Ayra Islam
Student ID: 490470589

OCMP5318: Assignment 2

Text Classification on AG News Dataset

Written By: Ayra Islam

Ayra Islam
Student ID: 490470589

Introduction

Overview

This report aims to demonstrate the effectiveness of neural networks and unsupervised machine learning methods for text classification tasks. The study leverages the AG News Dataset, a collection of over 1 million news articles gathered from more than 2000 sources, compiled by ComeToMyHead. The dataset was created by Xiang Zhang for academic research in data mining. The dataset is split into 120,000 training and 7600 test samples. Each article sample is labeled by news type: World, Sports, Business, and Science/Technology (Zhang, 2024). Categories are classified from 1 to 4, respectively.

Background

Text classification has several applications across various industries, including spam detection, news classification, and sentiment analysis. It has been a prominent topic in Natural Language Processing (NLP) research for decades, and has played a major role in the growing scale of text data in recent years. A popular approach for text classification is the use of deep learning models; however, the motivation behind this report is to demonstrate and compare how traditional and state-of-the-art machine learning methods address the recurring challenges of text classification problems, including contextualization and word preprocessing (Minaee et al., 2021).

Problem Statement

The primary goal is to develop and evaluate models for classifying news articles into their correct class labels (World, Sports, Business or Sci/Tech). The input of this project is a collection of articles, with each article being a sequence of text-based news content. The output is one of the predicted news classes. The primary challenge is to apply appropriate preprocessing methods and predictive tuning to ensure the built models can accurately classify news articles. This report evaluates the performance of both deep learning and clustering-based methods for this task.

Methodology

Data Preprocessing

Text Cleaning for Natural Language Processing (NLP) is essential for enhancing prediction results. It is used to enhance data quality, standardise texts, and improve readability for both humans and machines.

The input data has four columns: index, class label, title, and description.
Columns title and description were combined into a single column for both training and tests to simplify subsequent preprocessing steps. The class label was separated from this column

and rescaled to be zero-indexed, e.g., {1,2,3,4} to {0,1,2,3} to align with machine learning libraries that require class labels to begin at zero.

Common NLP preprocessing methods from the NLTK package were applied to standardise and remove redundant text. This included lowercasing, removal of basic punctuation, numbers, URLs, tokenisation, and stopword removal. Stopwords consisted of the most frequently used English words and HTML entities such as gt (greater than) and quot (quotation). Finally, Word Net Lemmatizer was applied by reducing words to their base words. This method improved predictive performance compared to conventional stemming because words were truncated into existing words, thus preserving their meaning (Silva, 2023)

Different feature preprocessing and dimensionality reduction techniques were applied for neural networks and clustering algorithms based on their input requirements.
Text sequences were tokenized and padded for neural networks to create fixed-length numerical arrays, ensuring the input data was suitable for model architecture.

In contrast, term frequency-inverse document frequency (TF-IDF) vectorisation was applied before building clustering models. TF-IDF emphasises important words depending on their frequency within a document relative to other documents (Coder, 2023). Highlighting keywords reduces unwanted noise generated from more frequent, insignificant words, thus making it easier for clustering methods to find textual commonalities. Parameter `max_features` was set to 5000 to restrict word count for vectorization whilst `n_grams` was set to find unigram and bigram sequences, since some individual words such as ‘new’ and ‘york’ were naturally associated with each other. Choosing both parameters ensured a balance was made between computational efficiency and semantic depth.

Furthermore, Truncated Singular Value Decomposition (SVD) was used to reduce the dataset from 1000 features to 2 components to address clustering sensitivity to high-dimensional data. Truncated SVD reduces data by creating linear combinations from raw data called singular vectors and is often used for text classification tasks. This method is preferred over other dimension-reduction methods due to its ability to handle large and sparse data (Khadka, 2023).

Methods

Chosen Models

A neural network is a machine learning model composed of multiple layers of interconnected units used to perform tasks such as classification or prediction. The typical architecture includes an input layer, one or more hidden layers, and an output layer. Each unit or neuron performs a linear transformation on the input, followed by a non-linear activation function, allowing the network to model complex relationships in the data.

Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specialized type of neural network often used for image and spatial data, though it can be adapted for text as well. Unlike standard neural

networks, CNNs use convolutional layers that apply small grids known as filters or kernels across the input. These filters scan over different regions of the input to detect important local patterns, producing a feature map that highlights significant information whilst reducing the input's dimensionality.

Padding is often applied to the input to maintain the original dimensions after convolution. Afterward, an activation function (e.g., ReLU) is used to introduce non-linearity, enabling the model to learn more complex patterns and relationships. The feature map is then downsampled using pooling layers. The layer aggregates information from small regions to reduce the spatial dimensions and computation cost while preserving important features.

Finally, the output is flattened into a one-dimensional vector and passed through one or more fully connected layers, which apply linear transformations using learned weight matrices to perform the final classification.

