1. Background

1.1 Text Embeddings

In natural language processing, text must be transformed into numerical representations before it can be consumed by a model. One classical representation is the bag-of-words (BoW) model, in which documents are encoded as vectors of term frequencies, which discard word order but retain counts of each term. However, term frequencies (TF) vary depending on context, topic, domain, style, and so on. This nonstationarity implies that the frequencies of words we observe are averages over contexts. As Piantadosi (2014) formalizes, the probability of uttering a word w is given by:

(1) $P(W = w) = \sum_c P(c)P(W = w|C = c)$ *where …*

$P(W = w|C = c)$ *is the probability of w in context c;*

$P(c)$ is the probability distribution over contexts.

Equation 1: Probability of an Observed Utterance (Piantadosi, 2014).

This equation shows that observed word usage reflects a mixture over hidden contexts, motivating the need for weighted representations like TF×IDF, Term Frequency-Inverse Document Frequency, which account for both local term relevance and global rarity across documents:

(2) $\mathrm{TF}(t,d) = \mathrm{count}(t,d)$

(3) $\mathrm{IDF}(t) = 1 + \ln\left(\frac{N+1}{DF(t)+1}\right)$

Equations 2 & 3: Term Weight as Count (Term Frequency) and Inverse Document Frequency.

Word frequency distributions in natural language often follow a Zipfian or near-Zipfian distribution, where the frequency f of a word is inversely proportional to its rank r in the sorted frequency list. The classical form of Zimpf's Law in equation four characterises term frequencies in terms of their relative proportions to reflect how observed frequencies vary with corpus size (Zipf, 1936, 1949). In practice, "the most frequent word $(r = 1)$ has a frequency proportional to 1, the second most frequent word $(r = 2)$ has a frequency proportional to $\frac{1}{2^\alpha}$, the third most frequent word has a frequency proportional to $\frac{1}{3^\alpha}$, and so forth," i.e., (Piantadosi, 2014):

(4) $f(r) \propto \frac{1}{r^\alpha}$ *where …*

Rank r is a positive integer: $r \in \mathbb{N}, r \geq 1;$

Decay rate α controls how steeply frequency drops with rank: $\alpha \geq 0, \alpha \approx 1.$

Equation 4: Zipf's Law (Zipf, 1936, 1949).

Zipf's Law describes a power-law decay in rank frequency (i.e., as rank r increases, frequency $f(r)$ decreases polynomially), or equivalently:

(5) $y = C \cdot x^{-\alpha}$ where $x = r, y = f(r),\ \alpha > 0$

Equation 5: Power Law decay for Rank Frequency.

This implies that a small subset of words occurs extremely frequently, while the vast majority of words are rare. However, power-laws decay slowly and tend to have long tails, which often causes the original formulation to overestimate the frequency of rare low-ranked words. To address this limitation, Piantadosi highlights the "current incarnation" of the Zipfian (i.e., near-Zipfian) expression, the Zipf-Mandelbrot law, a generalisation that introduces parameter shift β to create a more accurate empirical fit (Piantadosi, 2014).

(6) $f(r) \propto \frac{1}{(r+\beta)^\alpha}$

Equation 6: Zipf-Mandelbrot Law (Mandelbrot, 1953, 1962; Zipf, 1936, 1949).

1.2 Text Prediction and Model Evaluation

We evaluate the performance of a classical predictive classifier trained on TF representations using a combination of foundational statistics: confusion matrices, accuracy, precision, recall, Fscore (or F1 score), and log loss. A confusion matrix is a tabular representation of a classifier's performance that records the number of correct and incorrect predictions for each class, providing a granular view of inter-class errors (Fawcett, 2006). From this, key evaluation metrics can be derived: Accuracy measures the overall proportion of all correct predictions but can be misleading in imbalanced settings. Precision class agreement of the data labels with the positive labels given by the classifier, while recall measures the effectiveness of a classifier to identify positive labels (Sokolova & Lapalme, 2009).

