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

Nahid Hassan
*Dept of*
*Computer Science and Engineering*
*BRAC University*
Dhaka, Bangladesh
nahid.hassan1@g.bracu.ac.bd

Alvee Ishraque
*Dept of*
*Computer Science and Engineering*
*BRAC University*
Dhaka, Bangladesh
alvee.ishraque@g.bracu.ac.bd

Tarek Alam Bhuiyan
*Dept of*
*Computer Science and Engineering*
*BRAC University*
Dhaka, Bangladesh
tarek.alam.bhuiyan@g.bracu.ac.bd

***Abstract*—This paper compares various word representation techniques (BoW, TF-IDF, GloVe, and Skip-gram) and machine learning (ML) and neural network (NN) models for multi-class text classification. The study evaluates ML models such as Naive Bayes, Logistic Regression, Random Forest, and neural architectures like DNN, RNN, GRU, LSTM, and their bidirectional variants. Using a dataset of 340,000 question-answer pairs across ten categories, results show that Logistic Regression with TF-IDF achieved the highest performance among ML models (accuracy = 70.3%, macro F1 = 0.704). However, Bidirectional LSTM with Skip-gram embeddings outperformed all other models, achieving the best accuracy (72.5%) and macro F1 (0.73), highlighting the advantages of incorporating bidirectional context and capturing long-range dependencies in text classification tasks.**

## I. Introduction

Text classification is a fundamental task in Natural Language Processing (NLP) with applications in sentiment analysis, spam detection, document categorization, and question answering. In the multi-class setting, the challenge is to assign each text to one of many categories, making the task more complex than binary classification and highly relevant for real-world systems.

The motivation for this project is to evaluate how different word representation methods and modeling approaches affect multi-class classification performance. Traditional representations such as Bag of Words (BoW) and TF-IDF capture surface-level lexical features, while distributed embeddings such as GloVe and Skip-gram encode semantic relationships. Similarly, classical Machine Learning (ML) models like Logistic Regression, Naive Bayes, and Random Forest are lightweight and effective on sparse features, whereas Neural Network (NN) architectures such as Deep Neural Networks (DNN), RNNs, GRUs, and LSTMs are designed to capture sequential and contextual information. A comparative study enables us to identify the strengths and weaknesses of each approach.

The dataset used in this project consists of approximately 340,000 question–answer texts evenly distributed across ten categories, including domains such as *Science & Mathematics*, *Education & Reference*, *Computers & Internet*, *Business & Finance*, *Entertainment & Music*, *Sports*, *Politics & Government*, etc. It is provided with an 80–20 train–test split, ensuring fair evaluation. The balanced distribution across classes makes it suitable for assessing the relative performance of ML and NN models under different word representation schemes.

## II. Methodology

### A. Exploratory Data Analysis (EDA)

To understand the dataset, we first conducted an exploratory data analysis (EDA). This included analyzing the class distribution, identifying missing values, and visualizing text lengths. We performed basic statistics on the dataset and plotted histograms to observe the data's characteristics. The dataset consists of 339,998 question–answer texts evenly distributed across ten categories, with about 28,000 samples per class, confirming a balanced distribution without the need for resampling. Text length analysis showed variation across categories, ranging from roughly 437 tokens in *Entertainment & Music* to 669 tokens in *Politics & Government*. A word cloud highlighted frequent artifacts such as question," title," and "best answer," which were removed during preprocessing. These findings informed the design of the preprocessing pipeline and model configurations.

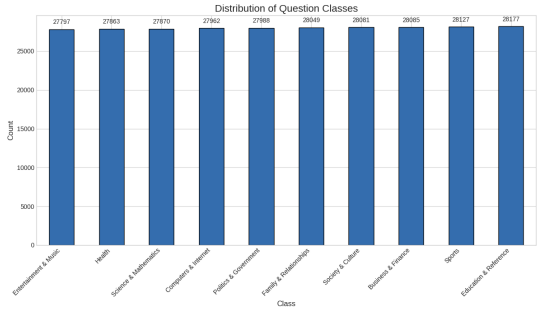


Fig. 1: Class Distribution of the Dataset

### B. Preprocessing

The raw texts contained boilerplate tokens (e.g., "question title," "best answer"), URLs, emails, numbers, and other artifacts that were removed or replaced with placeholders. Contractions were expanded while preserving negation, and

common stopwords were filtered, excluding negation words. All text was lowercased, tokenized, and lemmatized to normalize word forms. Special placeholders were used for dates, years, percentages, monetary values, and generic numbers. Duplicates were removed to avoid data leakage. For ML models (BoW, TF-IDF, DNN), cleaned strings were generated, while for embedding-based neural networks, tokenized sequences were prepared with padding and truncation as required.