Long Short-Term Memory

Long Short-Term Memory (LSTM) is a specialized type of Recurrent Neural Network (RNN). RNNs are deep learning models designed to handle sequential or time-series data, capable of making predictions based on patterns in earlier inputs. A key downside of RNNs is their susceptibility to vanishing or exploding gradients, whereby updates to gradients can grow either too large or too small. This can make it difficult to retain or update information from earlier time steps during training. LSTMs resolve this issue by incorporating a gating mechanism that allows the network to selectively retain or discard information.

An LSTM network consists of a cell state, which controls the flow of sequential information and is modified by the gating mechanism. Its gating mechanism involves

- Forget gate: Decides which information to discard by comparing the previous input with the current input.
- Input gate: Determines which part of the new information can be retained.
- Output gate: Outputs information for the next time step.

K-means Clustering

K-means clustering is a popular unsupervised learning algorithm that partitions data points into k distinct groups (or clusters) based on their distance to a central point known as a centroid. A centroid represents the average position of all points within a cluster and serves as a summary of that cluster’s characteristics. The main objective of K-means is to minimize the within-cluster variance—in other words, to reduce the distance between data points and their assigned centroids—so that clusters are as compact and distinct as possible.

Mathematically, this can be expressed as:

$$minimise \sum_{i=1}^{k} \sum_{x \in C_i} \left|\left| x - u_i \right|\right|^2$$

whereby k denotes the number of clusters, C_i referring to the ith cluster, x referring to a training point in the cluster and u_i referring to the centroid of the ith cluster.

The algorithm proceeds as follows:

1. Select a value for k and randomly choose k data points as initial centroids.
2. Assign each data point to the nearest centroid based on a distance metric (typically Euclidean distance).
3. Update the centroids by computing the mean of all points assigned to each cluster.
4. Repeat steps 2 and 3 until the centroids stabilize (i.e., they no longer change significantly), indicating convergence.

Choosing an appropriate k is crucial, as it directly influences the quality of clustering. Additionally, random initialization of centroids can sometimes lead to poor convergence or suboptimal clusters. To address this, the K-means++ algorithm introduces a smarter initialization strategy that spreads out the initial centroids more effectively, helping the model converge faster and improving clustering performance (Sharma, 2021).

Gaussian Mixture Models

Gaussian Mixture Models (GMM) represent each cluster using a Gaussian (normal) distribution. Unlike K-means, which assigns each observation to exactly one cluster, GMM estimates the probability that a data point belongs to each cluster. The final assignment is based on the weighted sum of the probabilities across all Gaussian components:

$$p(x) = \sum_{i=1}^{k} \pi_k N\left(x|\mu_k, \sum_k \right)$$

- $p(x)$ referring to the probability density for data point x,
- k denoting the number of clusters/Gaussian component
- N referring to the Gaussian probability density function with mean μ_k and variance $\sum_k$.
- π_k as the prior probability for component k

GMM uses the Expectation-Maximization (EM) algorithm to iteratively estimate the parameters (mean, covariance, and mixing weights) of each Gaussian distribution. The goal is to maximize the likelihood that the observed data was generated from the estimated mixture of Gaussians. Due to this probabilistic approach, GMM can produce more flexible clusters than K-means. While K-means tends to form spherical clusters, GMM can model elliptical shapes and better capture the underlying structure of data, especially when clusters vary in size and orientation (Olamendy, 2024).

Justification for Chosen Models

Convolutional Neural Networks

Despite specialisation with image data, in recent research, CNNs have been shown to perform extremely well in text classification tasks. Words in a sequence are converted into a numerical vector known as a word embedding, which holds the semantic meaning of the sequence. Subsequently, filters are applied to extract key features from the text. CNNs are less sensitive to word order, making them more efficient than LSTMs for certain tasks. They are also typically faster to train due to simpler architectures. They also tend to generalize better on smaller datasets, like product reviews or tweets, due to fewer trainable parameters (*RNN vs. CNN: Understanding Key Differences in Text Classification*, 2024).

Long Short-Term Memory

RNNs, including LSTMs, are a natural choice for text classification due to the sequential nature of words in a sentence. LSTMs preserve global context by retaining information across longer sequences, which is beneficial for tasks requiring a deeper understanding of sentence structure or meaning. However, this can lead to longer training times and greater data requirements than CNNs. Common applications requiring more text include machine translation of one language to another and text summarisation (*RNN vs. CNN: Understanding Key Differences in Text Classification*, 2024).

K-Means Clustering

K-means clustering was chosen as a baseline model against deep learning methods. It is a popular method of choice given its simplicity, computational speed, and interpretability of results. However, it assumes spherical clusters of similar size, making it difficult to capture overlapping topics. Furthermore, it struggles to handle sparse data, which is usually the case for textual data. Despite this, studies have shown reasonable capture of words by K-means when coupled with techniques such as TF-IDF and Truncated SVD (*Clustering Text Documents Using K-Means*, n.d.).

