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

Ishraq Kamal Adib
Dept. of
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
ishraq.kamal.adib@g.bracu.ac.bd

Mostakim Morshed
Dept. of
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
mostakim.morshed@g.bracu.ac.bd

Stanley Matthew Das
Dept. of
Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
stanley.matthew.das@g.bracu.ac.bd

***Abstract*—This project applies a comprehensive experimental pipeline to a multi-class question-answer topic classification dataset. We performed exploratory data analysis (EDA), applied standard text preprocessing, implemented four word representations (Bag-of-Words, TF-IDF, GloVe, Skip-gram), and trained a suite of models: three classic machine learning classifiers (Logistic Regression, Naive Bayes, Random Forest) and seven neural network architectures (DNN, SimpleRNN, GRU, LSTM, and their bidirectional variants). We trained the ML models on BoW,TF-IDF; all NN architectures on GloVe,Skip-gram with and DNN on all four. The best ML model found was Logistic Regression with TF-IDF and the best NN model was a Bidirectional GRU with Skip-gram embeddings.**

***Index Terms*—text classification, TF-IDF, BoW, GloVe, Skip-gram, RNN, LSTM, GRU, word embeddings**

I. Introduction

This project addresses multi-class classification of short question titles into topical labels (e.g. Science & Mathematics, Education & Reference, Politics & Government, Entertainment & Music, Sports, etc.). The goal is to compare classical feature-based classifiers with neural sequence models across different word representations and to document reproducible experimental choices and outcomes. The provided dataset was already split into an 80% training set and 20% test set; 10% of the training set was further used for validation of the training data during hyperparameter tuning and early stopping.

II. Methodology

A. Dataset & Exploratory Data Analysis

We began by examining the dataset to understand its structure and distribution before designing preprocessing and modeling strategies. The dataset contains short question titles annotated with multiple topical categories. To characterize the data, we performed exploratory data analysis (EDA) and observed the following:

- **Word Cloud:** Frequently occurring words include "question," "answer," "content," and "title," along with high-frequency verbs such as "say," "know," and "want." This suggests the dataset contains both domain-specific and conversational vocabulary.
- **Class Distribution:** Classes are relatively balanced, with no extreme skew, although certain categories such as *Education & Reference* and *Science & Mathematics* are slightly more dominant.
- **Text Lengths:** Most samples are very short, typically below 20 tokens, reinforcing the short-text nature of the task.
- **Missing Values:** The dataset has no missing values across relevant fields (QA text, class labels, length/count features).

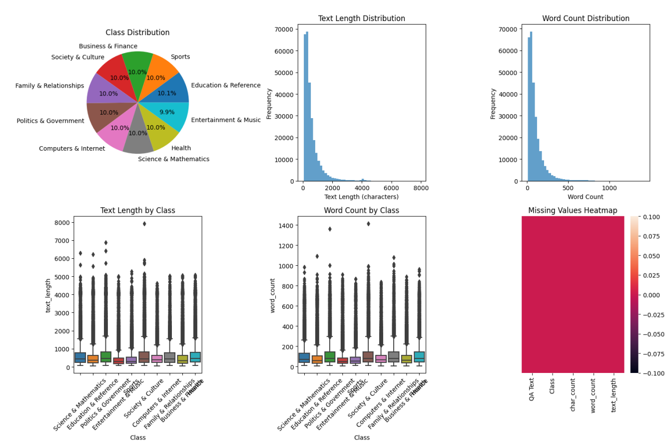

Fig. 1. EDA visualizations: (a) class distribution, (b) text length distribution, (c) word count distribution, (d) text length by class, (e) word count by class, and (f) missing values heatmap.

These EDA results provided the basis for preprocessing decisions. For example, we retained topical nouns and key tokens while removing URLs and user handles, since they were identified as noise. Furthermore, the short length of most titles suggested that recurrent architectures would not require deep stacking to capture dependencies effectively.

