# Language Modelling with RNN

# AG’s News Topic Classification

Ritesh Kumar

2024SP\_MS\_DSP\_458-DL\_SEC61: Artificial Intelligence and Deep Learning

Module 6

# Third Research/Programming Assignment

# Edward Arroyo and Narayana Darapaneni

# June 15, 2024

Table of Contents

[Abstract 2](#_Toc169104187)

[Introduction 3](#_Toc169104188)

[Literature Review 4](#_Toc169104189)

[Methods 7](#_Toc169104190)

[Results 1 11](#_Toc169104191)

[Results 2 15](#_Toc169104192)

[Conclusion 18](#_Toc169104193)

[References 22](#_Toc169104194)

Abstract

In this assignment, we explore alternative network topologies to improve text classification performance using the AG’s news topic classification dataset. Our primary objective is to evaluate the effectiveness of different neural network structures: fully connected (dense) networks, recurrent neural networks (RNNs) with and without long short-term memory (LSTM), and one-dimensional convolutional neural networks (1D CNNs). By comparing these models, we aim to determine their respective strengths in handling text sequences and their computational efficiency.

The motivation for choosing each topology stems from their distinct capabilities. Dense networks, serving as a baseline, are straightforward but often struggle with sequential data. RNNs, particularly those with LSTM units, are designed to process sequences, making them well-suited for text data that has inherent temporal dependencies. 1D CNNs, although primarily used for image data, can capture local patterns in text sequences through their convolutional operations.

This analysis informs our preprocessing steps, such as vocabulary size adjustments and sequence length truncation, ensuring a robust foundation for model training.

Our experiments include:

1. Evaluating a fully connected (dense) network as a baseline.
2. Testing RNNs with variations in architecture, bidirectional/unidirectional layers, and regularization techniques.
3. Experimenting with LSTM-based networks to leverage their memory capabilities for better handling of sequential data.
4. Implementing 1D CNNs to explore their efficacy in capturing local textual patterns.

Each model is trained with consistent parameters (e.g., document truncation to 128 tokens, batch size of 100, 10 epochs, same optimizer, and cross-entropy loss function) to ensure fair comparisons. We measure the models' predictive accuracy and processing requirements, presenting results in a comparative table highlighting accuracy, loss for train/validation/test sets, and training time.

This comprehensive analysis allows us to identify the most effective network topology for text classification, considering both performance and computational efficiency.

Introduction

The goal of this assignment is to explore and compare various neural network topologies for text classification, focusing on the AG’s news topic classification dataset. Text classification is a crucial natural language processing (NLP) task that involves assigning predefined categories to textual data. Given the sequential nature of text, choosing the appropriate network architecture significantly impacts the model’s performance and efficiency.

We begin by establishing a baseline using fully connected (dense) networks, which, despite their simplicity, often struggle with capturing temporal dependencies in text. To address this limitation, we investigate recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, which are designed to handle sequences effectively by maintaining contextual information over time. Additionally, we examine one-dimensional convolutional neural networks (1D CNNs), which can detect local patterns within text sequences through convolutional operations.

Our approach includes conducting extensive exploratory data analysis (EDA) to understand the dataset's structure and ensure thorough preprocessing. By comparing these models under consistent conditions, we aim to identify the most effective network topology for text classification, considering both predictive accuracy and computational efficiency. This investigation provides insights into the strengths and limitations of different neural network architectures in handling sequential text data.

Literature Review

The exploration of various neural network topologies for text classification tasks has garnered significant attention in the research community. The fundamental challenge in text classification lies in effectively capturing the contextual information embedded within sequences of words. Researchers have experimented with different neural network architectures, such as fully connected (dense) networks, recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and convolutional neural networks (CNNs), to address this challenge.

Fully Connected (Dense) Networks: Dense networks have been traditionally employed for text classification tasks due to their simplicity and ease of implementation. However, they often fail to capture sequential dependencies inherent in textual data. Early works, such as the application of dense networks in the classification of Reuters-21578 news articles, demonstrated the limitations of this approach in handling the temporal structure of text (Sebastiani, 2002).

Recurrent Neural Networks (RNNs) and LSTMs: RNNs and LSTMs have been extensively researched for their ability to model sequential data. Mikolov et al. (2010) highlighted the potential of RNNs in language modeling, demonstrating their ability to predict word sequences. However, traditional RNNs suffer from vanishing gradient problems, which LSTMs address through gated mechanisms that manage long-term dependencies. Research by Graves et al. (2013) showcased the efficacy of LSTMs in handwriting recognition, emphasizing their capability in sequential prediction tasks. In text classification, Zhou et al. (2018) applied LSTMs to sentiment analysis, illustrating improved performance over traditional RNNs and dense networks.