(7) $Accuracy = \frac{TP+TN}{TP+FN+FP+TN}$,

$Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$

Equation (Set) 7: Accuracy, Precision, and Recall**.**

The Fscore is the harmonic mean of precision and recall, offering a balanced metric when both false positives and false negatives are relevant (Bullen, 2003). In another way of understanding it, the Fscore communicates the relations between data's positive labels and those given by a classifier (Sokolova & Lapalme, 2009):

(8) $Fscore = \frac{(\beta^2+1) \cdot TP}{(\beta^2+1) \cdot TP + \beta^2 FN + FP}$

Equation 8: Fscore, the Harmonic Mean of Precision and Recall (Bullen, 2003; Sokolova & Lapalme, 2009)**.**

Next, we remind ourselves of the theoretical quantities that allow the log loss to evaluate a classifier's probabilistic confidence by penalising incorrect predictions with high certainty, thus making it a sensitive measure of model calibration. Equation 9 shows that a likelihood is a function to measure how well a model explains observed data (Knott, 2025). In the Bayesian framework, this relationship is formalised via Baye's Theorem (Equation 10). Here, the likelihood function $\mathcal{L}(\theta)$ can be represented as $P(\theta|\text{data})$, the probability of model parameters theta, given observed data. Despite the likelihood's formal definition [from probability theory] as a conditional probability of the data given the parameters, it can be reinterpreted as a function θ, with the observed data held fixed. This distinction allows it to serve as the central quantity in maximum likelihood estimation (MLE), where likelihood is maximised with respect to θ, and in Bayesian inference, where it shapes the posterior distribution (Kleijn, 2024). Mathematically, this is expressed as:

(9) $\mathcal{L}(\theta) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta)$

Equation 9: Likelihood as the Joint Probability of Observed Class Labels (Knott, 2025).

(10) $P(\theta|\text{data}) \propto P(\text{data}|\theta) \cdot P(\theta)$

Equation 10: Posterior Factorised as Likelihood × Prior (Kleijn, 2024).

As Knott 2025 states: "Where all observed corpus events are assumed to be independent, a parameterisable model $L(\theta)$ predicts the total probability of all occurrences and iteratively updates its parameters to make this probability as high as possible" (Knott, 2025). This statement describes the statistical inference method for MLE, where we aim to assess the plausibility of model parameters θ given observed data.

(11) $\theta^* = \arg\max_{\theta} \prod_{n \in A} p(x^{(n)}|\theta)$ *for …*

Independent Experiments: $p(x^{(1)}, \dots, x^{(|A|)}) = \prod_{n \in A} p(x^{(n)})$

Equation 11: MLE Objective Under Independent Observations.

However, because multiplying many small probabilities leads to numerical instability and is difficult to optimise, we instead take the log of the likelihood, turning products into sums to simplify optimisation. This is referred to as the maximum log likelihood (ML) method. It is ubiquitous and does not interfere with the selection of optimal θ (Kleijn, 2024).

(12) $\log \mathcal{L}(\theta) = \sum_{i=1}^{n} \log p(y^{(i)}|x^{(i)};\theta)$

Equation 12: Log-Likelihood Function for Independent Observations.

(13) $\theta^* = \arg\max_{\theta} \sum_{n \in A} \log p(x^{(n)}|\theta)$

Equation 13: Maximum Log Likelihood (ML) Objective Under Independent Observations.

In supervised classification, we want to measure how well our model's predicted probabilities align with the true labels. For this, we can also use the log loss (i.e., cross entropy loss). Unlike accuracy, which simply checks whether the predicted class label is correct, log loss penalises incorrect predictions more harshly when the model is confident, but wrong. The log loss is defined as:

(14) $\mathcal{LL}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log p(y^{(i)}|x^{(i)};\theta)$ *where …*

$x^{(i)}$ *is an input feature vector;*

$y^{(i)}$ *is the true class label;*

$p_\theta(y|x)$ is the model's predicted probability of class y.

Equation 13: Log-Loss Function for Independent Observations.

Equation 13 simply expresses the negative average log-likelihood per sample. Moreover, as rigorously established in many standard references on probabilistic modelling and machine learning, minimising the log loss is equivalent to maximising the log-likelihood because minimising the negative of a function is equivalent to maximising the function itself (Murphy, 2012).

2. TF Classifier

2.1 Model Summary & Data Preprocessing

Model one trained a Multinomial Naïve Bayes classifier using <u>only TF representations</u>.

Before modelling a text classifier built on TF representations, we first verified that the training corpus conformed to a Zipfian distribution, such that we might validate the data's quality and select an appropriate vocabulary size for training.