B. Preprocessing

The preprocessing pipeline implemented in the notebook includes:

- Lowercasing, trimming whitespace.
- punctuation stripping except where meaningful.
- Tokenization , stopword removal , and lemmatization

Fig. 2. Word cloud of the training dataset showing most frequent tokens. Common topical words and conversational markers dominate

- For embedding experiments, an explicit OOV embedding initialization was used for words.
- After visualizing wordcloud of the above figure we preprocessed some unnecessary words which were very frequent

C. Word Representations

We used four representations required by the project:

- Bag-of-Words (BoW): scikit-learn CountVectorizer (unigram; optional n-grams in experiments), tuned min_df and max_features.
- TF-IDF: scikit-learn TfidfVectorizer with sublinear_tf, tuned ngram_range and max_df/min_df.
- GloVe: pretrained GloVe embeddings loaded and mapped to the dataset vocabulary.
- Skip-gram: skip-gram embeddings trained on the training corpus (gensim/Keras implementation) and used to initialize the embedding layer for sequence models.

D. Model Architectures

Machine Learning Models:

- **Logistic Regression:** A linear model used with both Bag-of-Words (BoW) and TF-IDF representations to classify text by finding a linear decision boundary.
  - **Hyperparameters:** `max_iter=5000`
- **Naive Bayes (MultinomialNB):** A probabilistic model suitable for text classification with discrete features (like word counts in BoW or TF-IDF). It's based on applying Bayes' theorem with the "naive" assumption of conditional independence between features.
  - **Hyperparameters:** Default parameters were used.
- **Random Forest:** An ensemble learning method that builds multiple decision trees and merges their predictions to improve accuracy and control overfitting. It was applied with both BoW and TF-IDF features.
  - **Hyperparameters:** `n_estimators=100`, `random_state=42`

Deep Learning Models:

- **Deep Neural Network (DNN):** A feedforward neural network with multiple layers using learning rate of 0.001. Used with BoW, TF-IDF, and averaged word embeddings (Skip-gram and GloVe) to learn non-linear relationships in the data.
  - **Hyperparameters:**
    - ∗ Hidden Layers: 2
    - ∗ Neurons per layer: 256 (first), 128 (second)
    - ∗ Activation function: ReLU
    - ∗ Dropout: 0.5 (after first), 0.3 (after second)
    - ∗ Output layer activation: Softmax
    - ∗ Optimizer: Adam
    - ∗ Loss function: Sparse Categorical Crossentropy
    - ∗ Epochs: 10
    - ∗ Batch Size: 32
- **Recurrent Neural Networks (RNNs):** Used with pre-trained GloVe and Skip-gram embeddings (embedding dimension 100), with embedding layer `trainable=False`.
  - **Hyperparameters (Common to SimpleRNN, GRU, and LSTM):**
    - ∗ RNN layer units: 128
    - ∗ Dropout: 0.2 (within RNN layer), 0.3 (after RNN layer), 0.2 (after first Dense)
    - ∗ Recurrent Dropout: 0.2
    - ∗ Dense layer neurons: 128
    - ∗ Activation: ReLU (Dense layers)
    - ∗ Output activation: Softmax
    - ∗ Optimizer: Adam
    - ∗ Loss: Sparse Categorical Crossentropy
    - ∗ Epochs: 5
    - ∗ Batch Size: 256
  - **Bidirectional RNNs (BiRNNs):** SimpleRNN, GRU, or LSTM layer wrapped in a Bidirectional layer.
    - ∗ **Hyperparameters:** Same as unidirectional counterparts, with Bidirectional wrapper.

E. Hyperparameter Tuning and Validation Protocol

Manual tuning guided by validation (held-out from training) prioritized weighted F1 as the model selection metric. We changed number of epochs to tune.

III. Results

This section reports the final test set metrics extracted from the notebook outputs and provides analysis and comparison.

A. Quantitative Results

Best models:

a) Best ML (vector models): Logistic Regression (TF-IDF) — Test Acc 0.6290, F1 0.6248.:

b) Best NN (sequence models): Bidirectional GRU (Skip-gram) — Test Acc 0.7111, F1 0.7047.: Worst models:

c) Worst ML (vector models): Random Forest (BoW) — Test Acc 0.5956, F1 0.5910.:

