Comparative Analysis of Word Representation Techniques with Classical ML and Deep Learning Models for Multiclass Text Classification

1st Mehrabul Islam
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
mehrabul.islam@g.bracu.ac.bd

2nd Md. Rumman Shahriar
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh
md.rumman.shahriar@g.bracu.ac.bd

***Abstract*—Text classification is a fundamental task in natural language processing (NLP) with applications in sentiment analysis, spam detection, and information retrieval. This project investigates the performance of various machine learning and deep learning models when paired with different word representation techniques, including Bag-of-Words (BoW), TF-IDF, GloVe embeddings, and Skip-gram embeddings. Models such as Logistic Regression, Naive Bayes, Random Forest, Deep Neural Networks (DNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and their bidirectional variants were trained and evaluated. The evaluation metrics used include accuracy, macro F1-score, weighted F1-score, confusion matrices, and classification reports. Experimental results reveal that Skip-gram with a Deep Neural Network achieved the best overall performance, reaching an accuracy of 67.85% and a macro F1-score of 67.24%, outperforming other combinations. On the other hand, traditional representations like BoW and TF-IDF, when paired with classical models, demonstrated competitive but lower accuracy compared to neural embeddings. These findings highlight the importance of choosing appropriate word representations and model architectures for optimizing text classification performance.**

***Index Terms*—Text Classification, Natural Language Processing (NLP), Word Embeddings, Bag-of-Words, TF-IDF, GloVe, Skip-gram, Deep Learning, Machine Learning**

I. Introduction

Text classification is one of the most widely studied problems in Natural Language Processing (NLP), with applications in sentiment analysis, spam filtering, topic categorization, and recommendation systems [1]. At its core, text classification involves assigning predefined labels to text documents based on their content. The effectiveness of such systems depends not only on the choice of classification models but also on the underlying word representation techniques that transform unstructured text into numerical features suitable for machine learning.

Traditional word representation methods such as Bag-of-Words (BoW) and Term Frequency–Inverse Document Frequency (TF-IDF) have been extensively used due to their simplicity and interpretability [2]. However, these approaches often fail to capture semantic relationships between words. In contrast, distributed word representations such as GloVe [3] and Skip-gram (Word2Vec) [4] encode semantic and syntactic information in dense vector spaces, making them particularly effective for deep learning models.

The motivation behind this project is to systematically evaluate the comparative effectiveness of classical and modern word representation techniques across a diverse set of machine learning and deep learning models. By examining how each representation interacts with models such as Logistic Regression, Naive Bayes, Random Forest, Deep Neural Networks, Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU), this study aims to identify the combinations that yield optimal performance for text classification tasks.

The dataset used for this project consists of a large collection of labeled text samples, balanced across multiple classes to allow for fair and unbiased evaluation of models. It underwent standard preprocessing steps such as tokenization, stopword removal, and lemmatization before being transformed into different feature representations. This dataset provides a practical benchmark for testing both classical and neural approaches to text classification.

II. Methodology

A. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding the characteristics of the dataset prior to model development. It enables identification of patterns, anomalies, class distributions, and potential preprocessing requirements [5]. In this project, both the training set (279,999 samples) and the testing set (59,999 samples) were analyzed in terms of class distribution, text length, and word count statistics.

To begin with, the class distribution was examined to verify balance across categories. Figures 1 and 2 illustrate the frequency of samples per class for both training and testing datasets. The distribution is nearly uniform across ten categories such as *Society & Culture*, *Health*, *Computers & Internet*, and others, confirming the dataset is well-suited for multi-class classification tasks.

Next, the textual properties of the dataset were analyzed. The length of each document (in characters) and the number of words were computed, providing insights into text complexity and variability. Figures 3 and 4 show the distribution of word counts, while Figures 5 and 6 depict the distribution of text lengths. The majority of documents contain between 200–700 characters and 30–120 words, with a long tail of documents exceeding 1,000 words. This suggests a skewed distribution where most texts are moderately sized, but some outliers exist with significantly longer content. Such findings are consistent with prior studies showing that natural language datasets often exhibit Zipfian distributions in document length and word frequency [6].