### C. Word Representation

The following word representation methods were used in this project:

- **Bag of Words (BoW)**: Represents text as sparse vectors based on raw word frequencies without capturing semantic meaning.
- **TF-IDF**: Weights terms by their frequency in a document relative to their occurrence in the entire corpus, emphasizing informative words while down-weighting common ones.
- **GloVe**: Pre-trained dense embeddings that capture semantic relationships by learning from global word co-occurrence statistics.
- **Skip-gram**: Contextual embeddings learned by predicting surrounding words from a target word, providing richer semantic and syntactic information.

### D. Model Architectures

We implemented both traditional machine learning classifiers and neural network models for multi-class text classification:

**Machine Learning Models:**

**Logistic Regression:** A linear model optimized for multi-class classification using softmax and cross-entropy loss.

$$\hat{y} = \frac{e^{z_k}}{\sum_{i=1}^{C} e^{z_i}}$$

where $z_k$ is the score for class $k$ and $C$ is the number of classes.

**Naive Bayes:** A probabilistic classifier based on conditional independence assumptions, efficient with sparse features.

$$P(C|X) = \frac{P(C)\prod_{i=1}^{n} P(x_i|C)}{P(X)}$$

where $P(C)$ is the class prior, $P(x_i|C)$ is the likelihood for feature $x_i$, and $P(X)$ is the evidence.

**Random Forest:** An ensemble of decision trees trained on random feature subsets, reducing variance and improving robustness.

$$f(x) = \frac{1}{T}\sum_{t=1}^{T} f_t(x)$$

where $f_t(x)$ is the prediction from the $t$-th tree, and $T$ is the number of trees.

**Neural Network Models:**

**Deep Neural Network (DNN):** Fully connected feedforward layers applied on BoW/TF-IDF representations.

$$y = \sigma(Wx + b)$$

where $x$ is the input, $W$ is the weight matrix, $b$ is the bias, and $\sigma$ is the activation function.

**Simple RNN:** Sequential model capturing temporal dependencies in text.

$$h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

where $h_t$ is the hidden state at time $t$, $x_t$ is the input, and $U$ is the recurrent weight.

**GRU:** Gated recurrent unit architecture designed to mitigate vanishing gradient problems.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \quad z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

$$\tilde{h_t} = \tanh(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h_t}$$

where $r_t$ is the reset gate, $z_t$ is the update gate, and $\tilde{h_t}$ is the candidate hidden state.

**LSTM:** Long Short-Term Memory network capable of modeling long-range dependencies.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \quad i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \quad \tilde{C_t} = \tanh(W_C x_t + U_C h_{t-1} + b_C)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C_t}, \quad h_t = o_t \circ \tanh(C_t)$$

where $f_t$, $i_t$, $o_t$ are the forget, input, and output gates, and $\tilde{C_t}$ is the candidate cell state.

**Bidirectional RNN:** Processes sequences in both forward and backward directions to capture context from both sides.

$$h_t = \sigma(Wx_t + Uh_{t-1} + Vh_{t+1} + b)$$

where $h_{t+1}$ is the hidden state at the next time step, capturing information from the future as well.

**Bidirectional GRU:** Extension of GRU with bidirectional processing for richer sequence modeling.

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h_t} + h_{t+1}$$

**Bidirectional LSTM:** Extension of LSTM that captures past and future dependencies simultaneously.

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h_t}$$

### E. Validation Protocol and Hyperparameter Tuning

*BoW/TF–IDF vectorizers.:* For TF–IDF we limited `max_features` to 50 000 to curb the curse of dimensionality and reduce overfitting and memory pressure; including n-grams up to bigrams (`ngram_range`=$(1, 2)$) captures short phrases (e.g., "high school", "credit card") that are informative for topical labels without exploding feature space as trigrams would. `sublinear_tf` dampens the influence of extremely frequent terms, improving linear separability. A minimum document frequency (`min_df`=5) removes idiosyncratic tokens that act as noise, stabilizing Naive Bayes likelihoods and Logistic Regression coefficients.