Gaussian Mixture Models

Gaussian Mixture Models were chosen to determine whether the data underlying the news dataset could be well described using more flexible cluster grouping compared to K-means. Since GMMs calculate probabilities rather than exact values, it may generate interesting insights by capturing new articles that overlap over different news genres, such as business and sports.

Experiments and Discussion

Experimental Setting

Deep Learning

For CNN and LSTM models, training data was split into smaller subsets to account for long training times. This resulted in 108,000 training samples and 12,000 validation samples.

Hyperparameter Tuning Strategy

Initially, a baseline model was built to gauge base performance and determine parameters that could be tuned. Hyperparameter tuning was conducted using Keras Tuner with Random Search. Validation accuracy was used as the primary metric. The search space included variations in embedding dimensions, the number of LSTM units, kernel sizes, dense layer units, dropout rates, and learning rates.

CNN

Figure 1 represents a detailed comparison between the baseline and best CNN model. The baseline CNN model consisted of an embedding layer, two convolutional layers (128 filters each, kernel size 5), max pooling, a dense layer with 64 units, and a dropout layer with a rate of 0.3. The final layer was a softmax classifier for four output classes.

After hyperparameter tuning over 5 trials, the optimal model had an increased embedding dimension (96), more filters in each convolutional layer (192), and a smaller kernel size (3). The dropout and learning rate were held constant. These changes allowed the model to capture more nuanced features and patterns. The dense layer was also increased to 96, establishing a more complex modeling of features.

LSTM

Figure 2 represents a detailed comparison between the baseline and best LSTM model. The baseline LSTM model consisted of an embedding layer, a Bidirectional layer (128 filters each), average pooling, a dense layer with 64 units, dropout layer with a rate of 0.5, and learning rate of 0.001. The final layer was a softmax classifier for four output classes.

Figure 2 highlights the differences between the baseline and best LSTM model. A bidirectional layer was chosen so that the model could capture contextual data in both the past and future, obtaining a more comprehensive understanding of the input sequence compared to a single LSTM layer. To enhance contextual understanding, an additional

bidirectional layer with 96 units was added to the final model. Pooling was changed from average to maximum pooling to emphasise the most important features in each time step. Finally, dropout rate was slightly decreased to 0.4 to simplify the model.

Unsupervised Learning

Prior to implementation, text data was vectorized using Term Frequency–Inverse Document Frequency (TF-IDF) and subsequently reduced in dimensionality using Truncated Singular Value Decomposition (SVD). This transformation compressed the high-dimensional feature space from 100 dimensions down to 2 principal components for both the training and test sets. This dimensionality reduction facilitated faster computation while retaining key semantic structures.

Hyperparameter Tuning Strategy

Both K-means and GMM models were initialised with 4 clusters. This was intentionally done to determine their ability to group news articles based on their content without relying on class labels. Furthermore, two primary metrics were used to evaluate performance: Adjusted Rand Index (ARI) and Silhouette Coefficient. ARI was used to assess similarity between predicted clusters and true labels, with a score closer to 1 indicating high similarity and well-separated clusters. Silhouette Coefficient was used to assess how well clusters were separated by comparing similarities between data points within clusters and between clusters. A score closer to 1 indicated well-separation (Yadav, 2024).

K-means Clustering

For K-means Clustering, Mini Batch K-Means and Grid Search were deployed for hyperparameter tuning. Mini Batch K-means is particularly suitable for handling large-scale, sparse datasets in batches, making it a more computationally viable option compared to traditional K-means.

Hyperparameters chosen for tuning included the number of initialisation runs `n_init` (10 to 50), the maximum number of iterations `max_iter` (100 to 500), and mini batch size `batch_size` (64 to 256). After tuning, the optimal model chose `n_init` = 20, `max_iter` = 100, and `batch_size` = 256.

Gaussian Mixture Models

Similarly, 4 components were chosen for GMM. The hyperparameter `covariance_type` was tuned to determine the constraint type on covariance matrices within each Gaussian component. The options were `full`, `tied`, `diagonal`, and `spherical`, with the most flexible option, `full`, as the optimal parameter.

Experimental Results

Model	Precision	Recall	F1-Score
CNN	0.91	0.91	0.91
LSTM	0.92	0.91	0.91
K-Means	0.61	0.40	0.35
GMM	0.58	0.38	0.34

Weighted average scores for each model

On average, both CNN and LSTM outperformed their unsupervised counterparts, achieving higher F1 scores that were reasonably close to 1. This indicates that, across all classes, these models were effective at correctly classifying news articles whilst minimizing false positives and false negatives. Although LSTM achieved slightly higher precision than CNN, CNN reached similar results in significantly less time, making it a more computationally efficient and effective method for news classification.

K-Means and GMM produced comparable F1 scores, with K-Means performing slightly better. However, as expected, both unsupervised methods underperformed relative to the deep learning models.

