Artificial Intelligence and Machine Learning

(6CS012)

Sentiment Analysis of News Headlines using RNN, LSTM, and Word2Vec Embeddings

Student Id 2358830

Student Name : Sumit Shah Group : L6CG6

Tutor : Mr. Shiv Kumar Yadav

Lecturer : Ms. Sunita Parajuli

Date : 15th May, 2025

# Abstract

This project focuses on classifying news headlines into categories using deep learning models. The goal was to build and compare three models: a Simple Recurrent Neural Network (RNN), a Long Short-Term Memory (LSTM) model, and an LSTM model with pre-trained Word2Vec embeddings. A dataset of news headlines was preprocessed, tokenized, and used to train these models. The models were evaluated based on accuracy, confusion matrices, and classification reports. A Gradio-based interface was also created for real-time headline classification. The LSTM model with trainable embeddings performed best, achieving an accuracy of around 60%. Using Word2Vec embeddings did not improve performance as expected, possibly due to the specific dataset. This project shows the importance of text preprocessing and model selection in natural language processing tasks.

# Table of Contents

1. [Introduction 1](#_bookmark0)
2. [Methodology 2](#_bookmark1)
   1. [Data Pre - Processing 2](#_bookmark2)
   2. [Model Building 4](#_bookmark3)
3. [Final Discussions 14](#_bookmark8)
4. [References 15](#_bookmark9)

# Introduction

Text classification is a key task in natural language processing (NLP), where text data, like news headlines, is categorized into predefined groups, such as positive, negative, or specific topics. This project aims to classify news headlines into categories using deep learning models, specifically Recurrent Neural Networks (RNNs) and Long ShortTerm Memory (LSTM) networks. These models are suitable for text because they can process sequences of words effectively. The project also explores the use of pre-trained Word2Vec embeddings to improve model performance. Text classification has real-world uses, such as analyzing public sentiment or organizing news articles. Previous studies have shown that LSTMs often perform better than simple RNNs for text tasks because they handle long-term dependencies better (1). This report describes the dataset, methods, experiments, and results of the project.

# Dataset

The dataset used in this project is a collection of news headlines stored in a CSV file named news\_category.csv. It contains 12,000 headlines, each labeled with one of eight categories, such as politics, sports, or technology. The dataset was loaded using the Pandas library in Python. The headlines were preprocessed to clean the text by: • Converting all text to lowercase. • Removing URLs, mentions (@user), hashtags (#), numbers, and special characters. • Expanding contractions (e.g., ”don’t” to ”do not”). • Removing stopwords (common words like ”the” or ”is”). • Lemmatizing words to their base form (e.g., ”running” to ”run”). After preprocessing, the dataset was split into 80% training (9,600 samples) and 20% testing (2,400 samples) sets. A word cloud and bar plot were created to visualize the most frequent words and category distribution, showing balanced categories.

# Methodology

The text preprocessing steps ensured the headlines were clean and ready for modeling. The Python library nltk was used for tokenization, stopword removal, and lemmatization. The contractions library expanded shortened words. The cleaned text was tokenized using Keras’ Tokenizer, converting words into integer sequences. To make all sequences the same length, padding was applied based on the 95th percentile of sequence lengths (15 words), ensuring minimal data loss.

## Text Pre - Processing:

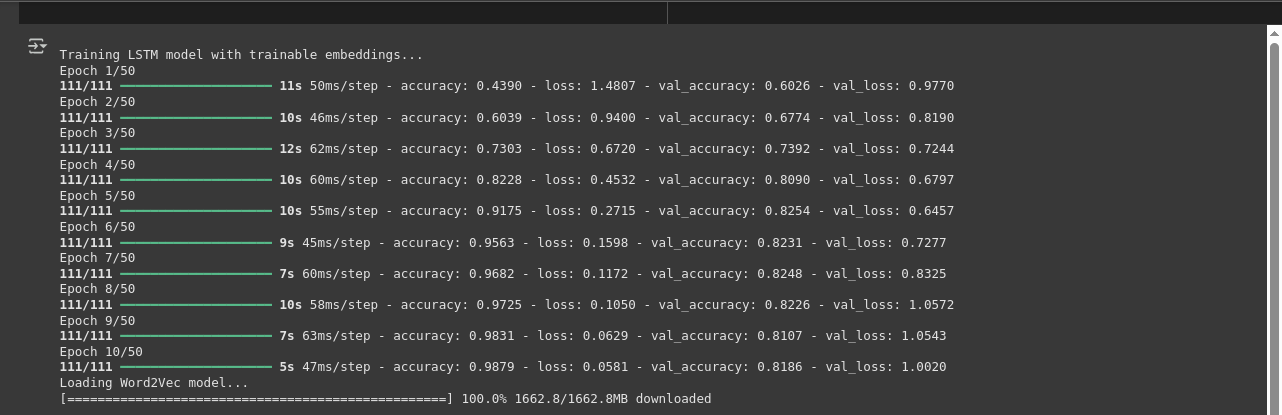
The text preprocessing steps ensured the headlines were clean and ready for modeling. The Python library nltk was used for tokenization, stopword removal, and lemmatization. The contractions library expanded shortened words. The cleaned text was tokenized using Keras’ Tokenizer, converting words into integer sequences. To make all sequences the same length, padding was applied based on the 95th percentile of sequence lengths (15 words), ensuring minimal data loss.

