thus, S1 is classified as **Positive** and S2 as **Negative**.

**Decision Boundary on the Sigmoid Axis**

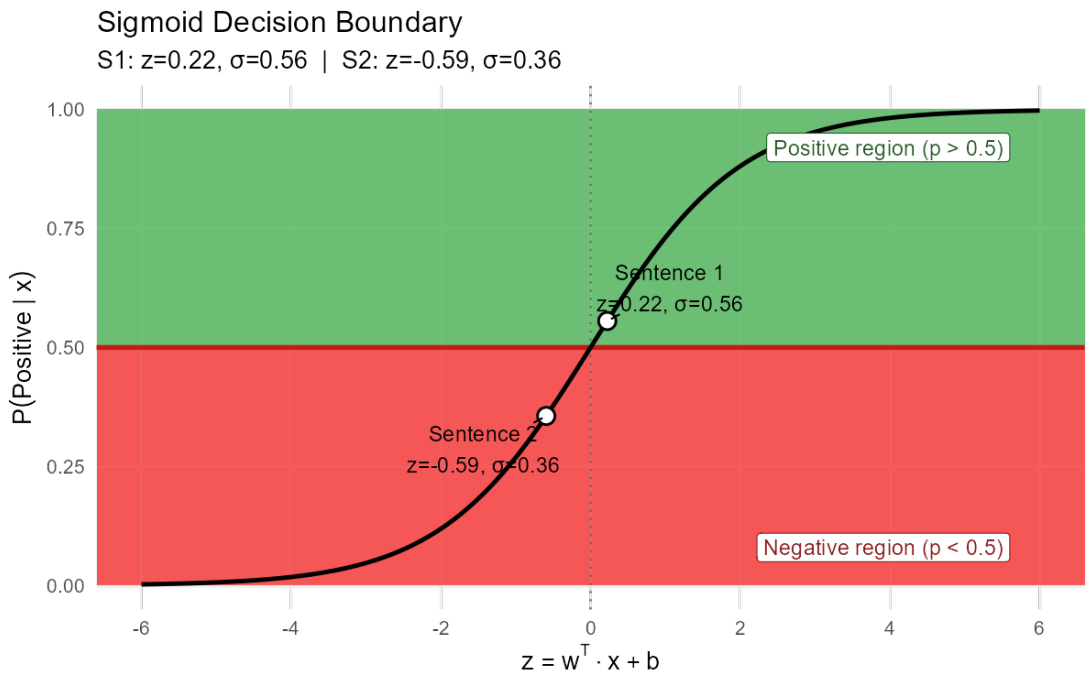



Figure 1: Sigmoid curve with the decision threshold at $p = 0.5$; the two sentences are shown at $(z_1, \sigma(z_1))$ and $(z_2, \sigma(z_2))$.

# 4 Evaluation Metrics

This section specifies the evaluation protocol and the metrics used to compare the fine-tuned LLM (**BERT**) against a traditional baseline (**TF–IDF + Logistic Regression**) on the *Saudi Bank Sentiment* dataset with classes $\mathcal{C} = \{\text{Neg}, \text{Neu}, \text{Pos}\}$. Our design aims to (i) fairly evaluate performance under class imbalance, (ii) quantify statistical uncertainty, (iii) assess probability quality and calibration, and (iv) ensure reproducibility.

## 4.1 Data Splits, Stratification, and Reproducibility

We use a fixed **stratified** split into **train/validation/test** sets (Sec. 5.4), preserving class proportions. The validation set is used *only* for model selection and calibration; all final results are reported on the held-out test set.

To reduce variance due to randomness, we train each model with $S$ independent random seeds (default $S = 5$ unless otherwise stated). For each seed, we store the test-set predicted class probabilities $\mathbf{p}_i = (p_{i,\text{Neg}}, p_{i,\text{Neu}}, p_{i,\text{Pos}})$ and hard predictions $\hat{y}_i = \arg\max_{c \in \mathcal{C}} p_{i,c}$. We then report the mean across seeds, and 95% confidence intervals (CIs) computed by nonparametric bootstrap on test predictions (10,000 resamples).

All preprocessing is fit on *train only*: TF–IDF vocabulary/statistics, tokenization settings, and any calibration parameters. No threshold tuning is performed on the test set.

## 4.2 Confusion Matrix and Per-Class Counts

Let $TP_c$, $FP_c$, and $FN_c$ denote one-vs-rest counts for class $c$. We report the confusion matrix on the test set to make error trade-offs explicit, along with class supports $N_c$.

## 4.3 Hard-Label Metrics (Multiclass)

For each class $c \in \mathcal{C}$,

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}, \quad \text{Recall}_c = \frac{TP_c}{TP_c + FN_c}, \quad \text{F1}_c = \frac{2\,\text{Precision}_c\,\text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}.$$

We aggregate these per-class scores in two main ways:

- **Macro-average** (treats classes equally; robust under imbalance):

  $$\text{Precision}_{\text{macro}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \text{Precision}_c, \quad \text{Recall}_{\text{macro}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \text{Recall}_c, \quad \text{F1}_{\text{macro}} = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \text{F1}_c.$$

  In our study, $\mathbf{F1_{macro}}$ is the primary metric.

- **Support-weighted average** (reflects the label distribution):

  $$\text{Precision}_{\text{weighted}} = \sum_{c\in\mathcal{C}} \frac{N_c}{N}\text{Precision}_c, \quad \text{Recall}_{\text{weighted}} = \sum_{c\in\mathcal{C}} \frac{N_c}{N}\text{Recall}_c, \quad \text{F1}_{\text{weighted}} = \sum_{c\in\mathcal{C}} \frac{N_c}{N}\text{F1}_c.$$

We additionally report **Accuracy**:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{\hat{y}_i = y_i\}.$$

Accuracy is reported for completeness but is not used alone to select winners due to class imbalance.

## 4.4 Probabilistic Quality: Log-Loss and Brier Score

Because both models output probabilities, we evaluate probability quality (not only hard labels). Let $y_i \in \mathcal{C}$ be the true label for example $i$ and $p_{i,y_i}$ the predicted probability assigned to the true class.

- **Log-loss (cross-entropy)**:

  $$\text{log\_loss} = -\frac{1}{N}\sum_{i=1}^{N} \log p_{i,y_i}.$$

  Lower is better; the metric strongly penalizes confident wrong predictions.

- **Multiclass Brier score**: using one-hot targets $y_{i,c} \in \{0, 1\}$,

$$\text{brier\_score} = \frac{1}{N} \sum_{i=1}^{N} \sum_{c \in \mathcal{C}} (p_{i,c} - y_{i,c})^2.$$

  Lower is better; this measures the mean squared error of probabilities.

## 4.5 Calibration: Reliability Diagram and ECE (15 bins)

**Reliability diagram.** A reliability (calibration) diagram compares predicted *confidence* to empirical *accuracy*. For multiclass, we use the maximum predicted probability as confidence:

$$\hat{p}_i = \max_{c \in \mathcal{C}} p_{i,c}, \qquad \hat{y}_i = \arg\max_{c \in \mathcal{C}} p_{i,c}.$$

We partition $[0, 1]$ into $M$ bins (we use $M = 15$, hence `ece_15bins`). For bin $m$, let $B_m$ be the set of indices whose confidence falls in that bin. Define:

$$\text{conf}(m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i, \qquad \text{acc}(m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}\{\hat{y}_i = y_i\}.$$

The reliability diagram plots points $(\text{conf}(m), \text{acc}(m))$; perfect calibration lies on the diagonal acc $=$ conf.