Model	Validation Accuracy	Test Accuracy
CNN	0.91	0.90
LSTM	0.91	0.91

As shown above, CNN and LSTM also achieved similar accuracy scores on the validation and test datasets, reinforcing their robustness across data splits. However, both models tended to overfit after a single iteration. Figure 7 demonstrates this by showing the decrease in accuracy and increase in loss on the validation set after one training iteration.

Figures 3 and 4 show the respective confusion matrices. Most news articles were classified correctly, with CNN achieving higher accuracy in the *World* and *Science/Technology* categories, while LSTM performed better in *Sports* and *Business*. Misclassifications tended to occur between *Business* and the *World* or *Science/Technology* categories, suggesting an area for future improvement.

Model	Adjusted Rand Index	Silhouette Coefficient
K-means	0.06	0.40
GMM	0.06	0.30

The results above from highlight the challenge of classifying textual data without access to labels. The Adjusted Rand Index (ARI) was 0.06, indicating that the clustering structure was only marginally better than random. The Silhouette Coefficient ranged between 0.30 and 0.40, suggesting that clusters were moderately defined but had noticeable overlap. The slightly lower silhouette score for GMM may be attributed to its flexible cluster shapes, compared to K-Means, which tends to produce more distinct boundaries.

As seen in Figures 5 and 6, unsupervised methods produced more unstable classification results compared to deep learning approaches. Notably, K-Means failed to identify the *Sports* class altogether, instead misclassifying those articles across the *World*, *Science/Technology*, and *Business* categories. The most accurate predictions were seen in the *Science/Technology* class, though some confusion persisted between *Sports* and *World*. On the other hand, GMM failed to identify *World* news articles and redistributed most of them into the *Sports* category. Interestingly, GMM achieved its best performance in the *Sports* category, though some misclassifications occurred with *Science/Technology*.

Discussion

In hindsight, more robust preprocessing methods could be implemented to enhance predictive accuracy, particularly for the unsupervised methods, which exhibited unstable and suboptimal clustering.

For clustering, the current approach used a TF-IDF vectorizer with `max_features` parameter set to 5000 and with the inclusion of bigram sequences. This approach aimed to strike a balance between computational efficiency and semantic enrichment. However, it was unable to capture nuanced word relationships. The output below demonstrates how the top 20 highest scoring TF-IDF words were dominated by temporal and topical nature such as “new” and day of week terms.

	Word	TF-IDF Score
10	39s	0.021698
2904	new	0.016423
4969	year	0.010352
916	company	0.009746
1641	first	0.009034
4628	two	0.008764
4925	world	0.008759
1769	game	0.008287
3025	oil	0.008026
3043	one	0.007710
2780	monday	0.007428
4841	wednesday	0.007334
4616	tuesday	0.007270
4496	thursday	0.007189
3383	price	0.007130
4844	week	0.007092
4203	state	0.007065
2234	iraq	0.007014
1748	friday	0.006875
2431	last	0.006848

Top 20 words by TF-IDF score

Alternative vectorising approaches include further experimentation of n-grams or alternative word embedding methods such as Word2Vec. Word2Vec is a popular method for vectorising word documents by identifying target words that frequently occur with other words, thus revealing the semantic closeness between words. Pre-trained embeddings such as GloVe and FastText could also be leveraged without needing to train embeddings from scratch. These methods produce dense vectors that store meaningful semantic connections rather than sparse high-dimensional representations such as TF-IDF, especially for clustering algorithms that struggle with high dimensionality.

The deep learning models tested in this study performed well, aided by techniques such as early stopping and the use of ReLU activation. While ReLU is computationally efficient and helps avoid vanishing gradients, it may lead to “dead neurons” when outputs are negative. An alternative such as Leaky ReLU allows a small, non-zero gradient for negative inputs, potentially improving model learning and reducing this risk. Furthermore, the models could improve their generalisability with the inclusion of dropout layers or L1 or L2 regularisation, which adds a penalty to the loss weight function to discourage larger weights.

Another area for improvement lies in the model architectures themselves. While CNN was effective in capturing local patterns and LSTM was better suited to global sequential patterns, a potential enhancement could be the implementation of a hybrid CNN-RNN model. This approach would leverage the strengths of both models and thus strike a balance between capturing local and global sentence structures.

Looking forward, Transformers such as Google’s BERT are novel architectures that have gained traction in recent years. Unlike RNNs, Transformers use self-attention mechanism to relate words in parallel regardless of difference. They are robust to long texts and flexible for a variety of tasks; however, they require more computational resources.