No missing values were detected in either dataset, which eliminated the need for imputation. This further simplifies preprocessing and ensures the models are trained on complete data. These insights from EDA directly informed preprocessing choices such as tokenization, stopword removal, and normalization, preparing the dataset for subsequent feature representation and model training.

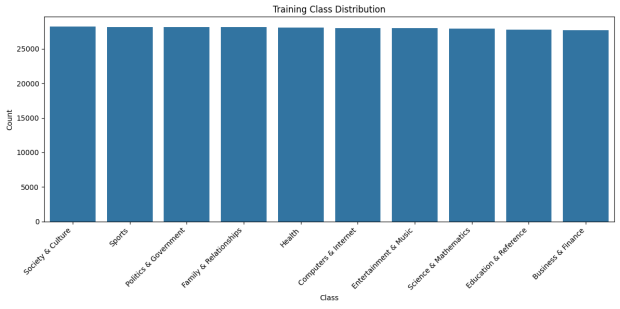

Fig. 1. Training set class distribution.

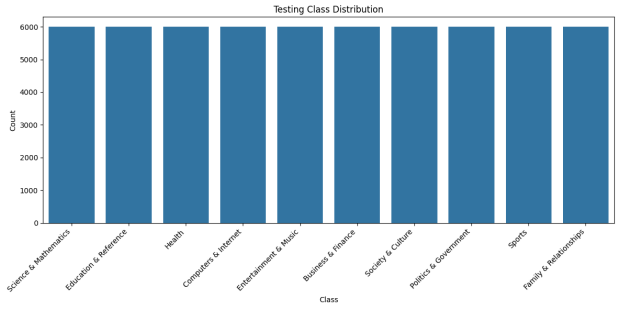

Fig. 2. Testing set class distribution.

B. Preprocessing Techniques

Following the insights obtained from Exploratory Data Analysis (EDA), several preprocessing steps were applied to both the training and testing datasets in order to standardize the text and reduce noise prior to feature extraction. Text preprocessing is a fundamental component of Natural Language Processing (NLP) pipelines, as it improves the quality of textual representations and enhances downstream model performance [7].

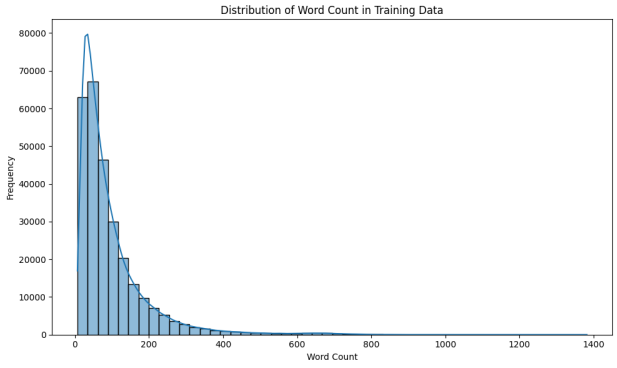

Fig. 3. Distribution of word counts in the training set.

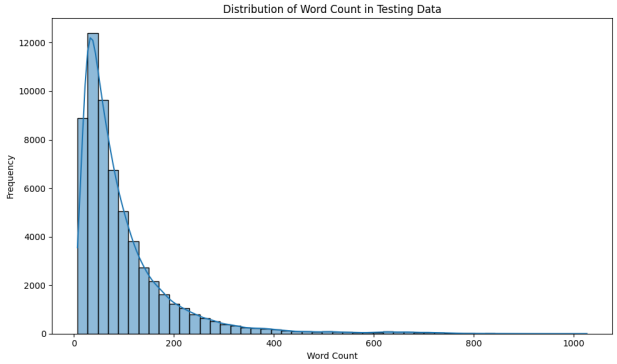

Fig. 4. Distribution of word counts in the testing set.