The results of this fitting (Figure 1) indicated that the Zipf-Mandelbrot model had a slightly stronger fit $(R^2 = 0.9906)$ to the empirical distribution than the Zipf model $(R^2 = 0.9839)$, which slightly underestimated the frequency of common words and overestimated the frequency of rare words $(\alpha = 0.8290)$. In the Zipf-Mandelbrot model, the β-shift reduced the overemphasis of the top 1-2 words and α balanced the decay rate more precisely across the rank range $(\alpha = 0.9320)$. This suggests that the that the frequency structure of the training corpus is more accurately captured by a near-Zipfian distribution, with 99.1% of the variance explained by the Zipf-Mandelbrot model.

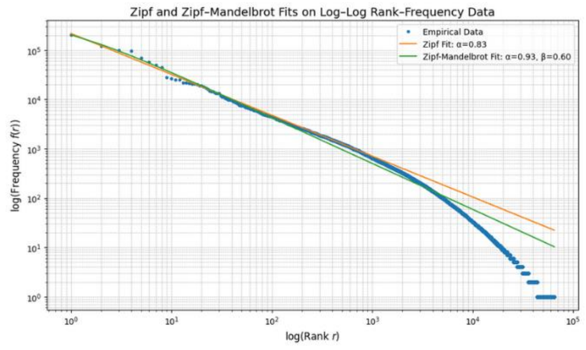


Figure 1: Zipf and Zipf-Mandelbrot Fit on Log-Log Rank-Frequency Data.

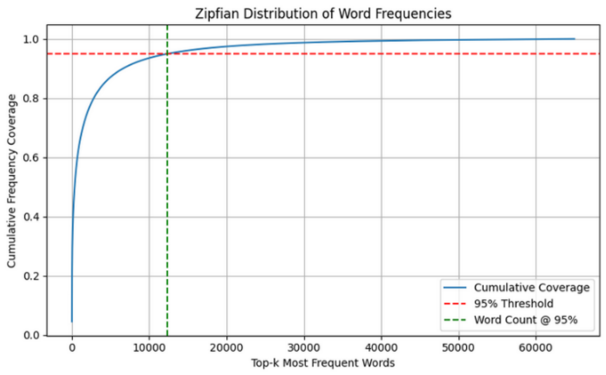


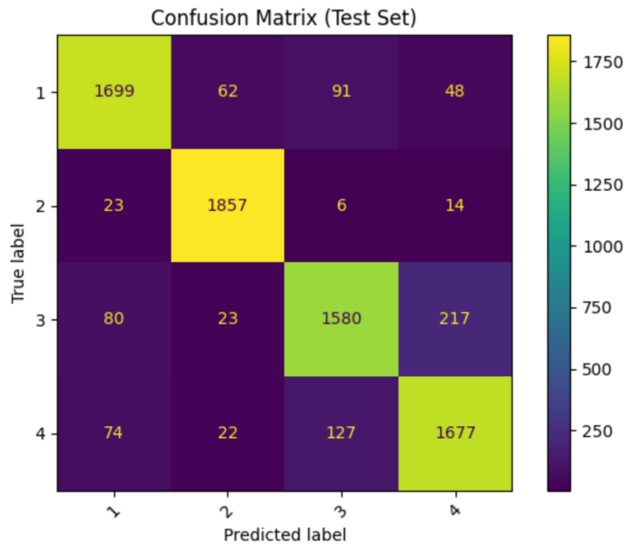
Figure 2: Cumulative distribution of term frequencies sorted by descending rank.

Moreover, to quantify the impact of this distribution on term coverage, we performed a Zipfian analysis of our training corpus by plotting the cumulative distribution of term frequencies sorted by descending rank. This allowed us to identify a lexical threshold for the minimum number of distinct terms required to cover a specified proportion of the total token mass (Figure 2). For this particular training corpus, we found that the top 12,382 most frequent terms account for 95% of all term occurrences.

This result indicates that although the raw vocabulary may contain over 65,000 unique terms, the majority of word frequency mass was concentrated in a much smaller subset, as expected. In this way, we were able to limit the vocabulary size to the top 13,000 ranks to reduce model complexity and memory footprint while preserving the most useful signal for classification based on TF representations.

2.2 Model Results

The NB classifier trained on TF representations achieved strong overall performance, with a training accuracy of 90.67% and a test accuracy of 89.64%. This relatively small performance gap suggests that the model generalised well to unseen data.