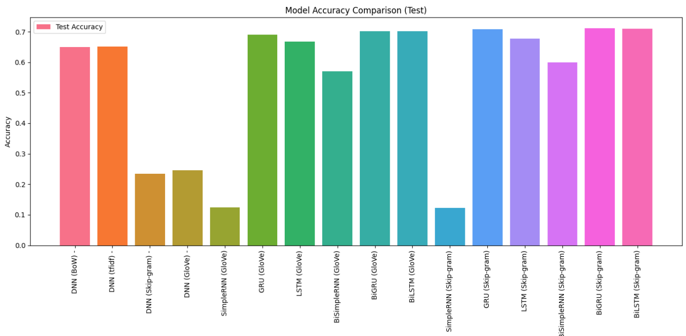

Fig. 3. Accuracy Comparison of All 22 Combinations

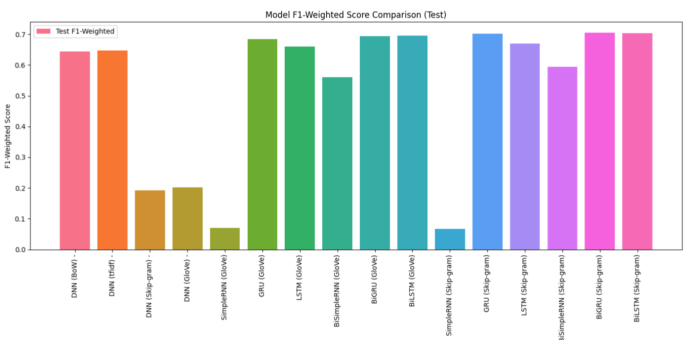

Fig. 4. F1 Score Comparison of All 22 Combinations

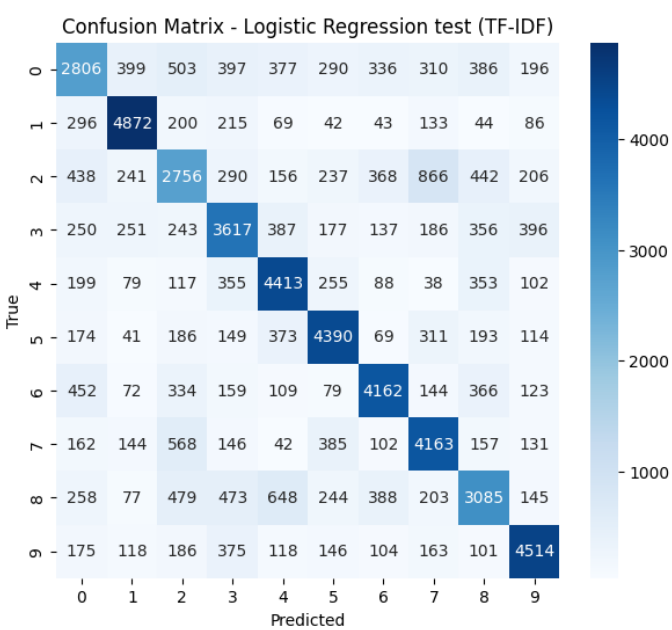

Fig. 5. Example of a figure caption.

TABLE I
MODEL PERFORMANCE: TEST ACCURACY

Model	Representation	Epochs	Test Accuracy
Logistic Regression	BoW	5000	0.6290
Logistic Regression	TF-IDF	5000	0.6474
Naive Bayes	BoW	N/A	0.6279
Naive Bayes	TF-IDF	N/A	0.6275
Random Forest	BoW	N/A	0.5956
Random Forest	TF-IDF	N/A	0.5990
DNN	BoW	10	0.6499
DNN	TF-IDF	10	0.6513
DNN	Skip-gram	10	0.2342
DNN	GloVe	10	0.2462
SimpleRNN	GloVe	5	0.1244
GRU	GloVe	5	0.6912
LSTM	GloVe	5	0.6680
BiSimpleRNN	GloVe	5	0.5711
BiGRU	GloVe	5	0.7014
BiLSTM	GloVe	5	0.7021
SimpleRNN	Skip-gram	5	0.1224
GRU	Skip-gram	5	0.7080
LSTM	Skip-gram	5	0.6773
BiSimpleRNN	Skip-gram	5	0.6001
BiGRU	Skip-gram	5	0.7111
BiLSTM	Skip-gram	5	0.7097

TABLE II
MODEL PERFORMANCE: TEST F1-SCORE (MACRO)