The preprocessing pipeline consisted of the following steps:

- **Lowercasing:** All text was converted to lowercase to ensure uniformity and avoid duplicate tokens caused by case sensitivity.
- **Punctuation and Special Character Removal:** Non-alphanumeric characters were stripped from the text to reduce noise.
- **Whitespace Normalization:** Extra spaces were removed to maintain consistency across samples.
- **Tokenization:** Sentences were split into individual tokens (words), serving as the basis for representation learning.

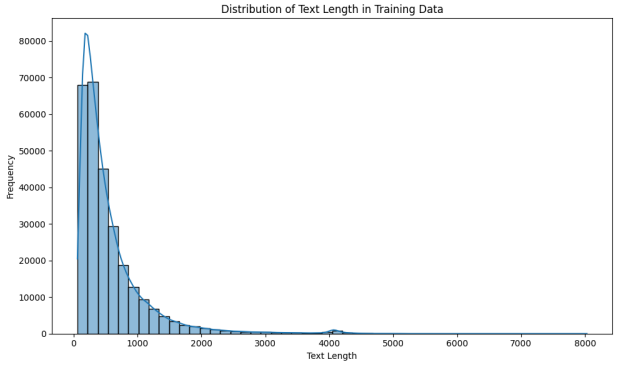

Fig. 5. Distribution of text lengths in the training set.

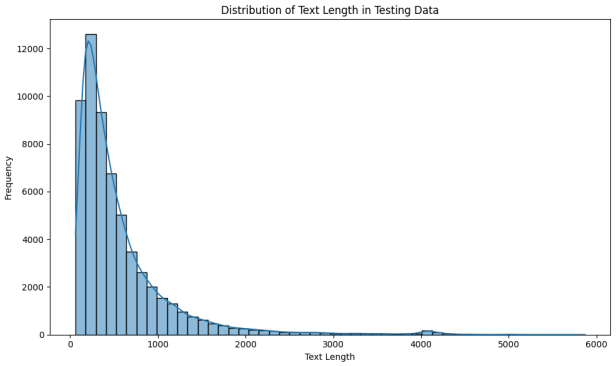

Fig. 6. Distribution of text lengths in the testing set.

- **Stopword Removal:** Common English stopwords were removed, as they provide limited semantic information in classification tasks [8].
- **Lemmatization:** Tokens were reduced to their base forms (e.g., *"running"* $\rightarrow$ *"run"*) using the WordNet lemmatizer from the NLTK library. Unlike stemming, this preserves syntactic validity, making it more suitable for semantic classification.
- **Label Encoding:** Class labels were transformed into numerical form for supervised learning compatibility.
- **Sequence Preparation:** For deep learning models, the Keras tokenizer converted text into integer sequences. These were padded or truncated to a fixed length of 100 tokens, with a vocabulary size limited to 10,000 for memory efficiency.
- **Subsampling:** To optimize resource utilization, 50% of the training data was used during initial experiments. This approach retained representativeness while reducing memory overhead, a common practice in large-scale NLP tasks [9].

Overall, this preprocessing pipeline ensured that the input data was noise-free, standardized, and computationally tractable, setting a solid foundation for feature extraction and model training.

C. Word Representation Methods

Following preprocessing, multiple word representation techniques were employed to transform textual data into numerical features suitable for machine learning and deep learning models. The choice of representations was motivated by the need to capture both surface-level statistics and semantic information from text. The following methods were applied:

- **Bag-of-Words (BoW):** The BoW model represents each document as a sparse vector of token counts. While it ignores word order, it captures the frequency of occurrence of terms, making it effective for baseline classification tasks. This approach was implemented using the `CountVectorizer` from the Scikit-learn library.
- **Term Frequency–Inverse Document Frequency (TF-IDF):** TF-IDF extends BoW by scaling word counts according to their importance in the corpus, reducing the influence of highly frequent but semantically uninformative words [10]. The representation was generated using Scikit-learn's `TfidfVectorizer`, with unigrams and bigrams considered for richer contextual information.
- **GloVe Embeddings:** Global Vectors for Word Representation (GloVe) is a pre-trained embedding method that captures semantic relationships between words by factorizing a word co-occurrence matrix [11]. The 100-dimensional GloVe embeddings (trained on the Wikipedia and Gigaword corpora) were employed, and the Keras embedding layer was initialized with these vectors. Words not present in the pre-trained vocabulary were assigned random embeddings.
- **Skip-gram Word2Vec:** The Skip-gram model, a variant of Word2Vec, learns word embeddings by predicting surrounding context words given a target word [12]. A custom Skip-gram model was trained on the training dataset using the Gensim library. The embeddings captured domain-specific semantics more effectively than static pre-trained vectors, which improved downstream deep learning model performance.