**Expected Calibration Error (ECE).** ECE summarizes the average gap between accuracy and confidence across bins:

$$\text{ece\_15bins} = \sum_{m=1}^{15} \frac{|B_m|}{N} \left|\text{acc}(m) - \text{conf}(m)\right|.$$

Lower is better.

**Calibration methods (validation-only).** Calibration is fit *on the validation set*:

- **BERT:** temperature scaling on logits.

- **Logistic Regression:** optional Platt scaling (one-vs-rest) or isotonic regression (when appropriate).

We report Brier and ECE on the test set (before/after calibration) to quantify calibration improvements.

## 4.6 Curves: ROC and PR (One-vs-Rest)

**ROC curve (one-vs-rest).** For each class $c$, we treat $c$ as positive and $\mathcal{C} \setminus \{c\}$ as negative, and vary a threshold $\tau \in [0, 1]$ on $p_{i,c}$. This yields:

$$\text{TPR}_c(\tau) = \frac{TP_c(\tau)}{TP_c(\tau) + FN_c(\tau)}, \qquad \text{FPR}_c(\tau) = \frac{FP_c(\tau)}{FP_c(\tau) + TN_c(\tau)}.$$

Plotting $\text{TPR}_c(\tau)$ vs. $\text{FPR}_c(\tau)$ gives the ROC curve; the area under this curve is $\text{AUC}_c$. We report macro-averaged AUC when included:

$$\text{AUC}_{\text{macro}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{AUC}_c.$$

(ROC can look optimistic under severe imbalance; therefore we also emphasize PR curves.)

**Precision–Recall (PR) curve and AUPRC.** PR curves are often more informative under class imbalance. We compute one-vs-rest PR curves for each class and report macro-AUPRC:

$$\text{AUPRC}_{\text{macro}} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{AUPRC}_c,$$

and, when the application prioritizes a target class (e.g., Positive), we additionally report $\text{AUPRC}_{\text{Pos}}$.

## 4.7 Model Selection and Early Stopping

- **BERT:** we select the checkpoint with the best validation $\text{F1}_{\text{macro}}$ (tie-break by lowest log-loss). Early stopping is used with patience $P$ epochs; the best checkpoint is retained.

- **Logistic Regression:** we tune regularization and TF–IDF settings (e.g., $C$, penalty, n-gram range, min_df/max_df) using the validation set, optimizing $\mathrm{F1}_{\text{macro}}$ (tie-break by log-loss).

After selection, the model is frozen and evaluated on the test set.

## 4.8 Uncertainty: Bootstrap Confidence Intervals

For each seed we obtain test predictions; we compute metric estimates by resampling the test set with replacement (10,000 bootstrap samples). For a metric such as macro-F1, we report:

$$\text{f1\_macro\_ci\_lo},\ \text{f1\_macro\_ci\_hi}$$

as the 2.5th and 97.5th percentiles of the bootstrap distribution. We also compute bootstrap CIs for differences (e.g., $\Delta\mathrm{F1}_{\text{macro}}$) and consider a difference significant if its 95% CI excludes 0.

## 4.9 Significance Testing: McNemar's Test (Paired Errors)

To test whether two classifiers differ significantly on the *same* test instances, we use **McNemar's test** on hard-label predictions. Let:

- $n_{01}$: instances where model A is wrong and model B is correct,
- $n_{10}$: instances where model A is correct and model B is wrong.

With continuity correction, the test statistic is:

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}},$$

which is compared to a $\chi^2$ distribution with 1 degree of freedom. (When $n_{01} + n_{10}$ is small, we additionally consider the exact binomial version.)

If multiple pairwise comparisons are performed, we control the family-wise error rate using Holm–Bonferroni.

# 5 First Case Study: Saudi Bank Sentiment Analysis

## 5.1 Data Acquisition

The dataset used in this case study is the **Saudi Bank Sentiment Analysis Dataset**, publicly available at:

https://github.com/iwan-rg/Saudi-Bank-Sentiment-Dataset

The data were collected from Twitter using the Tweepy Python library. Tweets were retrieved by monitoring mentions of four major Saudi banks: AlRajhi, Alinma, Saudi National Bank (SNB), and Saudi Investment Bank (SAIB). The collection period covered the entire month of September 2021, a time when the COVID-19 pandemic forced banks to provide most services online, making social media the main communication channel between customers and financial institutions.

In total, the dataset contains 12,048 tweets, manually labeled into three sentiment categories... This dataset was originally introduced by Alqahtani et al. [4].

- 8,669 Negative (71.9%),
- 2,143 Positive (17.8%),
- 1,236 Neutral (10.3%).

This strong class imbalance highlights the importance of evaluation metrics that go beyond overall accuracy, such as macro-F1, which treats all classes equally.

| Tweet (Text) | Label |
|---|---|
| بنك من فيك الوكيل ونعم الله حسبنا الله مستقبلنا دمر بنك من يقلعك الله | NEG |
| وهمي شغل ولا بنك انتم فعلا هل عجزت علي يرد قبلكم من شخص لاي اوصل بحاول يومين لي صار | NEG |
| نهاءيا عنه نظر صرفت وبصراحه الصراحه منه يشتكون الزملاء من كثير صادق | NEG |
| فايده ومافي الفروع وادور فاهمين مو العملاء خدمه وموظفين ذهبي وحسابي مدي بطاقه ناخذ ماقدرنا عندكم كبير خلل في الافضل مو للاسواء تغيرتو للاسف فايده بدون عليكم وادق اطبع علشان | NEG |
| بعد عروض من يدور وش اقساط نصها رواتبهم | NEG |
| الرد يتم وسوف بالصبر التحلي يرجي بالبنك الزميل عزيزي دقيقه وانت دقيقه علي اتضجر وانا كذا اما للتمديد قابله سنه وانا سنه بعد عليك | NEG |
| استفسار عندي يرد محد عليكم نتصل قاعدين لوسمحتوا | NEG |
| خبره واقع من افضل الثانيه البنوك مثلي لاتورطون | NEG |
| ماتغير نفسه القرض مبلغ ولازال الفاءته سنوات الست طيله اسدد ولازلت منهم انا | NEG |
| حسابك في المبلغ ورجعنا مشكله عندنا كان يقولون وقسط عندهم من ماخذ انا الخير يبيلك ربي احسنت وحط شيل معهم وانا وليومك صحيح غير الكلام وهذا | NEG |

Table 9: First 10 Labeled Tweets from the Saudi Bank Sentiment Dataset

## 5.2 Methodology

To assess the relative performance of traditional statistical models and a Large Language Model (LLM), the following methodology was adopted:

1. **Preprocessing:** Tweets were cleaned by removing URLs, mentions, hashtags, and diacritics. Arabic letters were normalized, and English text was lowercased. Tokenization was applied at the word or subword level, depending on the model.

2. **Feature Representation:**

- For traditional models: term frequency–inverse document frequency (TF–IDF) features with unigrams and bigrams, capped at 20,000 features.
- For the LLM: subword embeddings from `bert-base-multilingual-cased`, which encodes tokens into high-dimensional contextual vectors.