	1	2	3	4
1	1699	62	91	48
2	23	1857	6	14
3	80	23	1580	217
4	74	22	127	1677

```
              precision    recall  f1-score

           1       0.91      0.89      0.90
           2       0.95      0.98      0.96
           3       0.88      0.83      0.85
           4       0.86      0.88      0.87

    accuracy                           0.90
   macro avg       0.90      0.90      0.90
weighted avg       0.90      0.90      0.90
```

Figure 3: Model One Confusion Matrix and Classification Report.

The confusion matrix (Figure 3) and classification report found that class label two had the highest precision and recall at 0.95 and 0.98, indicating that the model had a strong signal for this class. Class three exhibited the most misclassifications at a slightly lower precision and recall of 0.88 and 0.83, suggesting possible semantic overlap or vocabulary similarity between classes three and four. The macro-averaged F1 score is 0.90, indicating balanced performance across classes despite minor inter-class confusion. The log loss of the model was ≈ 1.1239. In the context of a four class problem, the theoretical log loss of random assignment (i.e., uniform probability) is $-\log\left(\frac{1}{4}\right) = \log(4) \approx 1.386$. This means that our model performs better than randomly guessing class labels. Combining all of our results, we can conclude that the model is often right, but not always confident when right, and occasionally confident when wrong.

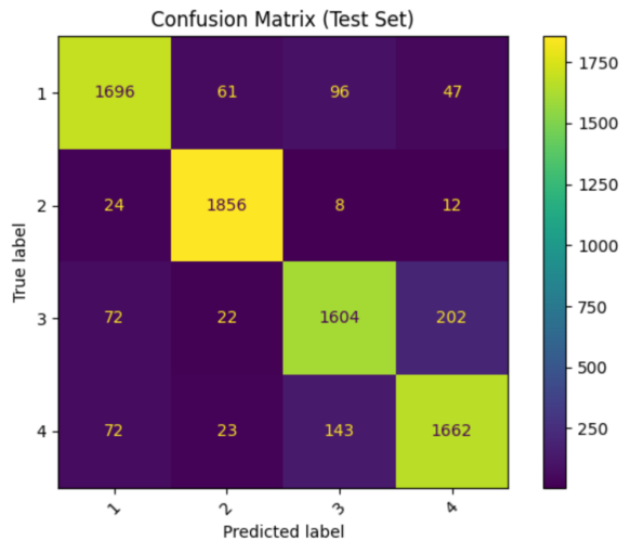
3. TF-IDF Classifier

3.1 Model Summary

Model two trained a Multinomial Naïve Bayes classifier using TF-IDF representations.

3.2 Model Results

The NB classifier trained on TF-IDF representations achieved slightly better performance than model one. The overall training accuracy was 90.90% and the test accuracy was 89.71%, again indicating that the model two generalised equally well to unseen data as model one.



```
              precision    recall  f1-score

           1       0.91      0.89      0.90
           2       0.95      0.98      0.96
           3       0.87      0.84      0.86
           4       0.86      0.87      0.87

    accuracy                           0.90
   macro avg       0.90      0.90      0.90
weighted avg       0.90      0.90      0.90
```

Figure 4: Model Two Confusion Matrix and Classification Report.

The confusion matrix (Figure 4) and classification report found that class label two remained the highest in precision and recall (0.95 and 0.98). Class three exhibited the most misclassifications at a slightly lower precision and recall of 0.87 and 0.84, suggesting that model two very marginally improved classification for class 3, its weakest class, as also indicated by a significant decrease in the test log loss at ≈ 0.318. The macro-averaged F1 score remained the same (0.90), indicating balanced performance across classes despite some inter-class confusions. Combining this result with our confusion matrix, classification report, and high rate of correct top-class predictions (accuracy), we can conclude that the performance differences between models one and two are negligible.