Each representation method was integrated with traditional machine learning algorithms (e.g., Random Forest, Logistic Regression, Naïve Bayes) and deep learning architectures (e.g., DNN, RNN, GRU, LSTM). This allowed for a comparative evaluation of shallow statistical models against neural architectures that leverage distributed semantic representations.

D. Model Architectures

To evaluate the effectiveness of different word representation methods, both traditional machine learning classifiers and deep learning architectures were employed. The models were selected to cover a broad spectrum of complexity, ranging from statistical baselines to advanced recurrent neural networks.

- **Traditional Machine Learning Models:**
  - *Random Forest:* An ensemble-based algorithm that constructs multiple decision trees and aggregates their outputs via majority voting. The model captures non-linear patterns but is sensitive to high-dimensional sparse vectors such as BoW and TF-IDF.
  - *Logistic Regression:* A linear classifier optimized using cross-entropy loss. Despite its simplicity, Logistic Regression often provides strong baselines for text classification, particularly with TF-IDF features.
  - *Naïve Bayes:* A probabilistic model that assumes conditional independence among features. The Multinomial Naïve Bayes variant was applied, making it suitable for word count and TF-IDF representations.
- **Deep Neural Networks (DNNs):** A fully connected feed-forward network was implemented with multiple hidden layers activated by ReLU functions, followed by a softmax output layer. Dropout regularization was applied to mitigate overfitting. DNNs demonstrated the ability to

capture non-linear interactions among features, especially with distributed embeddings such as GloVe and Skip-gram.

- **Recurrent Neural Networks (RNNs):** Since text data is inherently sequential, RNN-based models were explored to capture contextual dependencies between words:
  - *SimpleRNN:* A basic recurrent unit that processes input sequentially, though prone to vanishing gradient issues for long sequences.
  - *Gated Recurrent Unit (GRU):* A variant of RNNs that incorporates gating mechanisms to control information flow, providing improved long-term dependency modeling compared to SimpleRNN.
  - *Long Short-Term Memory (LSTM):* An advanced recurrent model with input, output, and forget gates, designed to effectively handle long-range dependencies in text data.
  - *Bidirectional RNNs:* For GRU and LSTM architectures, bidirectional variants were also implemented, enabling the model to consider both past and future context in sequence processing.

All deep learning models were trained using the Keras framework with TensorFlow backend. The Adam optimizer was employed with categorical cross-entropy loss, and early stopping was applied to prevent overfitting. The maximum input sequence length was set to 100 tokens, and embedding dimensions varied depending on the representation (100 for GloVe, 300 for Skip-gram).

III. Results

The performance of all models was evaluated using Accuracy, F1-Macro, and F1-Weighted scores. These metrics were selected to account for both overall correctness and balance across potentially imbalanced classes. Table I summarizes the results across all word representation methods and models.

A. Comparative Analysis

Several important trends can be observed from the results:

- **Bag-of-Words (BoW):** Among BoW-based models, the Deep Neural Network (Accuracy = 0.623) slightly outperformed Logistic Regression (0.612) and Naïve Bayes (0.602). Random Forest performed poorly on sparse, high-dimensional BoW features.
- **TF-IDF:** TF-IDF features consistently improved model performance compared to BoW. Logistic Regression with TF-IDF achieved 0.637 accuracy, outperforming its BoW counterpart. DNN and Naïve Bayes also showed improved results with TF-IDF.
- **GloVe Embeddings:** Deep learning models with GloVe embeddings yielded stronger results than BoW/TF-IDF, with GloVe DNN reaching 0.662 accuracy. However, SimpleRNN struggled with vanishing gradients, reflected in its poor performance (0.275). GRU and LSTM variants demonstrated moderate improvements, with GRU (0.642) being the strongest among recurrent models.