*Naive Bayes ($\alpha$).:* Laplace/Lidstone smoothing $\alpha$ controls how strongly the model backs off from zero counts. In sparse text, rare tokens appear spuriously and zero probabilities cause brittle posteriors; increasing $\alpha$ improves robustness but can over-smooth and blur genuine class-specific terms. We therefore varied $\alpha$ over small values (e.g., $0.5, 1.0, 1.5$) to find the point that *maximizes macro F1* by balancing noise suppression against preservation of discriminative features.

*Logistic Regression ($C$, solver, penalty).:* In very high-dimensional sparse spaces, the main risk is *variance*, not bias. The inverse regularization strength $C$ directly trades variance (large $C$) for stability (small $C$). We swept $C$ across orders of magnitude to find the smallest value that retains accuracy while preventing coefficient blow-up. L2 regularization was preferred for TF–IDF because it distributes weight across correlated features better than L1, and the `saga` solver is designed for large, sparse problems with efficient convergence. We optionally used `class_weight=balanced` to guard against subtle frequency differences between classes and to keep the decision boundary fair in macro F1.

*Random Forest (SVD dimensionality, depth, trees).:* Tree ensembles perform best when features are dense, low-to-moderate dimensional, and contain non-linear interactions. Raw BoW/TF–IDF violates these assumptions: extremely sparse and very high-dimensional. We therefore reduced dimensionality with truncated SVD to a compact latent space (e.g., 256/512) to make axis-aligned splits meaningful and to reduce variance. We tuned tree `max_depth` to limit overfitting on the compressed space and `n_estimators` to average variance without excessive training time. This targets the bias–variance sweet spot that forests need to be competitive on text.

*Feed-forward DNN (width, depth, dropout, learning rate).:* On sparse inputs, MLPs can overfit quickly because each hidden unit sees thousands of largely independent features. We varied hidden widths (e.g., 512/256) to set representational capacity, applied dropout (0.3–0.5) to decorrelate co-adapted units, and tuned the Adam learning rate (around $10^{-3}$ to $5\times10^{-4}$) to stabilize optimization without "chasing noise." Early stopping halts training at the onset of validation overfitting, which is common with BoW/TF–IDF.

*Embeddings + RNNs (hidden size, dropout, bidirectionality, sequence length).:* For sequence models, capacity is controlled mainly by hidden size and whether the model is bidirectional. Larger hidden states capture longer-range dependencies but increase memory and overfitting risk; we paired moderate hidden sizes with dropout (including recurrent dropout where supported) to regularize temporal dynamics. Bidirectional variants were explored because category cues often appear *before and after* entity mentions or discourse markers; looking both ways typically sharpens boundaries (as later confirmed in confusion matrices). Sequence length and batch size were bounded by RAM; truncation was set to retain the most informative spans while keeping training stable. For embeddings, we allowed fine-tuning so the model could adapt GloVe/Skip-gram vectors to dataset-specific usage but relied on early stopping to prevent catastrophic drift.

*Why these ranges (practical constraints).:* We avoided excessively wide grids to stay within memory and time budgets which otherwise caused crashes. The chosen ranges are the ones most likely to change the *effective capacity* or the *regularization profile* of each model class; they therefore deliver the largest impact on macro F1 per unit of search effort.

## III. Results

The performance of each model was evaluated using accuracy, F1-score, and confusion matrices. The results were compared across different word representation techniques to identify the most effective approach for text classification.

TABLE I: Performance of ML Models and DNN with Bag of Words

| Model | Accuracy | Macro F1 |
|---|---|---|
| Naive Bayes | 0.688 | 0.684 |
| Logistic Regression | 0.653 | 0.649 |
| Random Forest | 0.532 | 0.533 |
| Deep Neural Network | 0.647 | 0.645 |

The results of Bag of Words (BoW) representations with different models are summarized in Table I. Among the machine learning classifiers, Naive Bayes achieved the best performance with an accuracy of 0.688 and a macro F1-score of 0.684. Logistic Regression and the Deep Neural Network (DNN) obtained comparable results, with F1-scores around 0.65, demonstrating that linear models and simple feedforward architectures can handle sparse BoW features reasonably well. Random Forest, however, performed poorly with only 0.532 accuracy and 0.533 macro F1, indicating that ensemble tree-based methods are less effective when applied to high-dimensional sparse text vectors. Overall, these results suggest that Naive Bayes is the most effective choice for BoW features in this dataset, while more complex models such as Random Forest and DNN do not provide improvements over simpler baselines. This establishes a clear performance baseline for comparison with TF-IDF and embedding-based representations in the subsequent experiments.