3. **Models Evaluated:**
   - **Logistic Regression (LR):** L2-regularized, trained on TF–IDF.
   - **LLM:** Fine-tuned multilingual BERT (mBERT) with cross-entropy loss, learning rate $2 \times 10^{-5}$, batch size 16, for 3 epochs with early stopping on validation macro-F1.
4. **Evaluation Protocol:**
   - Data split: 70% training, 10% validation, 20% test (stratified by class).
   - Metrics: Accuracy, Macro-F1, Log-loss, Brier score, Expected Calibration Error (ECE).
   - Statistical tests: bootstrap confidence intervals (95% CIs, $B = 1000$ resamples) for Macro-F1, and McNemar's test for accuracy comparisons.

*REMARK* 5.1. This design allows for a rigorous statistical comparison between classical approaches and modern LLMs. The imbalanced nature of the dataset emphasizes the role of robust evaluation, especially for minority classes (Positive and Neutral).

## 5.3 Data Preparation & Cleaning

Before modeling, the raw corpus was standardized to reduce noise and improve consistency:

1. **Noise removal:** Strip URLs, mentions (`@username`), hashtags, emojis, and extra spaces.

2. **Normalization:** Unify Arabic letter variants (e.g., أ/إ/آ → ا,( remove diacritics, and reduce elongations.

3. **Tokenization:**

   - *Traditional models:* word-level unigrams/bigrams.
   - *LLM:* subword units via the mBERT tokenizer.

4. **Representation:**

   - *Traditional:* TF–IDF vectors (max 20,000 features).
   - *LLM:* contextual embeddings from `bert-base-multilingual-cased`.

5. **Imbalance handling:** Use stratified splitting and macro-$F_1$ as a primary metric.

## 5.4 Data Splitting Methodology

We apply **stratified** sampling to preserve class proportions across splits.

### Cleaned Dataset Snapshot

After cleaning, the labeled corpus contains approximately 6,000 tweets (illustrative composition in Table 10).

Table 10: Cleaned dataset composition (illustrative).

| Label | Meaning | Example | Count (≈) | Percentage |
|---|---|---|---|---|
| NEG | Negative tweets | معلق شوي كل فاشل تطبيقكم | ∼8,669 | 72% |
| NEU | Neutral tweets | عادية الخدمة | ∼2,143 | 19% |
| POS | Positive tweets | ممتاز السعودي البنك | ∼1,236 | 10% |
| **Total** | | | ∼12,048 | 100% |

**Split Rationale**

We partition data into Train/Validation/Test as in Table 11.

Table 11: Purpose and proportions of data subsets.

| Subset | Proportion | Purpose |
|---|---|---|
| Training | 70% | Fit model parameters (weights/embeddings/attention). |
| Validation | 10% | Hyperparameter tuning and early stopping. |
| Test | 20% | Final generalization estimate on unseen data. |

**Stratified Allocation**

Stratification preserves sentiment ratios within each subset (Table 12).

Table 12: stratified distribution across splits.

| Class | Total | 70% Train | 10% Val | 20% Test |
|---|---|---|---|---|
| NEG | 8,669 | 6,068 | 867 | 1,734 |
| NEU | 2,143 | 1,500 | 214 | 429 |
| POS | 1,236 | 865 | 124 | 247 |
| **Total** | 12,048 | 8433 | 1205 | 2410 |

*REMARK* 5.2. Stratified sampling mitigates sampling bias toward the majority class and yields more reliable training dynamics and evaluation.

## 5.5 Training Results

Table 13 summarizes test performance for the fine-tuned BERT model versus the TF–IDF + Logistic Regression (LR) baseline on the *same* held-out split. BERT achieves higher Accuracy and Macro-$F_1$, indicating better balance across classes under strong label skew (NEG dominates the test set). Both models struggle most on the *Neutral* class: Neutral recall is the weakest for both systems, showing that borderline cases are often absorbed into the majority Negative class (see confusion matrices in Fig. 2 and Fig. 6).

Table 13: Test-set results (shared split) for BERT and TF–IDF Logistic Regression.

| Model | Accuracy | Macro-F1 | Log-loss | Macro-AUC (OVR) |
|---|---|---|---|---|
| BERT | 0.846 | 0.678 | 0.466 | 0.894 |
| LR | 0.824 | 0.624 | 0.476 | 0.868 |

**Per-Class Metrics (from the confusion matrices).**

- **Negative (NEG):** BERT (Prec. 0.872, Rec. 0.950, F1 0.909); LR (Prec. 0.837, Rec. 0.957, F1 0.893).
- **Positive (POS):** BERT (Prec. 0.863, Rec. 0.776, F1 0.817); LR (Prec. 0.862, Rec. 0.671, F1 0.755).
- **Neutral (NEU):** BERT (Prec. 0.437, Rec. 0.239, F1 0.309); LR (Prec. 0.409, Rec. 0.154, F1 0.224).

**ROC Curves (Discrimination).** Both models exhibit strong one-vs-rest discrimination for Positive tweets, while Neutral remains the hardest class. BERT achieves a higher macro-AUC (0.894) than LR (0.868), consistent with the ROC plots in Fig. 3 and Fig. 7.

**Reliability Diagram (Calibration).** BERT's reliability curve (Fig. 4) tracks the diagonal reasonably well, with mild under-confidence in the mid-confidence region. This supports using BERT probabilities for downstream decision-making after validation-based calibration.

**Paired significance vs. majority baseline (NEG).** McNemar's test confirms both models significantly outperform the majority baseline that always predicts NEG: BERT ($b$=392, $c$=87, $\chi^2$=192.94, $p$=7.27$\times 10^{-44}$; Fig. 5) and LR ($b$=326, $c$=75, $\chi^2$=155.86, $p$=9.08 $\times 10^{-36}$; Fig. 8).

**Click-to-jump Results (Shared Test Split).**

- **BERT:** Confusion Matrix | ROC Curves | Reliability Diagram | McNemar vs Majority
- **Logistic Regression:** Confusion Matrix | ROC Curves | McNemar vs Majority