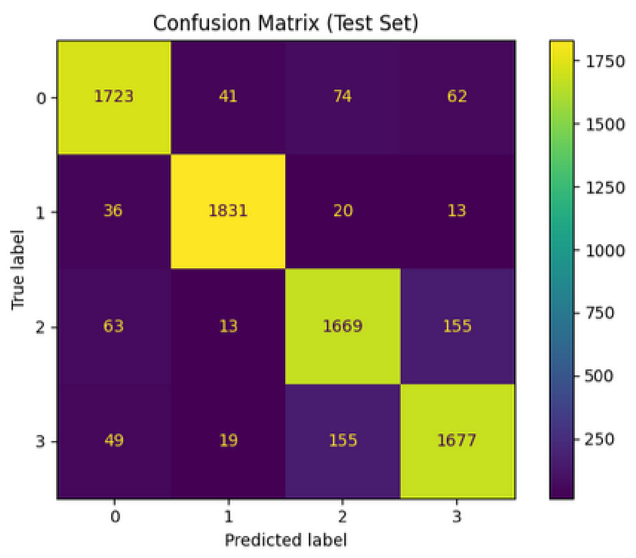
4. CNN Classifier

4.1 Model Summary

Model three was a Convolutional Neural Network (CNN)-based text classifier implemented using JAX/Flax. It began with a trainable embedding layer that mapped each of the 70 input tokens to a 128-dimensional vector space. The embedded sequence was then processed by a 1D convolutional layer with 128 filters and a kernel size of 5, followed by a ReLU activation. A global average pooling layer reduced the temporal dimension, and the resulting vector was passed through a dense (fully connected) layer that output logits for the 4 target classes. Logits represent unnormalised scores used to calculate the final classification probabilities via a softmax function. Here, softmax activation was applied implicitly through the use of a cross-entropy loss function to suit the multiclass nature of the problem. The number of filters, kernel size, padding/truncation, and global average pooling convolutional approach were informed by the lecture example.

4.2 Model Results

Model three's performance was comparable to model two. Model three's overall training and testing accuracy reached 98.05% and 90.79%, respectively. The log loss was a bit higher than the NB/TF-IDF representations, despite some improvements in class 3 and class 4 classifications (Figure 5). Overall, our baseline CNN trained on token embeddings performed similarly to our NB/TF-IDF model, but took approximately 2-3x's longer to run.



```
              precision    recall  f1-score

           0       0.92      0.91      0.91
           1       0.96      0.96      0.96
           2       0.87      0.88      0.87
           3       0.88      0.88      0.88

    accuracy                           0.91
   macro avg       0.91      0.91      0.91
weighted avg       0.91      0.91      0.91
```

Figure 5: Model Three Confusion Matrix and Classification Report.

Model three also reported the highest F1-score of 0.91 compared to all other models at 0.90, however this may not represent a statistically significant difference in overall precision.

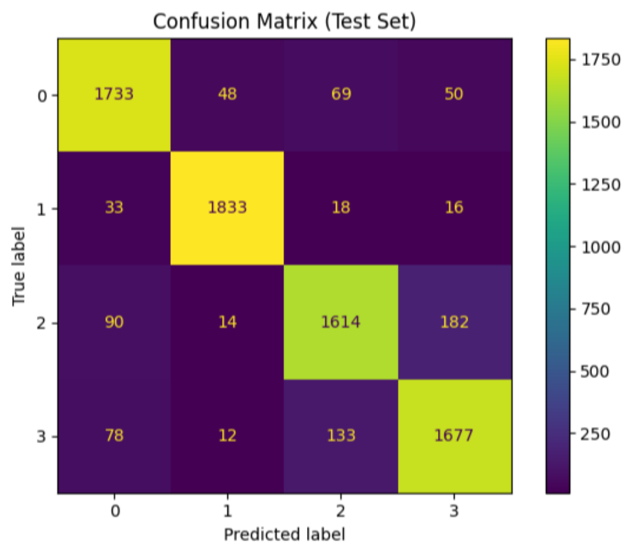
5. CNN Classifier (GloVe Embeddings)

5.1 Model Summary

Model four was another CNN-based text classifier implemented using JAX/Flax. It began with a non-trainable embedding layer initialised with 100-dimensional pre-trained GloVe embeddings. Each input sequence was padded/truncated to 53 tokens (adapted to 95th percentile). The embedded input was processed by a 1D convolutional layer with 64 filters and a kernel size of 3, followed by a ReLU activation. A global max pooling layer was used in place of an average pooling layer to reduce the temporal dimension to fixed-size representation. This intermediate output then passed through another dense layer to produce logits for the 4 target classes. We once again applied a softmax distribution implicitly using the multiclass cross-entropy loss function during training.