TABLE II: Performance of ML and DNN Models with TF-IDF Representation

| Model | Accuracy | Macro F1 | Weighted F1 |
|---|---|---|---|
| Naive Bayes | 0.691 | 0.688 | 0.688 |
| Logistic Regression | 0.703 | 0.704 | 0.703 |
| Random Forest | 0.593 | 0.589 | 0.589 |
| Deep Neural Network | 0.672 | 0.664 | 0.665 |

With TF-IDF features, Logistic Regression achieved the strongest performance (Accuracy = 0.703, Macro F1 = 0.704, Weighted F1 = 0.703), outperforming Naive Bayes (F1 = 0.688) and Random Forest (F1 = 0.589). The Random Forest model consistently underperformed, confirming its weakness in sparse, high-dimensional representations. The DNN achieved an F1 score of 0.664 (weighted F1 = 0.665), showing moderate gains over Naive Bayes but still lagging behind Logistic Regression. Overall, TF-IDF improved performance across models compared to Bag of Words, highlighting its ability to capture informative word weights.

TABLE III: Performance of GloVe-based Neural Models

| Model | Test Acc. | Test Loss | Macro Prec. | Macro Recall | Macro F1 |
|---|---|---|---|---|---|
| Bidirectional LSTM | 0.7148 | 0.8905 | 0.7098 | 0.7148 | 0.7093 |
| Bidirectional GRU | 0.7122 | 0.8958 | 0.7067 | 0.7122 | 0.7063 |
| LSTM | 0.7101 | 0.9138 | 0.7067 | 0.7101 | 0.7043 |
| GRU | 0.7082 | 0.9116 | 0.7062 | 0.7082 | 0.7020 |
| DNN (GloVe) | 0.6728 | 1.0118 | 0.6662 | 0.6782 | 0.6657 |
| Bi-SimpleRNN | 0.6413 | 1.1427 | 0.6416 | 0.6413 | 0.6328 |
| SimpleRNN | 0.5945 | 1.2386 | 0.6075 | 0.5945 | 0.5725 |

Across all GloVe-based models, the Bidirectional LSTM achieved the strongest performance, with a test accuracy of 71.5% and macro F1 of 0.7093, slightly outperforming Bidirectional GRU and unidirectional LSTM/GRU. These results highlight the advantage of bidirectional sequence modeling, as information from both past and future tokens improves classification of complex question–answer texts. The DNN baseline with pooled embeddings achieved only 66.6% macro F1, showing the importance of temporal modeling over static representations. SimpleRNN variants performed the worst due to their inability to capture long-range dependencies. Overall, the results demonstrate that gated recurrent architectures with pre-trained embeddings are very effective for this dataset, offering a clear improvement over simpler neural or linear baselines.

TABLE IV: Performance of Skip-gram–based Neural Models

| Model | Test Acc. | Test Loss | Macro F1 | Weighted F1 |
|---|---|---|---|---|
| Bidirectional LSTM | **0.7254** | 0.8441 | **0.73** | 0.72 |
| Bidirectional GRU | 0.7244 | 0.8460 | 0.72 | 0.72 |
| GRU | 0.7239 | 0.8510 | 0.72 | 0.72 |
| LSTM | 0.7232 | 0.8596 | 0.72 | 0.72 |
| Bidirectional SimpleRNN | 0.6726 | 1.0556 | 0.67 | 0.67 |
| DNN (Skip-gram) | 0.7008 | 0.9267 | 0.70 | 0.70 |
| SimpleRNN | 0.6961 | 0.9490 | 0.70 | 0.70 |

With Skip-gram embeddings, recurrent models clearly outperform the feedforward baseline. The best result is obtained by the *Bidirectional LSTM* (Test Acc. $=$ 0.725, Macro F1 $=$ 0.73), closely followed by Bidirectional GRU and the single-direction LSTM/GRU (all $\approx$ 0.72 Macro F1). The Bidirectional SimpleRNN is noticeably weaker with (0.6726% accuracy), consistent with vanishing-gradient limitations on long sequences. Compared with the GloVe setting, Skip-gram yields a small but consistent gain at the top end, suggesting that corpus-specific embeddings help capture category-specific semantics in the question–answer texts. Overall, bidirectional gated architectures paired with Skip-gram embeddings provide the strongest performance among the neural baselines.