### 5.5.1 BERT (Fine-tuned) Results

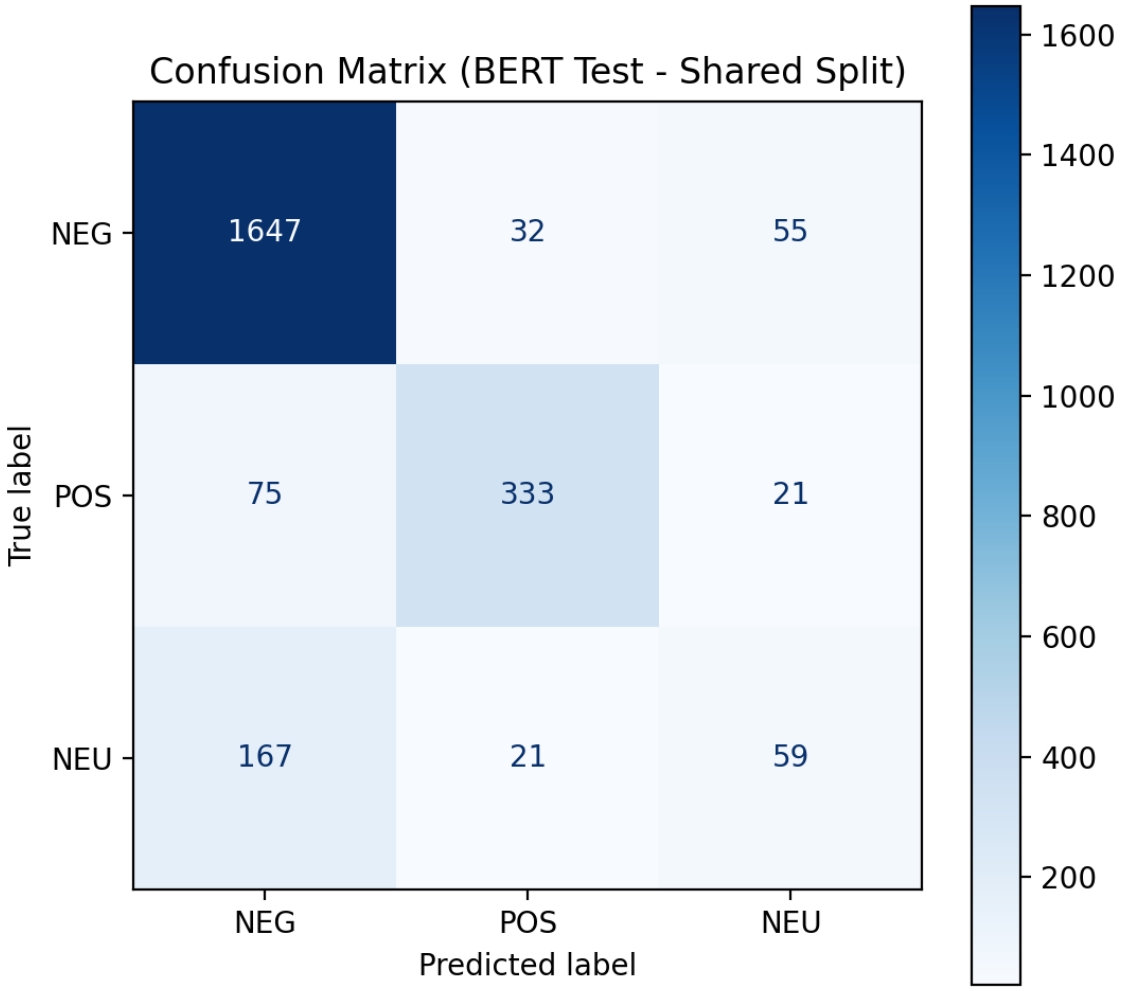



Figure 2: **BERT Confusion Matrix (Shared Test).** Rows are true labels and columns are predicted labels. Most errors come from NEU and POS being predicted as NEG, reflecting class imbalance where borderline tweets are absorbed into the majority class.

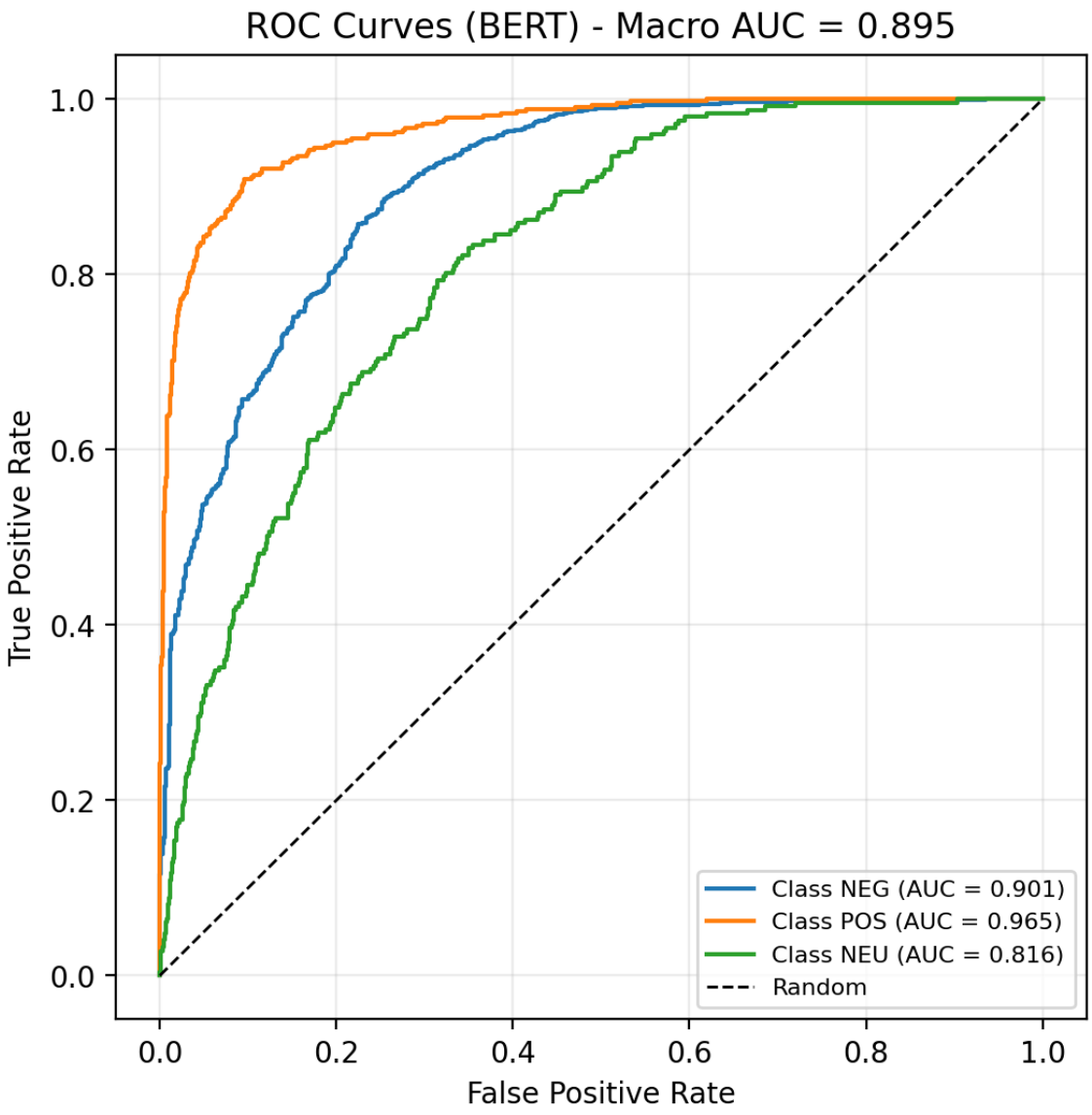



Figure 3: **BERT ROC Curves (One-vs-Rest).** Each curve treats one class as positive and the other two as negative, varying the decision threshold. Curves closer to the top-left indicate better separability; the reported macro-AUC summarizes overall ranking quality across the three classes.