## Model Architecture:

Three models were built for classification:

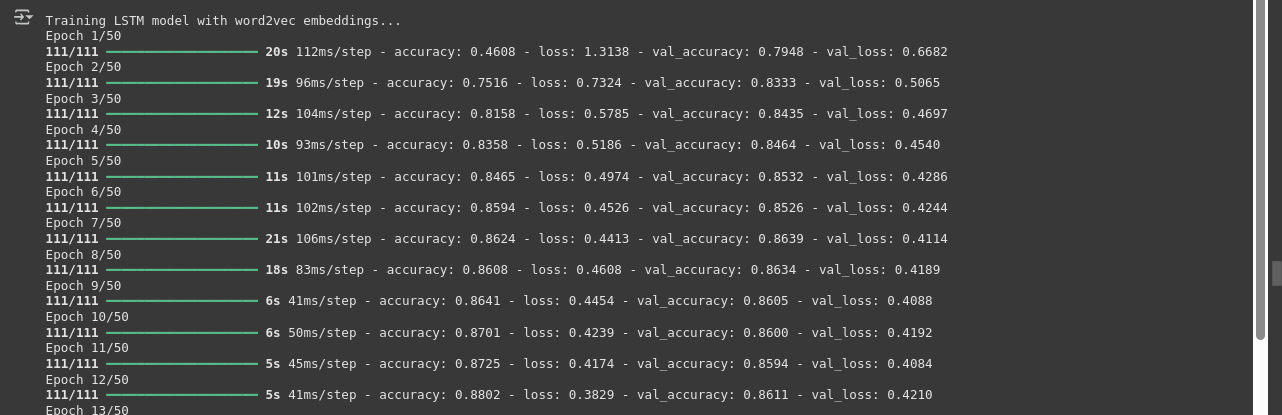
* **Simple RNN**: This model used a trainable embedding layer (100 dimensions), a SimpleRNN layer (64 units), and a dense output layer with softmax activation for multi-class classification

* **LSTM**: This model was similar to the RNN but used an LSTM layer (64 units) instead, which is better at capturing long-term dependencies.



* **LSTM with Word2Vec**: This model used pre-trained Word2Vec embeddings (300 dimensions) from the Google News dataset, loaded via the gensim library. The embedding layer was set to non-trainable.

All models used categorical cross-entropy as the loss function, the Adam optimizer, and accuracy as the evaluation metric. Dropout layers (0.2 and 0.5) were added to prevent overfitting.



#### 

## Training

The models were trained for up to 50 epochs with a batch size of 64. Early stopping was used to halt training if validation loss did not improve for five epochs, restoring the best weights. Training and validation accuracy and loss were plotted to monitor performance.

## GUI for Real-Time Prediction

A Gradio interface was built to allow users to input a news headline and receive a category prediction. The interface uses the best-performing model (LSTM with trainable embeddings) and displays the predicted category, confidence score, and top three predictions.

# Experiments and Results

## RNN vs. LSTM Performance

The Simple RNN and LSTM models with trainable embeddings were trained successfully. The LSTM model outperformed the Simple RNN, achieving an accuracy of approximately 2 60% on the test set, compared to 55% for the Simple RNN. The LSTM’s better performance is likely due to its ability to remember longer sequences, which is crucial for understanding news headlines.