5.2 Model Results

Model four performed significantly better than previous models. It achieved a final training and testing accuracy of 96.92% and 90.22%, respectively.



```
              precision    recall  f1-score

           0       0.90      0.91      0.90
           1       0.96      0.96      0.96
           2       0.88      0.85      0.86
           3       0.87      0.88      0.88

    accuracy                           0.90
   macro avg       0.90      0.90      0.90
weighted avg       0.90      0.90      0.90
```

Figure 6: Model Four Confusion Matrix and Classification Report.

There remained similar levels of misclassification between classes three and four (Figure 6). However, the log loss was 0.2884, meaning the model had higher confidence in its correct predictions. Further, the model's run time was halved compared to model three, making it both more accurate and efficient.

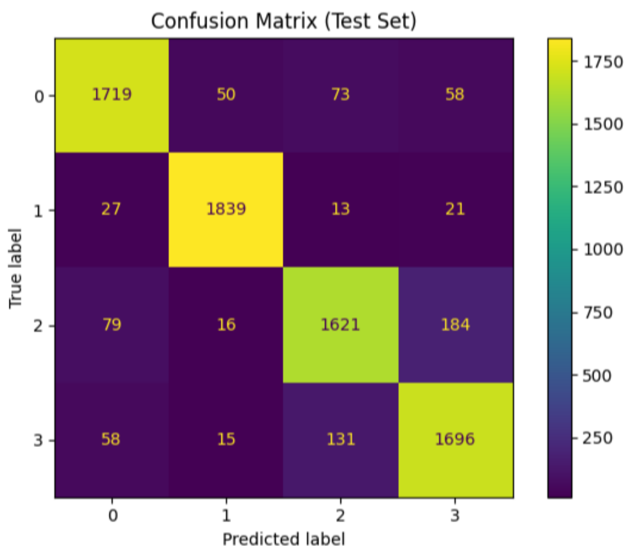
6. CNN Classifier (Modified)

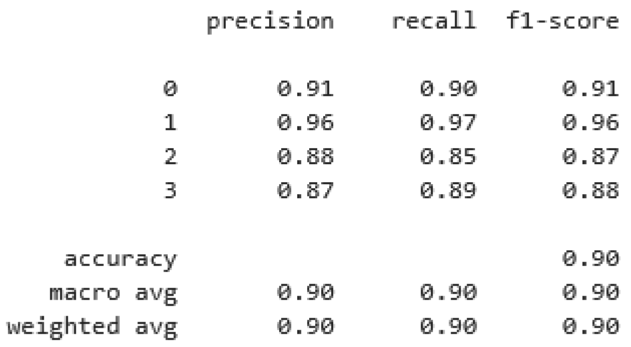
6.1 Model Summary

Model five was another CNN-based text classifier implemented in JAX/Flax. It used the same core architecture as model four, but used training, testing, and validation sets, and introduced the Natural Language Tool Kit (NLTK) library for data preprocessing. This model used NLTK tools to enhance data preprocessing by excluding stopwords (e.g., "the", "is", "in") and cleaning stemmed words (e.g., "jumping"→"jump", "loved"→"love", etc.). Here, stemming was applied to each word in the cleaned test after the removal of stopwords .

6.2 Model Results

Model five achieved a training accuracy of 95.11% and a validation accuracy of 90.97%, which were the highest of all models. The confusion matrix and classification report (Figure 7) show class label 1 maintained the highest precision (0.96) and recall (0.97), while class 2 continued to be the most difficult to classify (recall of 0.85) throughout all practical experiments. The model achieved a log loss of 0.297, which is a relatively negligible loss compared to the previous model. As with most models, the macro-averaged F1 score remained stable at 0.90, reflecting balanced performance across all classes. Collectively, these results suggest that the architectural and preprocessing enhancements improved model calibration and robustness without sacrificing overall accuracy or efficiency.



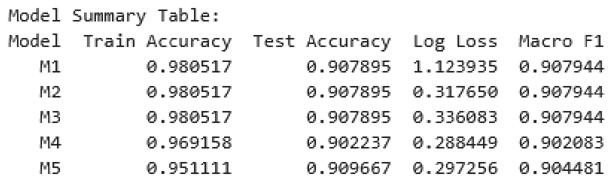


```
              precision    recall  f1-score

           0       0.91      0.90      0.91
           1       0.96      0.97      0.96
           2       0.88      0.85      0.87
           3       0.87      0.89      0.88

    accuracy                           0.90
   macro avg       0.90      0.90      0.90
weighted avg       0.90      0.90      0.90
```

Figure 7: Model Five Confusion Matrix and Classification Report.