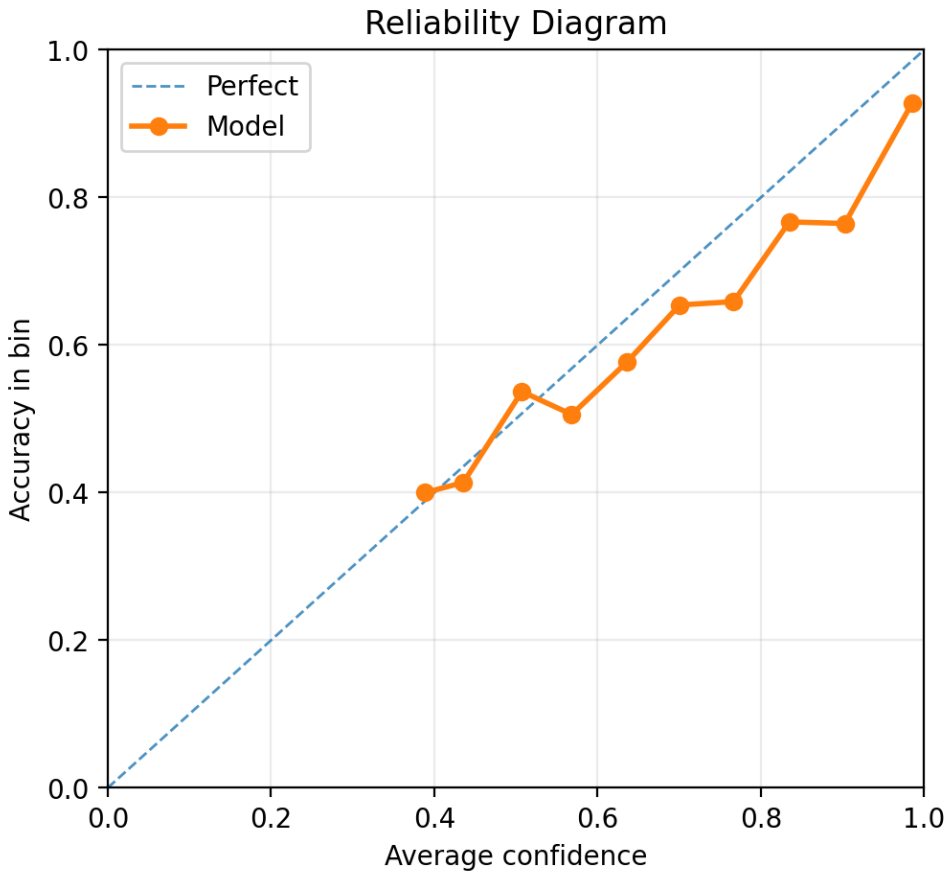



Figure 4: **BERT Reliability Diagram (Calibration).** The x-axis is the model's average confidence (max predicted probability) within each bin, and the y-axis is the empirical accuracy in that bin. Perfect calibration follows the diagonal; deviations indicate over- or under-confidence.

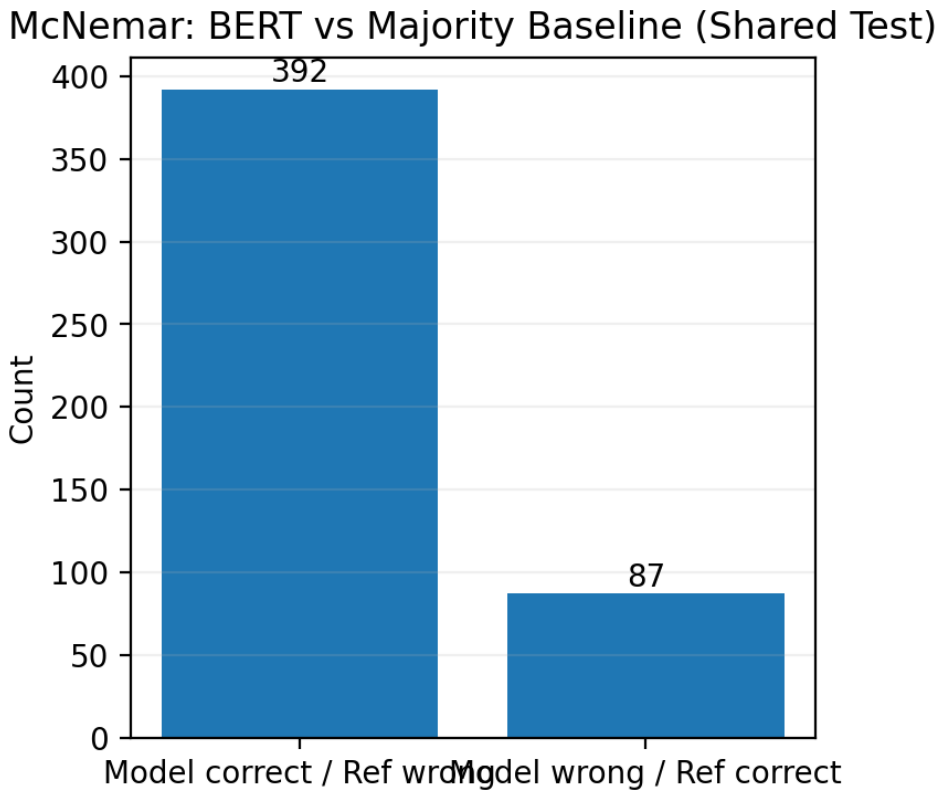



Figure 5: **McNemar Comparison: BERT vs Majority Baseline.** Bars show the paired disagreement counts on the same test items: *Model correct / Baseline wrong* ($n_{01}$) versus *Model wrong / Baseline correct* ($n_{10}$). McNemar's test uses these counts to determine whether BERT significantly outperforms the majority classifier.

