Multi-Class Text Classification: A Comparison of Word Representations and ML/NN Models

Juboraj Ahmed
Department of Computer Science and Engineering
BRAC University
Course: CSE440 - Natural Language Processing II Lab
Dhaka, Bangladesh
Email: juboraj.ahmed@g.bracu.ac.bd

***Abstract*—This project explores the performance of various machine learning and neural network architectures for multi-class text classification. We compare four distinct word representation techniques—Bag of Words (BoW), TF-IDF, GloVe embeddings, and Skip-gram (Word2Vec)—across a range of models including Logistic Regression, Naive Bayes, Random Forest, and several variants of Recurrent Neural Networks (RNN). A total of 22 experiments were conducted to evaluate the interaction between feature extraction methods and model complexity. Results are measured using Accuracy and Macro F1-score to assess the robustness of each approach. This report details the full pipeline from exploratory data analysis and preprocessing to model evaluation and final deployment using Flask.**

***Index Terms*—NLP, Text Classification, Word Embeddings, LSTM, Machine Learning, Lab Report.**

I. Introduction

Text classification is a fundamental task in Natural Language Processing (NLP) with applications ranging from intent detection to topic categorization. As part of the CSE440 Lab project, this work investigates how different numerical representations of text influence the performance of both shallow and deep learning models. While classical machine learning (ML) models rely heavily on frequency-based features like TF-IDF, modern neural networks (NN) leverage dense vector representations to capture semantic relationships.

The goal of this project is to perform a comparative study using the Question Answer Classification Dataset to identify the optimal combination of word representation and architecture for this specific task. We also demonstrate a production-ready extension by deploying the best-performing model using the Flask framework.

II. Dataset Description

The dataset utilized is the *Question Answer Classification Dataset*. It is provided in a structured format consisting of:

- 5 Training CSV files
- 1 Test CSV file

Each file contains two primary columns: `text` (the question snippet) and `label` (the target category). The training files were merged into a single training set of [INSERT TOTAL TRAIN SIZE] samples, while the test file provided [INSERT TEST SIZE] samples for final evaluation.

III. Exploratory Data Analysis (EDA)

Initial data exploration was conducted to understand the distribution and quality of the text data. Key findings include:

- **Class Distribution**: The labels were found to be [Balanced/Imbalanced], with certain categories like [INSERT CATEGORY] appearing more frequently than others.
- **Text Length**: The average length of questions was approximately [INSERT AVG LENGTH] words.
- **Data Quality**: No significant missing values were found, but [INSERT COUNT] duplicate entries were identified and removed during the cleaning phase.

IV. Methodology

A. Preprocessing

To convert raw text into a clean format suitable for vectorization, the following steps were applied:

1) **Lowercasing**: All characters were converted to lowercase to reduce vocabulary variance.
2) **Punctuation & Number Removal**: Non-alphabetic characters were stripped using regular expressions.
3) **Stopword Removal**: Common English words (e.g., "the", "is") were removed using the NLTK library.
4) **Lemmatization**: Words were reduced to their dictionary root (lemma) to group different inflections of the same word.

B. Word Representations

We implemented four types of representation:

- **BoW (Bag of Words)**: Simple counts of word occurrences.
- **TF-IDF**: Weighting words based on their frequency in a document vs. the entire corpus.
- **GloVe**: Pre-trained dense vectors capturing global word co-occurrence statistics.
- **Skip-gram (Word2Vec)**: Locally trained embeddings using the gensim library to predict context words.

C. ML and NN Model Architectures

We evaluated three ML models: Logistic Regression (LR), Naive Bayes (NB), and Random Forest (RF). For Deep Learning, we implemented seven architectures:

- **DNN**: A standard feed-forward network with Global Average Pooling.
- **SimpleRNN, GRU, LSTM**: Standard recurrent layers.
- **Bidirectional SimpleRNN/GRU/LSTM**: Layers that process sequences in both directions to capture forward and backward context.

D. *Hyperparameter Tuning Strategy*

Hyperparameters were tuned manually through iterative experimentation. For ML models, we adjusted regularization strengths (C in LR) and tree depths in RF. For NN models, we focused on:

- **Batch Size**: [INSERT VALUE] (chosen for memory efficiency).
- **Epochs**: [INSERT VALUE] (monitored via Early Stopping).
- **Dropout**: [INSERT VALUE] (used to mitigate overfitting).
- **Hidden Units**: [INSERT VALUE] (balanced for complexity vs. speed).

V. Experimental Results

We conducted exactly 22 experiments as per the project requirements. The results are summarized in Table I.