Conclusion

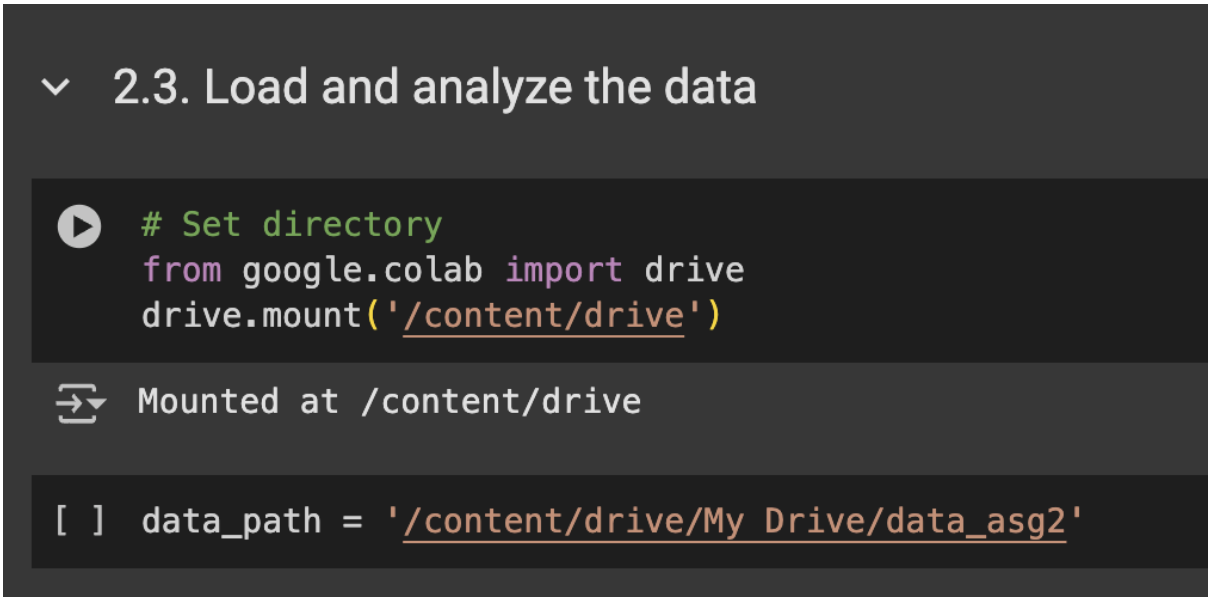
Deep learning approaches such as CNN and LSTM consistently outperformed unsupervised methods in the text classification task, making them the preferred choice for this domain. However, the relatively poor performance of unsupervised models highlights opportunities for improvement — particularly in the integration of more advanced text vectorisation and embedding techniques. Enhancing the quality of input representations could improve clustering outcomes and further elevate the performance of deep learning architectures. This, in turn, may support the development of even more powerful models, such as Transformer-based architectures and CNN-RNN hybrids, which can more effectively capture both local and global patterns in text.

Appendix

All code was run using Google Colab using an Apple M1 Macbook Air Laptop with 8GB. Neural Network models were run using T4 GPU with remaining code run on CPU.

Instructions for running code

1. Include `train.csv` and `test.csv` into Google Drive
2. Proceed running code from section ***2.3 Load and analyse the data***. Please change directory to the location where the dataset has been saved.



(see next page for Appendix 1)

Figure 1: Comparison of Baseline and Best CNN model

Features	Base Model	Best Model
Embedding Dimension	32	96
Conv1D Filters	2 layers, both 128 neurons	2 layers, both 192 neurons
Kernel Sizes	5	3
Dense Layer Units	64	96
Dropout Rate	0.3	0.3
Learning Rate	0.001 (Adam)	0.001

Figure 2: Comparison of Baseline and Best LSTM model

Features	Base Model	Best Model
Embedding Dimension	32	32
LSTM Layer and Units	1 Bidirectional Layer (128)	2 stacked Bidirectional Layers (128, then 96)
Pooling Type	Global Average Pooling	Global Max Pooling
Dense Layer Units	64	64
Dropout Rate	0.5	0.4
Learning Rate	0.001 (Adam)	0.001