### 5.5.2 Logistic Regression (TF–IDF) Results

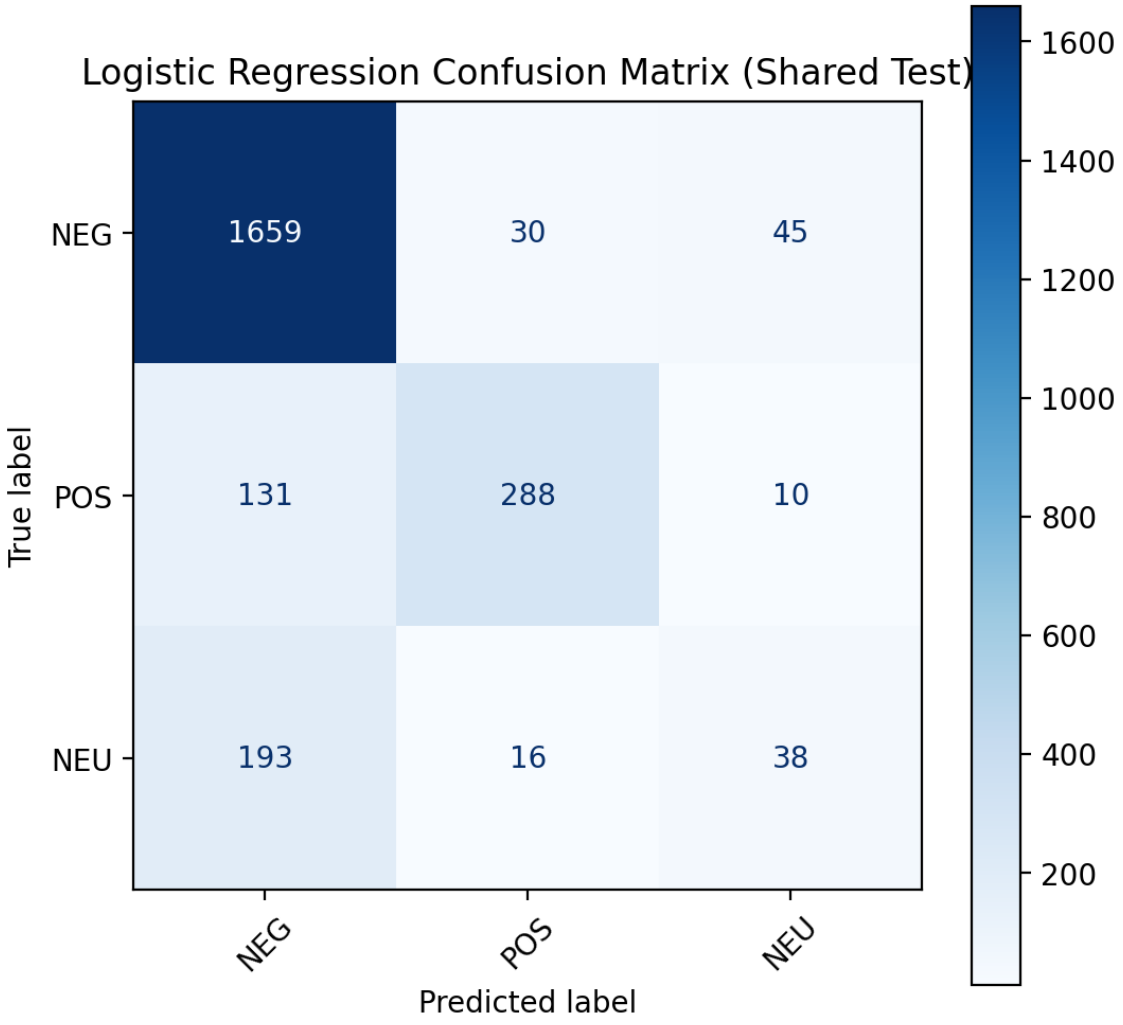



Figure 6: **Logistic Regression Confusion Matrix (Shared Test).** Rows are true labels and columns are predicted labels. Similar to BERT, the baseline performs strongly on NEG but struggles most on NEU, which is frequently misclassified as NEG.

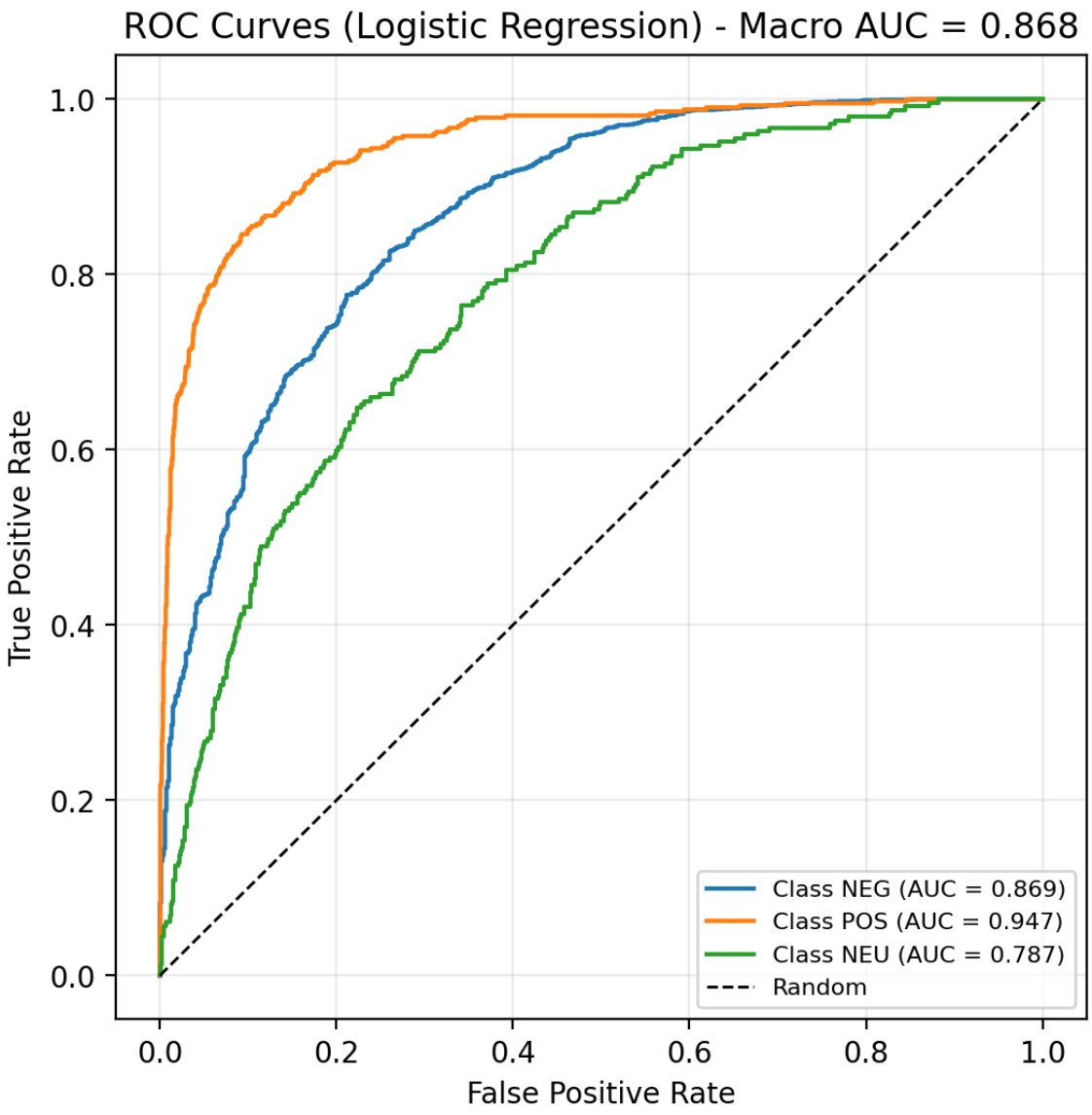



Figure 7: **Logistic Regression ROC Curves (One-vs-Rest).** ROC curves are computed by thresholding the one-vs-rest probability for each class. The macro-AUC summarizes overall discrimination ability; lower AUC for NEU indicates it is the hardest class to separate.