Convolutional Neural Networks (CNNs): CNNs, originally designed for image processing tasks, have been adapted for text classification due to their proficiency in capturing local features. Kim (2014) demonstrated the effectiveness of CNNs in sentence classification, showing that convolutional operations can extract salient features from word embeddings. Subsequent research by Zhang et al. (2015) explored character-level CNNs for text classification, indicating that CNNs can effectively handle textual data without explicit word boundaries.

Comparative Studies: Several comparative studies have investigated the performance of these neural network architectures on text classification tasks. For instance, Conneau et al. (2017) conducted a comprehensive comparison of CNNs and RNNs on large-scale text classification, concluding that both architectures have unique strengths. Their findings suggest that CNNs excel in capturing local dependencies, while RNNs and LSTMs are superior in modeling long-range dependencies.

Hybrid Approaches: In recent years, hybrid models that combine the strengths of CNNs and RNNs/LSTMs have been proposed. Yoon Kim (2014) introduced a hybrid architecture that leverages convolutional layers to capture local features and LSTM layers to model sequential dependencies. Such models have shown promising results in various text classification benchmarks, including sentiment analysis and news categorization.

Motivation and Contributions: This assignment builds on the existing body of research by systematically comparing dense networks, RNNs, LSTMs, and CNNs for the AG’s news topic classification dataset. The motivation is to understand the trade-offs between different architectures in terms of predictive accuracy and computational efficiency. By conducting extensive exploratory data analysis (EDA) and standardizing experimental conditions, this study aims to provide insights into the most effective neural network topology for text classification, considering both performance and processing requirements.

In summary, the research landscape on text classification has evolved from simple dense networks to more sophisticated architectures like RNNs, LSTMs, and CNNs. Comparative studies and hybrid approaches have further enriched our understanding of these models. This assignment contributes to this ongoing research by empirically evaluating the performance of these neural network topologies on a real-world text classification task.

Methods

Research Design and Modeling Methods: The primary objective of this research is to evaluate and compare various neural network topologies for text classification, focusing on the AG’s news topic classification dataset. The research design includes the following steps:

1. Baseline Model: Implement a fully connected (dense) network to establish a performance baseline.
2. Advanced Architectures: Explore more complex architectures such as Recurrent Neural Networks (RNNs) with and without Long Short-Term Memory (LSTM) units, and One-Dimensional Convolutional Neural Networks (1D CNNs).
3. Comparative Analysis: Compare the performance of these models in terms of accuracy, loss, and computational efficiency.
4. Consistent Evaluation: Maintain consistent experimental conditions across models to ensure a fair comparison. This includes using the same data preprocessing steps, batch sizes, number of epochs, optimizer, and loss function.

Implementation and Programming: The implementation is conducted using Python, leveraging TensorFlow and Keras libraries for model building, training, and evaluation. The key components include:

1. Model Definitions: Define the neural network architectures for the dense network, RNN, LSTM, and 1D CNN. Each model is implemented with variations in architecture (e.g., bidirectional/unidirectional layers) and hyperparameters (e.g., regularization techniques).
2. Activation Functions: The `ReLU` activation function is used in hidden layers for its efficiency in training deep networks, while `softmax` is used in the output layer for multi-class classification.
3. Loss Function: The `SparseCategoricalCrossentropy` loss function is employed, which is suitable for multi-class classification problems with integer labels.
4. Optimizer: The `RMSprop` optimizer is chosen for its adaptive learning rate capabilities, which help in faster convergence.
5. Regularization: Techniques such as dropout are applied to mitigate overfitting. Dropout rates of 0.5 are used in most configurations to ensure a good balance between retaining and dropping units.

Data Preparation, Exploration, and Visualization

1. Data Preparation: The AG’s news topic classification dataset, available through TensorFlow Datasets, is used for this study. Data preparation steps include:

* Loading the Dataset: Load the dataset and split it into training, validation, and test sets.
* Text Vectorization: Implement text vectorization using Keras's `TextVectorization` layer. This involves tokenizing the text, removing stop words, and converting text to sequences of integers.
* Padding/Truncation: Ensure that all sequences have a uniform length by padding shorter sequences and truncating longer ones. For this study, sequences are truncated to 128 tokens.