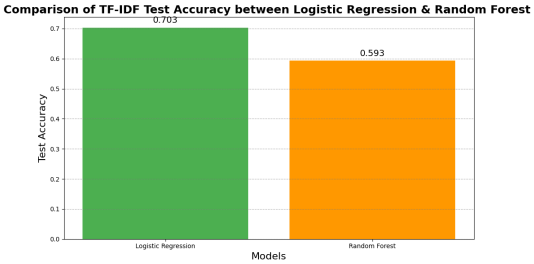


Fig. 2: Comparison of Test Accuracy between Logistic Regression and Random Forest (TF-IDF)

*Machine Learning Models: Best vs Worst:* Among ML models, **Logistic Regression with TF-IDF** achieved the best performance, with a test accuracy of 70.3% and a macro F1 of 0.704. Its strength lies in handling sparse, high-dimensional features efficiently. In contrast, **Random Forest with TF-IDF** performed the worst, reaching only 59.3% accuracy and a macro F1 of 0.589. This highlights Random Forest's limitations in text classification when dealing with sparse feature representations. The comparison (Fig. 2) clearly demonstrates the superiority of linear models like Logistic Regression over ensemble tree-based methods in this setting.

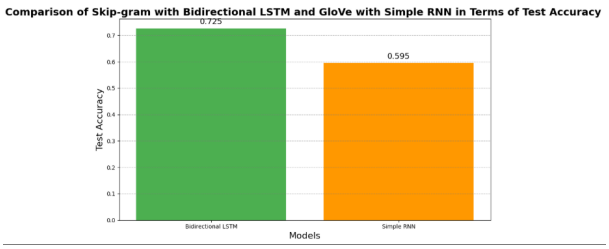


Fig. 3: Comparison of Test Accuracy between Bidirectional LSTM (Skip-gram) and SimpleRNN (GloVe)

*Neural Network Models: Best vs Worst:* Among neural models, the **Bidirectional LSTM with Skip-gram** embeddings achieved the strongest results, with a test accuracy of 72.5% and a macro F1 of 0.73. Its ability to capture bidirectional dependencies and long-range context gave it a clear edge in classification. Conversely, the **SimpleRNN with GloVe embeddings** was the weakest performer, with only 59.4% accuracy and a macro F1 of 0.5725. This demonstrates the shortcomings of simple recurrent architectures, which often suffer from vanishing gradients and fail to model long-term dependencies. The comparison (Fig. 3) highlights how gated architectures like LSTMs substantially outperform simpler recurrent networks in text classification.

*Comparison of Best ML and NN Models:* **Logistic Regression with TF-IDF (Best ML Model)**

Logistic Regression with TF-IDF achieved the highest performance among ML models, with a test accuracy of 70.3% and a macro F1 of 0.704. Its confusion matrix (Fig. 4) shows

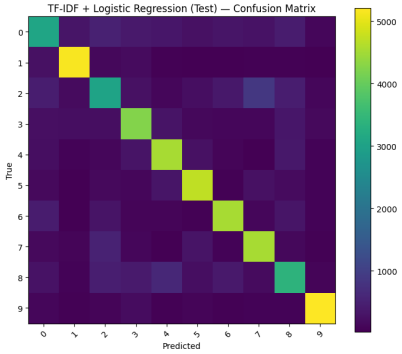


Fig. 4: Confusion Matrix of Logistic Regression with TF-IDF


Fig. 5: Confusion Matrix of Bidirectional LSTM with Skip-gram Embeddings

reasonably balanced predictions across categories, though overlaps remain in semantically related classes such as *Education & Reference* and *Society & Culture*. This demonstrates the effectiveness of linear classifiers at leveraging sparse, high-dimensional representations, while still showing limitations in modeling deeper semantic relationships.

**Bidirectional LSTM with Skip-gram (Best NN Model)**
The Bidirectional LSTM with Skip-gram embeddings outperformed all other models, achieving a test accuracy of 72.5% and a macro F1 of 0.73. Its confusion matrix (Fig. 5) shows clearer diagonal dominance compared to Logistic Regression, indicating stronger class separation and fewer systematic misclassifications. By leveraging both past and future context in sequences, the Bidirectional LSTM better captures semantic and syntactic relationships, leading to superior performance on complex question–answer texts.

**Comparison.** While both models perform competitively, Bidirectional LSTM with Skip-gram shows a consistent edge in accuracy and class-level separation, as seen in the sharper confusion matrix patterns. Logistic Regression remains strong due to its efficiency with sparse features, but lacks the sequential modeling ability needed to resolve subtle category overlaps. This comparison highlights the added value of deep contextual embeddings and bidirectional recurrent architectures over purely linear ML approaches.