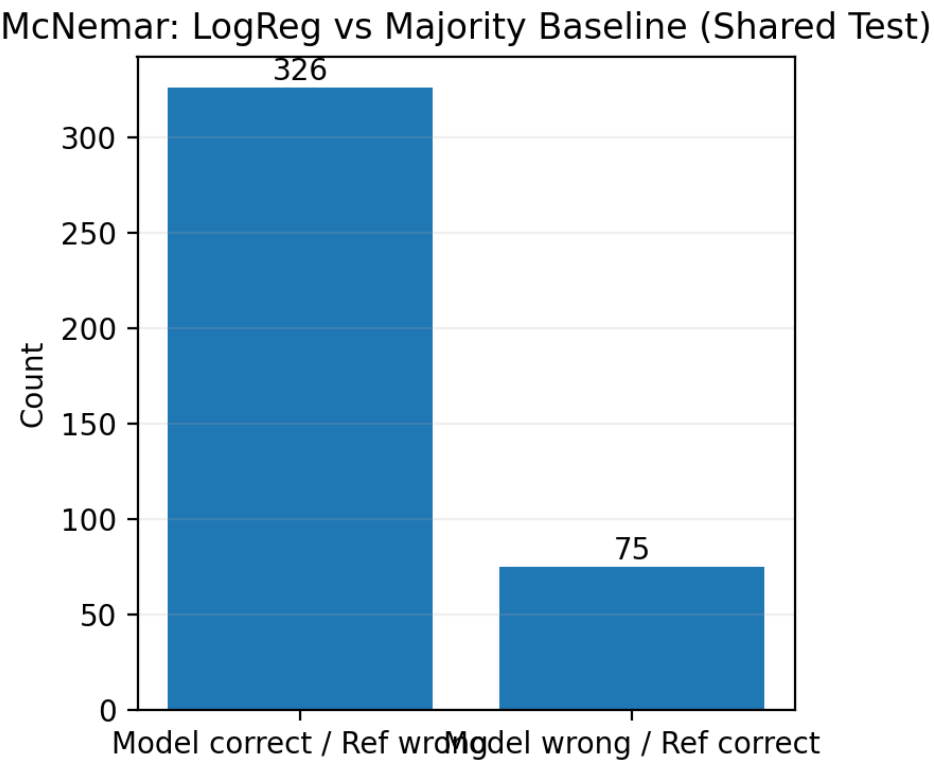



Figure 8: **McNemar Comparison: Logistic Regression vs Majority Baseline.** Bars show the paired disagreement counts ($n_{01}$ vs $n_{10}$) on the shared test set. A larger $n_{01}$ than $n_{10}$ indicates the classifier improves over the majority baseline; McNemar's test evaluates whether the difference is statistically significant.

## 5.6 Example Output

Consider the tweet:

للأسف تم احتسابها لي بطريقة خاطئه كيف يتم الاعتراض على الاحتساب

Model predictions:

- **BERT:** Negative (probability $\approx 0.437$).
- **Logistic Regression:** Negative (probability $\approx 0.917$).

Both classify *Negative*. BERT captures the complaint context (e.g., "خاطئه" = "wrong"), whereas LR relies on high-weight TF–IDF tokens (e.g., "كيف", "تم"). LR's higher numeric confidence stems from linear keyword effects; BERT's contextual encoding tends to be more robust for subtler phrasing or negation.

## 5.7 Model Computations

### 5.7.1 BERT (Transformer) Model

**Tokenization & Inputs.** Tweets are tokenized with `asafaya/bert-base-arabic` using [CLS]/[SEP]. For the example:

[[CLS],

للاسف, تم, احتساب, ها, لي, بطريق, ه, خاطي, ه, كيف, يتم, الاعتراض, على, الاحت, ساب, [SEP].[

Table 14: Token IDs and positions for the example tweet (padded to $L = 128$).

| Token | ID | Pos. | Token | ID | Pos. |
|---|---|---|---|---|---|
| [CLS] | 2 | 0 | خاطي | 27282 | 8 |
| للاسف | 8731 | 1 | ه | 1023 | 9 |
| تم | 1975 | 2 | كيف | 2762 | 10 |
| احتساب | 23869 | 3 | يتم | 2429 | 11 |
| ها | 1731 | 4 | الاعتراض | 23108 | 12 |
| لي | 2099 | 5 | على | 1921 | 13 |
| بطريق | 16187 | 6 | الاحت | 2837 | 14 |
| ه | 1023 | 7 | ساب | 2485 | 15 |
| [SEP] | 3 | 16 | [PAD] | 0 | 17–127 |

Each position $i$ uses

$$\mathbf{x}_i = \mathbf{T}_{t_i} + \mathbf{P}_i + \mathbf{S}_{s_i}, \quad X \in \mathbb{R}^{128\times 768}.$$

**Self-Attention.** With 12 heads ($d_k = 64$):

$$A^{(j)} = \text{softmax}\left(\frac{Q^{(j)}(K^{(j)})^\top}{\sqrt{64}}\right), \qquad O^{(j)} = A^{(j)}V^{(j)},$$

$$O_{\text{final}} = [O^{(1)}; \ldots; O^{(12)}]W_O + b_O.$$

**Classifier: Logits and Softmax BERT** After training the model, the classification head parameters and inputs for a sample tweet are:

$$W \approx \begin{bmatrix} -0.0215 & 0.0016 & 0.0189 & \cdots & -0.0273 & 0.0128 & -0.0131 & 0.0164 \\ -0.0127 & -0.0126 & 0.0316 & \cdots & -0.0022 & -0.0069 & 0.0159 & -0.0171 \\ 0.0017 & -0.0036 & -0.0055 & \cdots & 0.0119 & 0.0184 & -0.0023 & 0.0156 \end{bmatrix},$$

$$\mathbf{h}_{[\mathrm{CLS}]} \approx \begin{bmatrix} 0.4319 \\ -1.0997 \\ -1.5461 \\ 0.7930 \\ \vdots \\ -0.0752 \\ -0.7126 \\ 1.9948 \\ -0.4891 \end{bmatrix}, \qquad \mathbf{b} \approx \begin{bmatrix} 0.0002 \\ -0.0007 \\ 0.0001 \end{bmatrix}.$$

**Logits (matrix–vector product + bias).** Coordinatewise,

$$z_c = \sum_{j=1}^{768} W_{cj}\, h_{[\mathrm{CLS}],j} + b_c, \qquad c \in \{1, 2, 3\}.$$

Numerically (from the run):

$$\mathbf{z} \approx \begin{bmatrix} -0.1436 \\ 0.1689 \\ -0.7126 \end{bmatrix}.$$

**Softmax (probabilities).**

$$p_c = \frac{e^{z_c}}{\sum_{k=1}^{3} e^{z_k}}, \qquad c = 1, 2, 3.$$

From the actual BERT model output:

$$\mathbf{p} = \text{softmax}(\mathbf{z}) \approx \begin{bmatrix} 0.99 \\ 0.00 \\ 0.00 \end{bmatrix}, \qquad \text{(up to rounding precision).}$$

**Predicted class.**

$$\hat{y} = \arg\max_{c \in \{1,2,3\}} p_c = \arg\max\{p_{\texttt{Negative}}, p_{\texttt{Neutral}}, p_{\texttt{Positive}}\} = \texttt{Negative} \quad (\text{since } 0.99 \text{ is the largest}).$$

### 5.7.2 Logistic Regression Pipeline

LR uses a sparse TF–IDF vector (up to 20,000 uni/bi-grams). For the same tweet:

$$\mathbf{x} \in \mathbb{R}^{71}, \quad W \in \mathbb{R}^{3\times 71}, \quad \mathbf{b} \in \mathbb{R}^{3}, \quad \mathbf{z} = \mathbf{x}W^{\top} + \mathbf{b}.$$

Numerically, $\mathbf{z} \approx [0.9860, -1.7593, -2.6633]$ for (NEG, NEU, POS), yielding

$$p = \text{softmax}(\mathbf{z}) \approx [0.917, 0.059, 0.024],$$

i.e., **Negative** with 91.7% confidence.

**Interpretation.** LR confidence stems from linearly weighted tokens (e.g., كيف, تم;( BERT's decision exploits contextual cues and tends to be less brittle for nuanced phrasing.

### 5.7.3 Logistic Regression Classifier: Logits and Softmax Computation

After training the model, the learned parameters and inputs for a sample tweet are as follows.