Figure 3: Confusion Matrix for CNN

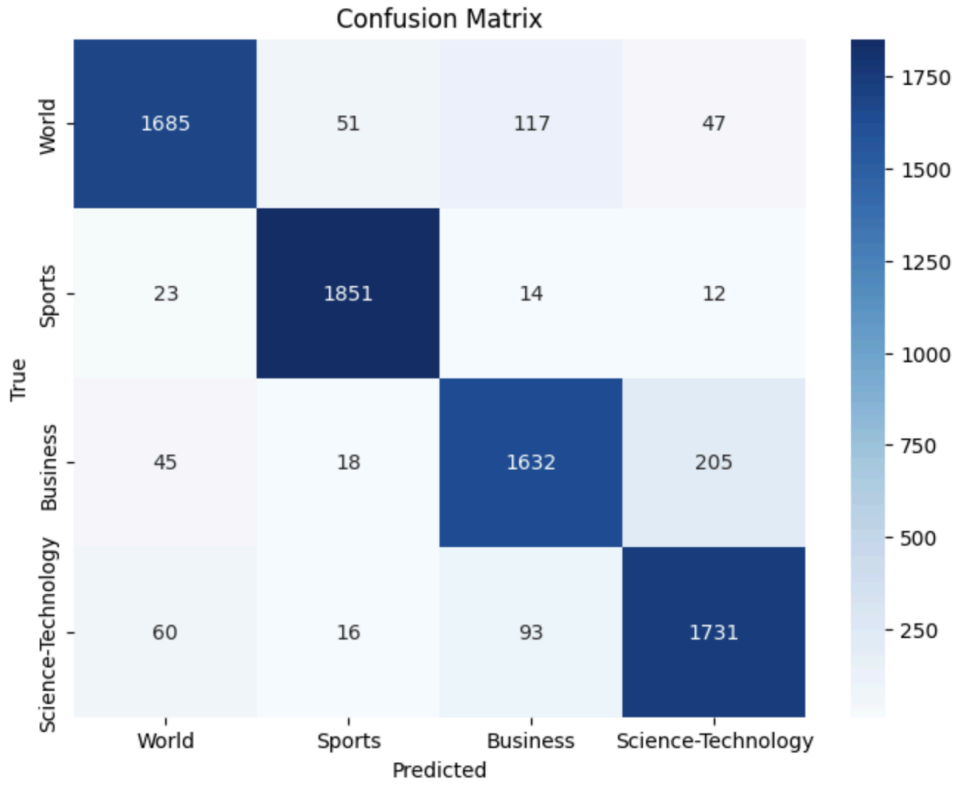


Figure 4: Confusion Matrix for LSTM

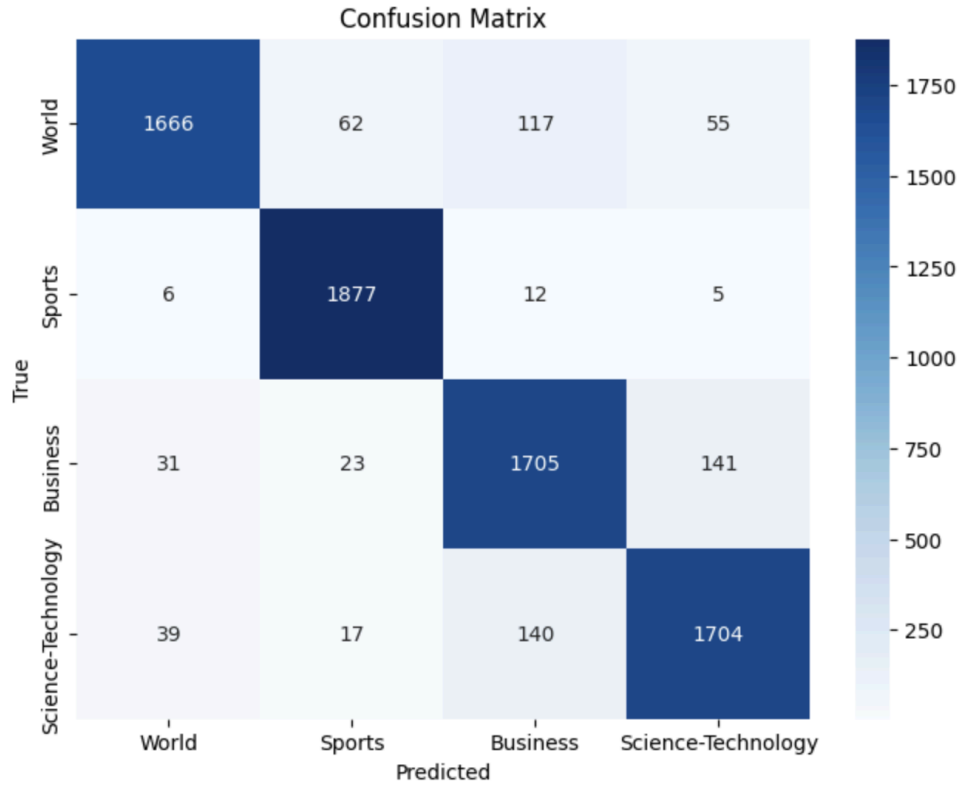


Figure 5: Confusion Matrix for K-means