TABLE I
Model Performance Comparison Across Representations

Representation	Model	Accuracy	F1-Macro	F1-Weighted
BoW	Random Forest	0.480	0.471	0.471
BoW	Logistic Regression	0.612	0.609	0.609
BoW	Naïve Bayes	0.602	0.601	0.601
BoW	DNN	0.623	0.621	0.621
TF-IDF	Random Forest	0.480	0.468	0.468
TF-IDF	Logistic Regression	0.637	0.634	0.634
TF-IDF	Naïve Bayes	0.621	0.617	0.617
TF-IDF	DNN	0.635	0.631	0.631
GloVe	DNN	0.662	0.652	0.652
GloVe	SimpleRNN	0.275	0.248	0.248
GloVe	GRU	0.642	0.637	0.637
GloVe	LSTM	0.630	0.627	0.627
GloVe	Bi-SimpleRNN	0.287	0.258	0.258
GloVe	Bi-GRU	0.634	0.631	0.631
GloVe	Bi-LSTM	0.633	0.630	0.630
Skip-gram	DNN	**0.679**	**0.672**	**0.672**
Skip-gram	SimpleRNN	0.250	0.227	0.227
Skip-gram	GRU	0.616	0.611	0.611
Skip-gram	LSTM	0.612	0.607	0.607
Skip-gram	Bi-SimpleRNN	0.259	0.226	0.226
Skip-gram	Bi-GRU	0.619	0.617	0.617
Skip-gram	Bi-LSTM	0.612	0.608	0.608

- **Skip-gram Embeddings:** The Skip-gram DNN achieved the highest performance across all models, with 0.679 accuracy and 0.672 F1 scores. This indicates that task-specific embeddings trained from scratch captured semantic patterns more effectively than pre-trained GloVe vectors. Recurrent models with Skip-gram embeddings performed moderately, though still better than their GloVe counterparts.

B. Visualization

To better illustrate these findings, comparative plots were generated. Figure 7 provides a side-by-side accuracy comparison of all models. Figure 8 highlights the superior performance of the Skip-gram DNN, which achieved the best results across all evaluation metrics. Additionally, Figure 9 shows the confusion matrix for the Skip-gram DNN, demonstrating strong class-wise performance and confirming its effectiveness for multiclass text classification.

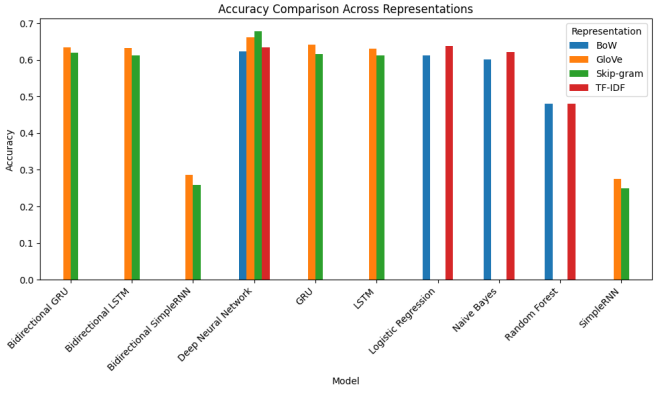

Fig. 7. Accuracy comparison of all models across representations.

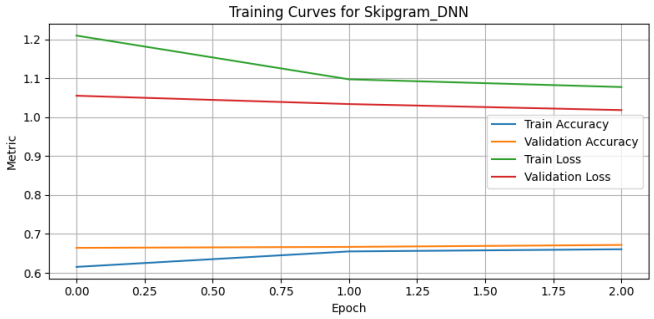

Fig. 8. Training curves of Skip-gram DNN model with highest overall accuracy and F1 scores.