A. *Discussion of Observations*

- **Best ML Model**: [INSERT BEST ML] achieved the highest metrics among classical models due to [Insert Reason, e.g., handling sparse features well].
- **Worst ML Model**: [INSERT WORST ML] performed poorly likely because [Insert Reason].
- **Best NN Model**: [INSERT BEST NN] outperformed others, likely benefiting from bidirectional long-term dependency captures.
- **Worst NN Model**: [INSERT WORST NN] showed [Insert Observation, e.g., vanishing gradient issues].
- **Comparing ML vs NN**: The best NN model provided a [INSERT GAIN]% improvement over the best ML model, justifying the use of deep architectures for this semantic task.

VI. Deployment

As a production-ready extension, the best-performing model was integrated into a Flask web application. The application provides a REST endpoint and a user-friendly interface where users can input text and receive a class prediction in real-time. This emphasizes the scalability of the developed pipeline.

VII. Conclusion

This lab project successfully compared several NLP techniques for multi-class classification. We found that [INSERT MAIN TAKEAWAY, e.g., GloVe + Bidirectional LSTM] provided the most stable performance. The results indicate that while simple frequency-based models are fast, they lack the depth required for complex sentence structures.

VIII. Limitations

- The model training was limited by [e.g., local CPU/GPU compute].
- Pre-trained GloVe embeddings may not cover specialized jargon in the dataset.
- Manual hyperparameter tuning might not have reached the absolute global optimum.

IX. Future Work

Future iterations could explore Transformer-based models such as BERT or RoBERTa for state-of-the-art performance. Implementing an automated search (using Keras Tuner) for hyperparameters would also be beneficial for optimization.

References

[1] T. Mikolov, et al. “Efficient Estimation of Word Representations in Vector Space,” arXiv:1301.3781, 2013.

[2] J. Pennington, et al. “GloVe: Global Vectors for Word Representation,” EMNLP, 2014.

[3] S. Bird, et al. “Natural Language Processing with Python,” O’Reilly Media, 2009.

[4] F. Chollet. “Keras,” https://keras.io, 2015.

TABLE I
SUMMARY OF 22 EXPERIMENTS ACROSS REPRESENTATIONS AND MODELS

Exp ID	**Representation**	**Model**	**Accuracy (%)**	**Macro F1**	**Notes**
1	BoW	Logistic Regression	[PLACEHOLDER]	[PLACEHOLDER]	Statistical Baseline
2	BoW	Naive Bayes	[PLACEHOLDER]	[PLACEHOLDER]	
3	BoW	Random Forest	[PLACEHOLDER]	[PLACEHOLDER]	
4	BoW	DNN	[PLACEHOLDER]	[PLACEHOLDER]	
5	TF-IDF	Logistic Regression	[PLACEHOLDER]	[PLACEHOLDER]	
6	TF-IDF	Naive Bayes	[PLACEHOLDER]	[PLACEHOLDER]	
7	TF-IDF	Random Forest	[PLACEHOLDER]	[PLACEHOLDER]	
8	TF-IDF	DNN	[PLACEHOLDER]	[PLACEHOLDER]	
9	GloVe	SimpleRNN	[PLACEHOLDER]	[PLACEHOLDER]	Unidirectional
10	GloVe	GRU	[PLACEHOLDER]	[PLACEHOLDER]	
11	GloVe	LSTM	[PLACEHOLDER]	[PLACEHOLDER]	
12	GloVe	Bidir-SimpleRNN	[PLACEHOLDER]	[PLACEHOLDER]	Bidirectional Context
13	GloVe	Bidir-GRU	[PLACEHOLDER]	[PLACEHOLDER]	
14	GloVe	Bidir-LSTM	[PLACEHOLDER]	[PLACEHOLDER]	
15	GloVe	DNN	[PLACEHOLDER]	[PLACEHOLDER]	
16	Skip-gram	SimpleRNN	[PLACEHOLDER]	[PLACEHOLDER]	Trained on Current Dataset
17	Skip-gram	GRU	[PLACEHOLDER]	[PLACEHOLDER]	
18	Skip-gram	LSTM	[PLACEHOLDER]	[PLACEHOLDER]	
19	Skip-gram	Bidir-SimpleRNN	[PLACEHOLDER]	[PLACEHOLDER]	
20	Skip-gram	Bidir-GRU	[PLACEHOLDER]	[PLACEHOLDER]	
21	Skip-gram	Bidir-LSTM	[PLACEHOLDER]	[PLACEHOLDER]	
22	Skip-gram	DNN	[PLACEHOLDER]	[PLACEHOLDER]	