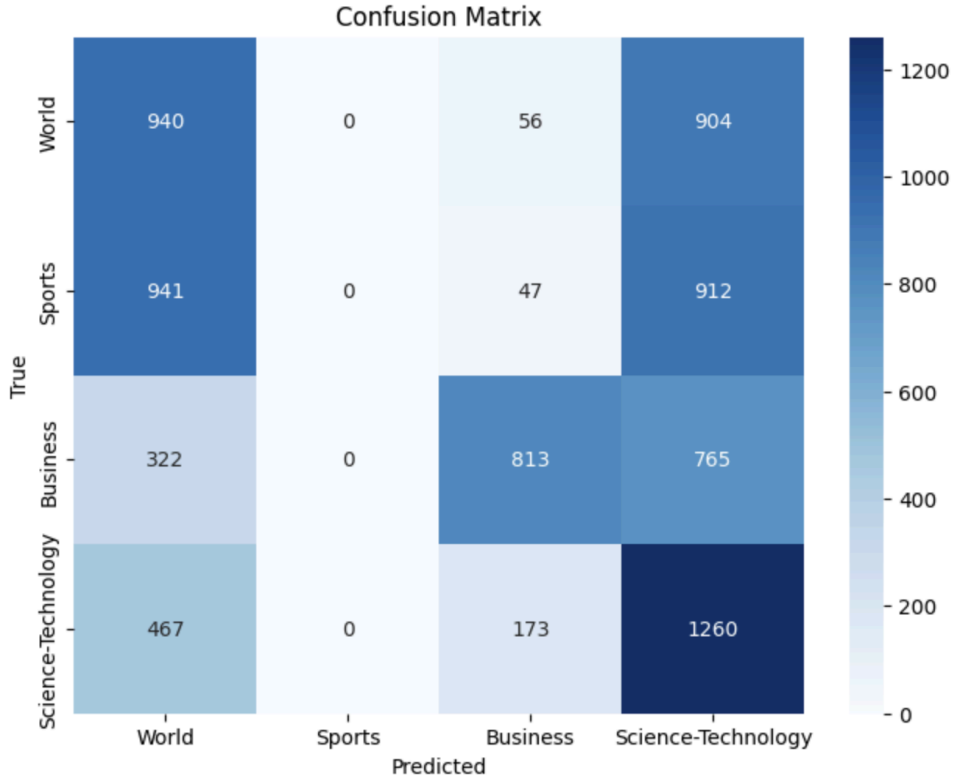


Figure 6: Confusion Matrix for GMM

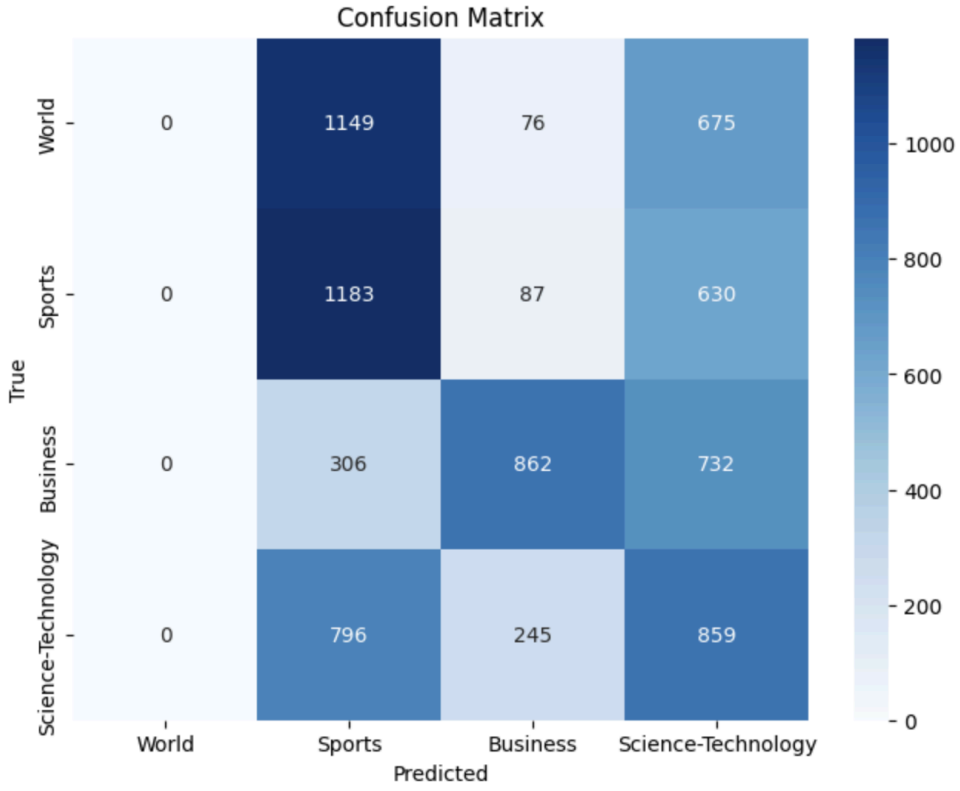
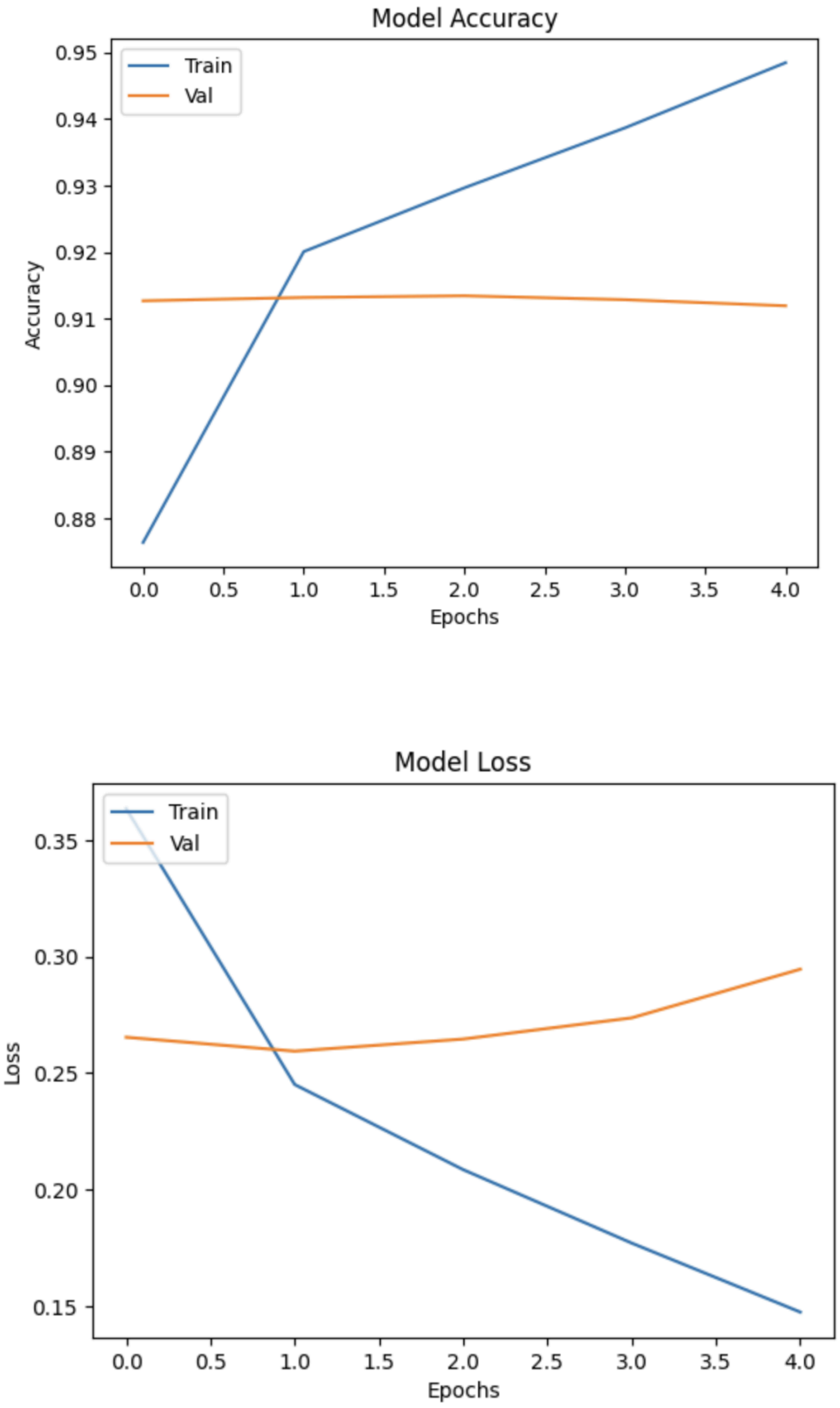