7. Conclusion (Summary)

Across all models, the CNN classifiers outperformed classical NB approaches in confidence and adaptability, particularly when enhanced with GloVe embeddings and preprocessing techniques. Model five achieved the best overall performance, and successfully balanced accuracy, log loss, and efficiency. These findings underscore the value of data preprocessing, robust embeddings, and careful attention to model design in modern text classification tasks.



```
Model Summary Table:
Model  Train Accuracy  Test Accuracy  Log Loss  Macro F1
   M1        0.980517       0.907895  1.123935  0.907944
   M2        0.980517       0.907895  0.317650  0.907944
   M3        0.980517       0.907895  0.336083  0.907944
   M4        0.969158       0.902237  0.288449  0.902083
   M5        0.951111       0.909667  0.297256  0.904481
```

Table 1: All Model Summary Table.

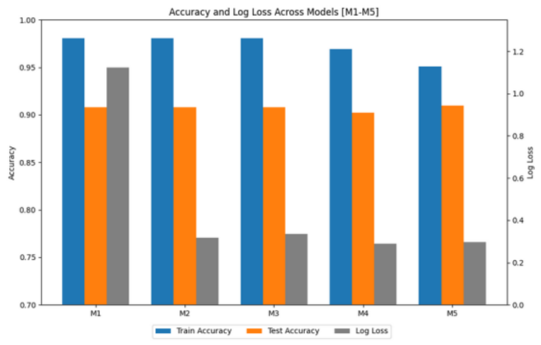


Figure 8: Accuracy and Log Loss Across Models.

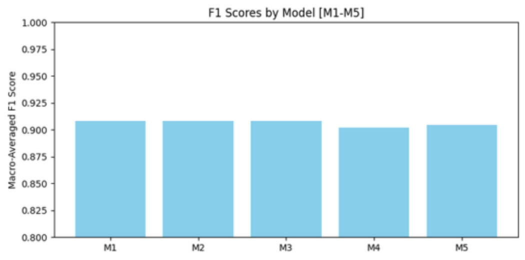


Figure 9: F1 Scores by Model.

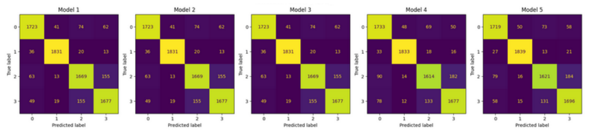


Figure 10: All Model Confusion Matrices.

8. References

Bullen, P. S. (2003). The Arithmetic, Geometric and Harmonic Means. In P. S. Bullen (Ed.), *Handbook of Means and Their Inequalities* (pp. 60–174). Springer Netherlands. https://doi.org/10.1007/978-94-017-0399-4_2

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/https://doi.org/10.1016/j.patrec.2005.10.010

Kleijn, B. (2024). Brief Review Probability Theory and Basic Information Theoric Quantities as Used in Deep Learning: AIML425 [Lecture 1 Notes]. *Master of Artificial Intelligence, Victoria University of Wellington*, 16–20. https://ecs.wgtn.ac.nz/foswiki/pub/Courses/AIML425_2024T2/LectureSchedule/probability.pdf

Knott, A. (2025). GPT History 1: Neural network language models [AIML428 Lecture Notes]. *Master of Artificial Intelligence, Victoria University of Wellington*.

Mandelbrot, B. B. (1953). An informational theory of the statistical structure of language. In W. Jackson (Ed.), *Communication Theory* (pp. 486–502). Butterworths.

Mandelbrot, B. B. (1962). On the theory of word frequencies and on related Markovian models of discourse. In *Structure of Language and its Mathematical Aspects* (Vol. 12, pp. 190–219). American Mathematical Society.

Murphy, K. P. (2012). Graphical Models and Dynamic Bayesian Networks. In *Machine Learning: A Probabilistic Perspective* (pp. 247–290). MIT Press.

Piantadosi, S. T. (2014). Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, *21*(5), 1112–1130. https://doi.org/10.3758/s13423-014-0585-6

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/https://doi.org/10.1016/j.ipm.2009.03.002

Zipf, G. (1936). The Psychobiology of Language. *London: Routledge*.

Zipf, G. (1949). Human Behavior and the Principle of Least Effort. *New York: Addison-Wesley*.