1. Exploratory Data Analysis (EDA): Conduct extensive EDA to understand the dataset’s structure and characteristics:

* Class Distribution: Analyze the distribution of documents across different news categories.
* Vocabulary Size: Determine the appropriate vocabulary size by experimenting with different levels (e.g., 5000, 10000, 20000 words).
* Frequency Analysis: Examine the frequency of words and identify the most common terms.

1. Visualization: Visualization techniques are employed to gain insights into the data and model performance:

* Data Distribution Plots: Visualize the distribution of documents across categories and the frequency of words in the corpus.
* Training Progress: Plot training and validation accuracy/loss over epochs to monitor model performance and diagnose potential overfitting or underfitting issues.
* Comparative Performance: Generate comparative plots to visualize the performance differences between the various models in terms of accuracy and loss.

Implementation Steps

1. Baseline Dense Network: Implement and train a fully connected network to serve as the baseline.
2. RNN and LSTM Models: Implement RNN and LSTM models with various configurations (e.g., bidirectional layers) and train them.
3. 1D CNN Model: Implement a 1D CNN model and train it on the same data.
4. Evaluation and Comparison: Evaluate each model on the test set and compare their performance using accuracy, loss, and processing time metrics.

Programming and Execution: The entire process is programmed in Jupyter Notebooks, enabling iterative development and visualization. TensorFlow and Keras are used for model implementation, while Pandas and Matplotlib are used for data handling and visualization.

**Experiment Details**

**Experiment A:** Here, we focus on critical preprocessing steps for NLP projects, aimed at optimizing the data before defining and training the models. These steps include:

1. Vocabulary Size Adjustment: We test the impact of different vocabulary sizes on model performance by using three levels of vocabulary sizes: 5000, 10000, and 20000 words. This helps in understanding the balance between vocabulary comprehensiveness and computational efficiency.
2. Editing the Vocabulary: We compare the performance of models using unedited vocabularies, which include the most frequent words as they appear in the dataset, against edited vocabularies, where common words such as articles ('the', 'a') are removed using a custom stopwords function. This step is designed to reduce noise and enhance the significance of meaningful terms.
3. Output Sequence Length: We evaluate the impact of sequence length on model performance by using the default sequence length versus setting a fixed output sequence length of 100 tokens. This helps in determining the optimal sequence length for the given task.

The combinations of these preprocessing steps form specific experiments, such as:

* Unedited vocabulary with different vocabulary sizes and sequence lengths.
* Edited vocabulary with different vocabulary sizes and sequence lengths.
* Fixed sequence length with unedited and edited vocabularies.

These preprocessing strategies aim to refine data quality and improve model performance by systematically altering the key parameters involved in text processing.

A graph of different colored lines

Description automatically generated A graph of different vocabulary sizes

Description automatically generated A graph of different colored lines

Description automatically generated A graph of different colored lines

Description automatically generated A graph of different vocabulary sizes

Description automatically generated A graph of different colored lines

Description automatically generated A graph of different vocabulary sizes

Description automatically generated A graph of different vocabulary sizes

Description automatically generated

These figures illustrate the performance of models with varying vocabulary sizes, both unedited and edited, across different sequence lengths. Each plot shows the model accuracy and loss over epochs for the training and validation sets. The goal is to evaluate the impact of vocabulary size, vocabulary editing, and sequence length on model performance.

1. Unedited Vocabulary

* Vocabulary Sizes: 5000, 10000, 20000 words.
* Models with larger vocabularies (20000 words) tend to achieve higher training and validation accuracies, indicating that a more comprehensive vocabulary helps in capturing more information.
* Loss decreases more rapidly for models with larger vocabularies, suggesting better convergence and learning.

2. Edited Vocabulary

* Vocabulary Sizes: 5000, 10000, 20000 words.
* Models with edited vocabularies also show improved performance with larger vocabulary sizes, although the improvement is less pronounced compared to unedited vocabularies.
* Editing the vocabulary to remove common stopwords seems to stabilize the validation accuracy and loss, reducing overfitting.

3. Unedited Fixed Length Vocabulary

* Models with fixed sequence lengths show similar trends to those with default sequence lengths, where larger vocabularies lead to higher accuracies and lower losses.
* Fixed length sequences help in standardizing the input, which can benefit model training.

4. Edited Fixed Length Vocabulary

* The trends are consistent with those observed in the unedited fixed length vocabulary experiments.
* Edited vocabularies with fixed lengths tend to reduce overfitting, as seen by the more stable validation accuracy and loss curves.