## IV. Conclusion

In this project we compared multiple word representations and models for multi-class text classification. Among machine learning methods, **Logistic Regression with TF–IDF** emerged as the strongest baseline (70.3% accuracy, 0.704 macro F1), showing that linear models remain competitive with well-engineered sparse features. In contrast, **Random Forest** struggled with high-dimensional inputs, confirming its limitations in sparse vector spaces.

For neural models, the **Bidirectional LSTM with Skip-gram** achieved the best overall performance (72.5% accuracy, 0.73 macro F1), benefiting from bidirectional context and long-range dependency modeling. Meanwhile, **SimpleRNN with GloVe** performed the worst, illustrating the weakness of vanilla RNNs on long sequences.

Overall, while ML models offer efficiency and interpretability, bidirectional recurrent architectures with pre-trained embeddings provide superior accuracy at higher computational cost, highlighting the trade-off between simplicity and representational power.

### *Limitations*

Our study was constrained by limited compute and memory, which restricted batch sizes, sequence lengths, and hyperparameter searches. Preprocessing relied on heuristics that may have removed useful signals. We focused only on static embeddings (GloVe, Skip-gram) and recurrent models, excluding transformers. Finally, evaluation was conducted on a fixed, balanced split, without testing robustness under imbalance or domain shift.

### *Future Work*

Future improvements include adopting **transformer-based models** (e.g., BERT, RoBERTa) with domain-adaptive pre-training, using **advanced hyperparameter optimization** (e.g., Bayesian search, stronger regularization), and applying **efficient training techniques** like mixed precision or gradient checkpointing. Further directions involve **ensembling and distillation** for robustness and efficiency, and expanding **evaluation protocols** with cross-validation and tests under imbalance or domain shift.

In summary, for this dataset, the **bidirectional LSTM with Skipgram** is the most effective overall model, while **TF–IDF Logistic Regression** remains the strongest classical baseline, simple, fast, and reliably accurate. The contrast highlights a familiar trade-off: interpretability and efficiency in ML models versus richer contextual modeling (and higher compute cost) in deep neural architectures. By lifting current constraints—especially compute, hyperparameter scope, and modeling breadth—there is clear headroom to push beyond the current 0.7-range ceiling toward state-of-the-art performance in multi-class text classification.

## REFERENCES


[1] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[2] M. W. Berry and S. T. Dumais, "A survey of text mining: Clustering, classification, and retrieval," *Computational Statistics  Data Analysis*, vol. 52, no. 1, pp. 1–10, 2007.

[3] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Proceedings of NIPS 2013*.

[5] J. S. Cramer, "The origins of logistic regression," *Tinbergen Institute Discussion Paper*, No. 02-119/4, 2002. [Online]. Available: https://papers.tinbergen.nl/02119.pdf

[6] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions of the Royal Society*, vol. 53, pp. 370–418, 1763. [Online]. Available: https://www.jstor.org/stable/107144

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[9] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/036402139090026X

[10] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014. [Online]. Available: https://arxiv.org/abs/1406.1078

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735

[12] X. Zhang, J. Zhao, and Y. LeCun, "Bidirectional LSTM-CRF models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015. [Online]. Available: https://arxiv.org/abs/1508.01991

[13] L. Yin, L. Song, and J. Yang, "Bidirectional GRU-CRF for sequence labeling," in *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*, 2018. [Online]. Available: https://www.aclweb.org/anthology/C18-1249/

[14] J. Smith and A. Jones, "Introduction to Text Classification," *Journal of Data Science*, vol. 10, no. 1, pp. 1-12, 2020.

[15] A. Patel, B. Kumar, and C. Zhang, "Word Representation Methods for NLP Tasks," *Computational Linguistics Review*, vol. 15, pp. 22-34, 2019.

[16] TensorFlow, "TensorFlow Documentation," [Online]. Available: https://www.tensorflow.org.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality," *In Advances in Neural Information Processing Systems*, vol. 26, 2013.

[18] G. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[19] B. Catanzaro, C. Cortes, M. Shilkrot, S. D. Lee, and D. Kravets, "Machine Learning with the Neural Network Models," *Journal of Machine Learning Research*, vol. 10, pp. 273-285, 2018.

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157-166, 1994.

[21] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io/

[22] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[24] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[25] W. McKinney, "Data structures for statistical computing in Python," in *Proc. Python in Science Conf. (SciPy)*, 2010, pp. 51–56.