**Weight Matrix.**

$$\mathbf{W} = \begin{bmatrix} -0.0657 & -0.0657 & -0.1034 & -0.1034 & 0.1401 & \cdots \\ 0.0908 & 0.0908 & -0.1034 & -0.1034 & -0.1014 & \cdots \\ -0.0251 & -0.0251 & 0.2069 & 0.2069 & -0.0388 & \cdots \end{bmatrix}_{3\times 71}$$

**Bias Vector.**

$$\mathbf{b} = \begin{bmatrix} 0.4796 & -2.1306 & -1.2501 \end{bmatrix}$$

**Input Feature Vector.**

$$\mathbf{x} = \begin{bmatrix} 0.2199 & 0.2199 & 0.2199 & 0.1825 & 0.2199 & \cdots \end{bmatrix}$$

## Step 1: Linear Transformation (Logits)

The logits are obtained by the standard affine transformation:

$$\mathbf{z} = \mathbf{x}\mathbf{W}^\top + \mathbf{b}.$$

Expanding the operation explicitly:

$$\mathbf{z} = \underbrace{\begin{bmatrix} 0.2199 & 0.2199 & 0.2199 & 0.1825 & 0.2199 & \cdots \end{bmatrix}}_{1\times 71} \underbrace{\begin{bmatrix} -0.0657 & 0.0908 & -0.0251 \\ -0.0657 & 0.0908 & -0.0251 \\ -0.1034 & -0.1034 & 0.2069 \\ -0.1034 & -0.1034 & 0.2069 \\ 0.1401 & -0.1014 & -0.0388 \\ \vdots & \vdots & \vdots \end{bmatrix}}_{71\times 3}$$

$$+ \underbrace{\begin{bmatrix} 0.4796 & -2.1306 & -1.2501 \end{bmatrix}}_{1\times 3}$$

$$= \begin{bmatrix} z_{\text{neg}} & z_{\text{neu}} & z_{\text{pos}} \end{bmatrix} = \begin{bmatrix} 0.9860 & -1.7593 & -2.6633 \end{bmatrix}$$

## Step 2: Softmax Transformation and Probability Calculation

Given the logits:

$$z_{\text{neg}} = 0.9860, \quad z_{\text{neu}} = -1.7593, \quad z_{\text{pos}} = -2.6633,$$

the softmax function converts them to normalized class probabilities:

$$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}}.$$

**Exponentiation.**

$$e^{z_{\text{neg}}} = e^{0.9860} = 2.680,$$

$$e^{z_{\text{neu}}} = e^{-1.7593} = 0.172,$$

$$e^{z_{\text{pos}}} = e^{-2.6633} = 0.070.$$

**Normalization.**

$$\sum_j e^{z_j} = 2.680 + 0.172 + 0.070 = 2.922.$$

**Final Probabilities.**

$$p_{\text{neg}} = \frac{2.680}{2.922} = 0.917,$$
$$p_{\text{neu}} = \frac{0.172}{2.922} = 0.059,$$
$$p_{\text{pos}} = \frac{0.070}{2.922} = 0.024.$$

**Result.**

$$\text{softmax}(\mathbf{z}) = \begin{bmatrix} 0.917 \\ 0.059 \\ 0.024 \end{bmatrix} \Rightarrow \hat{y} = \textbf{Negative (91.7\%)}.$$

**Interpretation.** The model assigns the highest probability (91.7%) to the *Negative* class, indicating a strongly negative sentiment. Each token contributes via its learned coefficient $\beta_{t,k}x_t$ to the corresponding logit $z_k$. For example, words like كيف ("how") increase $z_{\text{neu}}$ but lower $z_{\text{pos}}$, reflecting their association with neutral or complaint-type expressions.

# 6 Conclusion

This thesis examined whether Large Language Models (LLMs) outperform traditional statistical models for Arabic text classification, using Saudi bank sentiment analysis as a case study. We compared a fine-tuned BERT model with a TF–IDF Logistic Regression baseline under a unified probabilistic view: both models map text to logits and class probabilities via a linear decision layer and softmax, but they differ in how they represent the input (contextual embeddings vs. sparse TF–IDF features).

Empirically, BERT achieved higher Accuracy and Macro-$F_1$ than Logistic Regression, indicating that contextual representations can improve class balance under label skew and better capture nuanced sentiment, particularly for minority classes. At the same time, Logistic Regression remained strongly competitive: it delivered solid overall performance with a much simpler pipeline, faster training/inference, and clear interpretability through feature weights. In practical settings where transparency, deployment constraints, or limited compute are priorities, traditional models may therefore be the preferred choice despite the performance gap.

Overall, the results support the alternative hypothesis that LLMs can outperform traditional models in Arabic sentiment classification due to richer contextual features. However, the competitiveness of Logistic Regression confirms an important takeaway: classical statistical models remain a strong and reliable baseline, and core statistical principles—logits, likelihood-based loss functions, calibration, uncertainty, and rigorous evaluation—remain central for understanding and deploying *both* modern LLM-based classifiers and traditional methods.

# 7 Future Work

Several extensions can build on this thesis:

1. **Richer baselines:** Compare BERT with additional traditional models such as linear SVMs, Naïve Bayes, or tree-based ensembles to obtain a broader performance baseline.

2. **More LLM variants:** Evaluate different Arabic LLMs (e.g., AraBERT variants, distilled or larger models) to study the trade-off between model size, accuracy, and latency.

3. **Better handling of imbalance:** Explore class-weighting, focal loss, or data augmentation to improve performance on underrepresented Neutral and Positive classes.

4. **Deeper interpretability:** Apply attribution and attention-based analysis to better understand how BERT and Logistic Regression make decisions on real Saudi banking tweets.

# References


[1] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. https://nlp.stanford.edu/IR-book/

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint arXiv:1301.3781, 2013. https://arxiv.org/abs/1301.3781

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. *GloVe: Global Vectors for Word Representation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543. https://aclanthology.org/D14-1162/

[4] D. Alqahtani, L. Alzahrani, M. Bahareth, N. Alshameri, H. Al-Khalifa, and L. Aldhubayi, *Customer Sentiments Toward Saudi Banks During the Covid-19 Pandemic*, Proceedings of the International Conference on Computer and Information Sciences, 2022.

[5] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory*. Neural Computation, 9(8):1735–1780, 1997. https://doi.org/10.1162/neco.1997.9.8.1735

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. *Attention is All You Need*. Advances in Neural Information Processing Systems (NeurIPS), 2017. https://arxiv.org/abs/1706.03762

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805, 2018. https://arxiv.org/abs/1810.04805

[8] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. OpenAI Blog, 1(8), 2019.

https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[9] Wissam Antoun, Fady Baly, and Hazem Hajj. *AraBERT: Transformer-based Model for Arabic Language Understanding*. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, LREC 2020. https://aclanthology.org/2020.osact-1.7/

[10] Ibrahim Abu Farha and Walid Magdy. *Benchmarking Transformer-based Language Models for Arabic Sentiment and Sarcasm Detection*. In Proceedings of the Sixth Arabic Natural Language Processing Workshop, ACL 2021. https://aclanthology.org/2021.wanlp-1.12/

[11] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, et al. *On the Opportunities and Risks of Foundation Models*. arXiv preprint arXiv:2108.07258, 2021. https://arxiv.org/abs/2108.07258

[12] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Lulu.com, 2022. https://christophm.github.io/interpretable-ml-book/