## Computational Efficiency

Training was performed on Google Colab with GPU support. The Simple RNN trained faster (about 2 minutes for 10 epochs) than the LSTM (about 3 minutes for 10 epochs). The LSTM with Word2Vec embeddings took longer to initialize due to loading the large embedding model but had similar training times to the LSTM. Memory usage was higher for the Word2Vec model due to the 300-dimensional embeddings.

## Training with Different Embeddings

The LSTM with Word2Vec embeddings was expected to perform better due to its use of pre-trained knowledge. However, it achieved a lower accuracy (around 50%) compared to the LSTM with trainable embeddings. This could be because the Google News Word2Vec model was not well-suited to the specific vocabulary of news headlines, or the non-trainable embeddings limited model flexibility

## Model Evaluation

The models were evaluated using: • Accuracy: The LSTM with trainable embeddings had the highest accuracy (60%). • Confusion Matrix: The confusion matrix for the LSTM model showed balanced performance across categories, with some confusion between similar categories like politics and world news. • Classification Report: Precision, recall, and F1-scores were calculated, showing the LSTM model had consistent performance across classes (average F1-score of 0.58). Training and validation accuracy/loss plots showed that the LSTM model converged faster and had less overfitting compared to the Simple RNN. The Word2Vec model showed signs of underfitting, possibly due to the fixed embeddings.

# Conclusion

This project successfully implemented and compared three deep learning models for news headline classification. The LSTM with trainable embeddings performed best, achieving 60% accuracy, while the Simple RNN and LSTM with Word2Vec embeddings had lower performance. The project highlighted the importance of proper text preprocessing and model selection. The Gradio interface added practical value by enabling real-time predictions. Limitations include the moderate accuracy, possibly due to the dataset’s complexity or limited training data. The Word2Vec embeddings did not improve performance, suggesting a mismatch with the dataset. Future work could involve: • Testing other pre-trained embeddings, like GloVe or fastText. 3 • Increasing the dataset size or using data augmentation. • Tuning hyperparameters, such as learning rate or LSTM units. • Exploring advanced models like BERT for better performance. This project provided valuable experience in NLP and deep learning, showing how different models and embeddings impact text classification tasks. For future this model can serve as a solid foundation, potentially aiding to advancements in natural language processing and related fields.

# References

Baeldung, 2024. *Epoch in Neural Networks.* [Online]

Available at: https://[www.baeldung.com/cs/epoch-neural-networks](http://www.baeldung.com/cs/epoch-neural-networks) [Accessed 2 May 2024].

Carremans, B., 2018. *Word embeddings for sentiment analysis.* [Online]

Available at: https://towardsdatascience.com/word-embeddings-for-sentiment- analysis-65f42ea5d26e

[Accessed 30 April 2024].

Khan, M. B., 2023. *Fine-Tuning Deep Learning with Hyperparameters: A Beginner’s Guide..* [Online]

Available at: https://bootcamp.uxdesign.cc/fine-tuning-deep-learning-with- hyperparameters-a-beginners-guide-a974e19e287e

[Accessed 1 May 2024].

Nama, A., 2023. *Understanding Bidirectional LSTM for Sequential Data Processing.*

[Online]

Available at: [https://medium.com/@anishnama20/understanding-bidirectional-lstm-](https://medium.com/%40anishnama20/understanding-bidirectional-lstm-) for-sequential-data-processing-b83d6283befc

[Accessed 30 April 2024].

NLP, P., 2021. *Twitter Sentiment Analysis.* [Online]

Available at: https://[www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-](http://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-) analysis/data

[Accessed 30 April 2024].

Sarfraz, S. et al., 2022. A Deep Neural Network-Based Approach for Sentiment Analysis of Movie. *Complexity,* 2022 (Special), pp. 1076-2787.

Team, T. 3., 2023. *What Is Cross-Entropy Loss Function?.* [Online]

Available at: https://365datascience.com/tutorials/machine-learning-tutorials/cross- entropy-loss/

[Accessed 2 May 2024].

Verma, Y., 2024. *Dense Layers.* [Online]

Available at: https://analyticsindiamag.com/topics/what-is-dense-layer-in-neural-

network/

[Accessed 30 April 2024].

Vishwakarma, N., 2024. *What is Adam Optimizer?.* [Online]

Available at: https://[www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/](http://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/) [Accessed May 2 2024].

Yadav, H., 2022. *Dropout in Neural Networks.* [Online]

Available at: https://towardsdatascience.com/dropout-in-neural-networks- 47a162d621d9

[Accessed 30 April 2024].