Model	Representation	Epochs	Test F1 (Macro)
Logistic Regression	BoW	5000	0.6247
Logistic Regression	TF-IDF	5000	0.6451
Naive Bayes	BoW	N/A	0.6245
Naive Bayes	TF-IDF	N/A	0.6244
Random Forest	BoW	N/A	0.5910
Random Forest	TF-IDF	N/A	0.5940
DNN	BoW	10	0.6445
DNN	TF-IDF	10	0.6465
DNN	Skip-gram	10	0.1925
DNN	GloVe	10	0.2015
SimpleRNN	GloVe	5	0.0703
GRU	GloVe	5	0.6841
LSTM	GloVe	5	0.6594
BiSimpleRNN	GloVe	5	0.5603
BiGRU	GloVe	5	0.6946
BiLSTM	GloVe	5	0.6948
SimpleRNN	Skip-gram	5	0.0660
GRU	Skip-gram	5	0.7022
LSTM	Skip-gram	5	0.6694
BiSimpleRNN	Skip-gram	5	0.5945
BiGRU	Skip-gram	5	0.7047
BiLSTM	Skip-gram	5	0.7037

d) Worst NN (sequence models): Simple RNN for both combination — Test Acc 0.1244, F1 0.703.:

B. Comparative Analysis

a) Representation impact: BoW/TF-IDF + linear models provide robust baselines (≈0.62–0.65). Sequence models with GloVe or Skip-gram embeddings outperform these baselines when architectures are properly configured — particularly BiGRU/BiLSTM.:

b) Architecture impact: GRU and LSTM (especially bidirectional) consistently beat SimpleRNN, indicating vanishing

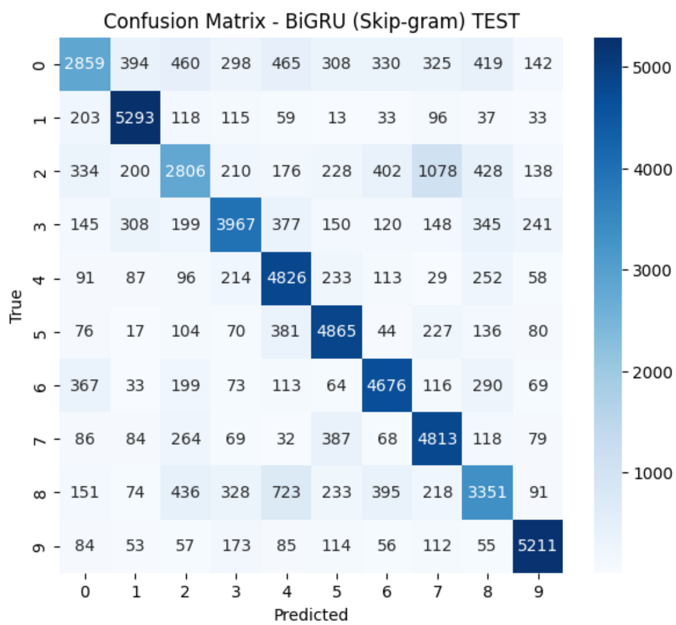

Fig. 6. Example of a figure caption.

gradient mitigation and better memory of longer context help, even for short titles.:

c) Stability issues: Some pairings (e.g., DNN with Skip-gram, SimpleRNN with embeddings) produced very low accuracies. Likely causes include poor embedding initialization, unsuitable learning rates, or architecture-representation mismatch.:

C. Confusion Matrices & Per-class Performance

Full confusion matrices and classification reports for top models are included in the notebook outputs. They show that most confusion occurs between topically adjacent classes (e.g., politics vs society). If desired, we can embed the notebook's confusion matrix images into the final PDF.

IV. Conclusion

We executed the full required experiment matrix and identified clear trends: for this short-text topic classification task, bidirectional recurrent models with Skip-gram embeddings yielded the best performance (BiGRU: Acc = 0.7111). TF-IDF with Logistic Regression give strong, fast baselines (Acc $\approx$ 0.6474). Compared to other NN models, DNN model shows disappointing results.

Limitations: Training NN models on such large datasets are computationally expensive on small hardware.

Future work: Future works can evaluate transformer-based encoders (BERT) and compare compute/accuracy tradeoffs.

References

[1] Seaborn Seaborn: statistical data visualization. Version 0.13.2. Online documentation. Available: seaborn.pydata.org seaborn.pydata.org

[2] Pandas pandas documentation. Version 2.3.1, July 07, 2025. Online documentation. Available: pandas.pydata.org Pandas

[3] NLTK (Natural Language Toolkit) Natural Language Toolkit. Online platform. Available: nltk.org nltk.org

[4] wordcloud wordcloud · PyPI (Python Package Index). Latest version documentation. Available: pypi.org

[5] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.