Figure 7: Validation accuracy and loss of CNN and LSTM



References

Zhang, X. (2024, August 28). *ag_news*. Huggingface.co. https://huggingface.co/datasets/fancyzhx/ag_news

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2021). Deep Learning--based Text Classification. *ACM Computing Surveys*, *54*(3), 1–40. https://doi.org/10.1145/3439726

Silva, M. D. (2023, April 30). *Preprocessing Steps for Natural Language Processing (NLP): A Beginner's Guide*. Medium. https://medium.com/@maleeshadesilva21/preprocessing-steps-for-natural-language-processing-nlp-a-beginners-guide-d6d9bf7689c9

Coder, C. (2023, April 10). *Understanding and Implementing TF-IDF in Python - Coldstart Coder - Medium*. Medium. https://medium.com/@coldstart_coder/understanding-and-implementing-tf-idf-in-python-a325d1301484

Khadka, N. (2023, October 11). *Ultimate Guide For Using Truncated SVD For Dimensionality Reduction - Dataaspirant*. https://dataaspirant.com/truncated-svd/

Sharma, N. (2021, November 4). *K-Means Clustering Explained*. Neptune.ai. https://neptune.ai/blog/k-means-clustering

Olamendy, J. C. (2024, January 2). *Understanding Gaussian Mixture Models: A Comprehensive Guide*. Medium. https://medium.com/@juanc.olamendy/understanding-gaussian-mixture-models-a-comprehensive-guide-df30af59ced7

RNN vs. CNN: Understanding Key Differences in Text Classification. (2024). Artiba.org. https://www.artiba.org/blog/rnn-vs-cnn-understanding-key-differences-in-text-classification

Clustering text documents using k-means. (n.d.). Scikit-Learn. https://scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html

Yadav, A. (2024, October 10). *Silhouette Score - Biased-Algorithms - Medium*. Medium; Biased-Algorithms. https://medium.com/biased-algorithms/silhouette-score-d85235e7638b

Jason Brownlee. (2017, October 10). *What Are Word Embeddings for Text?* Machine Learning Mastery. https://machinelearningmastery.com/what-are-word-embeddings/