Overall, the experiments reveal that:

* Larger Vocabularies: Tend to improve model performance by capturing more linguistic features.
* Vocabulary Editing: Helps in reducing overfitting and stabilizes model performance, especially noticeable in validation metrics.
* Fixed Sequence Lengths: Provide a standardized input format, aiding in consistent model training and evaluation.

By systematically adjusting vocabulary sizes, editing vocabularies, and setting sequence lengths, the experiments highlight the importance of these preprocessing steps in optimizing model performance for text classification tasks.

**Experiments B:** The following experiments focus on various Recurrent Neural Network architectures and their variations to evaluate their effectiveness in text classification tasks:

1. Simple RNN: Implement a basic RNN architecture with a single recurrent layer. This experiment aims to establish a baseline performance using `ReLU` activation functions and applying dropout for regularization to mitigate overfitting. The goal is to understand how a straightforward RNN handles sequential data.
2. Bidirectional Simple RNN: Extend the simple RNN model to a bidirectional architecture. This approach captures dependencies in both forward and backward directions, providing a more comprehensive understanding of the sequential context within the text. The bidirectional setup is expected to enhance the model's ability to learn from the data.
3. Regularized Bidirectional LSTM: Introduce LSTM units in a bidirectional setup, incorporating regularization techniques such as dropout. LSTM units are designed to handle long-term dependencies more effectively than simple RNNs, and the bidirectional configuration further improves this capability. Regularization helps prevent overfitting, ensuring the model generalizes well to new data.
4. GRU with Regularization: Explore Gated Recurrent Units (GRUs) as an alternative to LSTMs. GRUs are known for their simpler architecture and faster training times while still capturing complex dependencies in the data. This experiment applies similar regularization strategies, such as dropout, to assess the performance and efficiency of GRUs compared to LSTMs.

**Experiments C:** In these experiments, we explore various configurations of LSTM networks to evaluate their performance in text classification tasks. The focus is on understanding how different architectural choices and regularization techniques impact model accuracy and generalization.

1. Basic LSTM: Implement a standard LSTM model with a single LSTM layer. This experiment serves to establish a baseline for LSTM performance, providing insights into how well a basic LSTM network can capture and learn from sequential data in comparison to simpler RNNs.
2. Bidirectional LSTM: Enhance the basic LSTM model by incorporating bidirectional layers. This architecture allows the network to capture dependencies in both forward and backward directions, providing a more comprehensive understanding of the sequential patterns within the text. The bidirectional LSTM is expected to outperform the unidirectional version by leveraging context from both ends of the sequence.
3. Stacked LSTM: Introduce multiple LSTM layers to create a deeper network architecture. This stacked configuration aims to explore the benefits of depth in neural networks, allowing the model to learn more complex representations of the data. By increasing the number of LSTM layers, we can investigate how deeper architectures influence learning capacity and performance.
4. Regularized LSTM: Apply dropout and other regularization techniques to the LSTM layers to mitigate overfitting. Regularization helps the model generalize better to unseen data by preventing it from relying too heavily on any single feature or pattern. This experiment tests the effectiveness of different regularization strategies, such as varying dropout rates, to enhance the robustness and reliability of the LSTM network.

**Experiments D:** One-Dimensional Convolutional Neural Networks (1D CNNs)

In these experiments, we implement and test various configurations of 1D CNNs to understand their effectiveness in text classification tasks. The focus is on how different architectural choices and regularization techniques impact the model's ability to extract meaningful features from the text.

1. Basic 1D CNN: Start with a simple 1D CNN architecture to establish its baseline performance. This basic model includes a single convolutional layer followed by pooling and fully connected layers. The aim is to understand how well a straightforward 1D CNN can handle text data and extract relevant features for classification.
2. 1D CNN with More Filters: Increase the number of convolutional filters in the basic 1D CNN architecture. This experiment examines the impact of having more filters on the model’s ability to extract finer and more detailed features from the text. By testing various filter counts, we can determine the optimal number of filters for improved performance.
3. Additional Conv1D Layer: Add more convolutional layers to the initial 1D CNN architecture. Introducing additional Conv1D layers allows the model to capture more complex and hierarchical patterns in the data. This experiment aims to explore the benefits of deeper convolutional networks in extracting and combining multi-level features from the text.
4. Regularized 1D CNN: Incorporate dropout and other regularization techniques into the CNN model to evaluate their effectiveness in preventing overfitting. By applying dropout between layers, the model’s ability to generalize to unseen data is enhanced, reducing the likelihood of overfitting. This experiment tests different regularization strategies to find the most effective method for maintaining robust performance across diverse datasets.