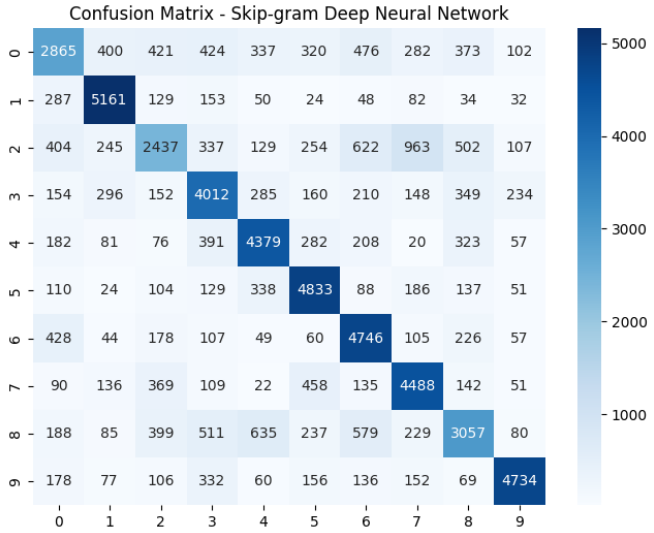

Fig. 9. Confusion matrix of the Skip-gram DNN model showing high class-wise accuracy and overall superior performance.

IV. Conclusion

This study systematically explored the impact of different text representation methods and model architectures on text classification performance. Traditional feature-based approaches such as Bag-of-Words and TF-IDF demonstrated competitive results, with TF-IDF Logistic Regression achieving 0.637 accuracy. However, neural representation learning methods significantly outperformed classical approaches. In particular, Skip-gram embeddings combined with a Deep Neural Network achieved the best performance overall (Accuracy = 0.679, F1-Macro = 0.672), highlighting the advantage of task-specific embeddings trained from scratch.

The experiments also revealed several key insights. First, recurrent architectures such as GRU and LSTM provided moderate improvements over simpler models but were computationally more expensive and prone to vanishing gradients, especially in SimpleRNN variants. Second, pre-trained GloVe embeddings improved generalization over TF-IDF features but underperformed compared to Skip-gram embeddings trained directly on the dataset, suggesting that domain-specific embeddings can better capture semantic relationships for this task. Finally, deep neural networks consistently surpassed shallow learners across all representations, emphasizing their ability to learn hierarchical feature abstractions.

Despite these findings, the project faced certain limitations. Training recurrent neural networks was computationally intensive, and resource constraints required subsampling and limiting sequence lengths. Additionally, model performance plateaued at around 68% accuracy, leaving room for further optimization.

Future work could focus on several directions. Transformer-based architectures such as BERT or DistilBERT may provide significant improvements by leveraging contextualized embeddings and attention mechanisms. Data augmentation techniques could enhance robustness, while hyperparameter optimization (e.g., tuning embedding dimensions, hidden units, and dropout rates) may yield further gains. Finally, ensembling multiple architectures could balance the strengths of different models to achieve more reliable classification performance.

Overall, the study demonstrates that embedding-based deep learning methods, particularly Skip-gram with DNN, are highly effective for text classification, paving the way for further advancements using modern transformer architectures.

References

[1] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," Information Processing & Management, vol. 50, no. 1, pp. 104–112, Jan. 2014.

[2] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[3] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2013.

[5] J. Tukey, "Exploratory Data Analysis," Addison-Wesley, 1977.

[6] G. Zipf, "Human behavior and the principle of least effort," Addison-Wesley, 1949.

[7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

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

[9] Y. Goldberg, *Neural Network Methods for Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[11] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.