A graph of different colored lines

Description automatically generated

The plots present the performance of different 1D CNN architectures over training epochs, comparing training and validation accuracy and loss. The models tested include Simple 1D CNN, 1D CNN with more filters, Additional Conv1D layers, and Regularized 1D CNN.

Training and Validation Accuracy

* Simple 1D CNN: Shows steady improvement in both training and validation accuracy, reaching around 0.86, indicating consistent learning.
* More Filters 1D CNN: Demonstrates higher initial accuracy compared to the simple 1D CNN, stabilizing at about 0.87. This suggests that more filters enhance feature extraction, improving performance.
* Additional Conv1D Layer: Displays a rapid increase in training accuracy but a slower rise in validation accuracy. This indicates potential overfitting as the model becomes more complex.
* Regularized 1D CNN: Maintains a balance between training and validation accuracy, with both improving steadily and aligning closely, suggesting effective regularization in preventing overfitting.

Training and Validation Loss

* Simple 1D CNN: Shows a consistent decrease in both training and validation loss, indicating effective learning and generalization.
* More Filters 1D CNN: Exhibits lower initial loss and continues to decline steadily, reflecting improved learning due to enhanced feature extraction capabilities.
* Additional Conv1D Layer: Experiences a significant initial drop in training loss, but validation loss shows slight fluctuations, again indicating potential overfitting.
* Regularized 1D CNN: Displays a steady decline in both training and validation loss, with the validation loss closely tracking the training loss. This indicates that regularization techniques are effective in maintaining generalization.

Summary

* More Filters 1D CNN: Provides the best performance in terms of accuracy, leveraging enhanced feature extraction to improve classification.
* Regularized 1D CNN: Balances accuracy and loss well, preventing overfitting and maintaining generalization.
* Additional Conv1D Layer: May overfit, suggested by the divergence in training and validation accuracy and loss.
* Simple 1D CNN: Offers consistent learning and serves as a reliable baseline.

These results highlight the importance of model complexity and regularization in achieving optimal performance in text classification tasks using 1D CNN architectures.

By systematically comparing various neural network topologies under consistent experimental conditions, this research aims to identify the most effective model for text classification tasks. The goal is to strike an optimal balance between predictive accuracy and computational efficiency. Through rigorous experimentation and analysis, we seek to determine which network architectures provide the best performance while maintaining manageable computational costs. This comprehensive evaluation includes considering factors such as training time, model complexity, and the ability to generalize well to unseen data. Ultimately, this research will provide valuable insights into selecting the most appropriate neural network topology for practical text classification applications.

Reference:

1. Debole, Franca & Sebastiani, Fabrizio. (2004). An Analysis of the Relative Difficulty of Reuters-21578 Subsets. ([Link](https://www.researchgate.net/publication/228825188_An_Analysis_of_the_Relative_Difficulty_of_Reuters-21578_Subsets))
2. Zhou, Kai & Long, Fei. (2018). Sentiment Analysis of Text Based on CNN and Bi-directional LSTM Model. 1-5. 10.23919/IConAC.2018.8749069. ([Link](https://www.researchgate.net/publication/334244376_Sentiment_Analysis_of_Text_Based_on_CNN_and_Bi-directional_LSTM_Model))
3. Kim, Yoon. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 10.3115/v1/D14-1181. ([Link](https://www.researchgate.net/publication/265052545_Convolutional_Neural_Networks_for_Sentence_Classification))
4. Yin, Wenpeng & Kann, Katharina & Yu, Mo & Schütze, Hinrich. (2017). Comparative Study of CNN and RNN for Natural Language Processing. ([Link](https://www.researchgate.net/publication/313443663_Comparative_Study_of_CNN_and_RNN_for_Natural_Language_Processing))
5. Shin, Joongbo & Kim, Yanghoon & Yoon, Seunghyun & Jung, Kyomin. (2018). Contextual-CNN: A Novel Architecture Capturing Unified Meaning for Sentence Classification. 491-494. 10.1109/BigComp.2018.00079. ([Link](https://www.researchgate.net/publication/325423879_Contextual-CNN_A_Novel_Architecture_Capturing_Unified_Meaning_for_Sentence_Classification/citation/download))