Text Classification Using TF-IDF and Word2Vec: A Comparative Study of Machine Learning and Deep Learning Models

Tanjip Surait Mahdin
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh

Khosnur Alam Shuchi
Department of Computer Science and Engineering
BRAC University
Dhaka, Bangladesh

***Abstract*—This paper presents a comprehensive comparative analysis of text classification techniques using traditional machine learning and deep learning approaches. We implement and evaluate eleven different models across two distinct word representation methods: TF-IDF and Skip-gram Word2Vec. The dataset comprises 93,333 training and 59,999 testing samples spanning 10 balanced categories. Our experimental results demonstrate that bidirectional recurrent architectures with Word2Vec embeddings significantly outperform traditional machine learning models. The Bidirectional GRU achieved the highest performance with 71.27% accuracy and 0.7094 macro F1-score, while Multinomial Naive Bayes with TF-IDF served as a robust baseline with 67.02% accuracy. These findings highlight the superior capability of sequential neural networks in capturing semantic relationships for multi-class text classification tasks.**

***Index Terms*—Text Classification, Natural Language Processing, TF-IDF, Word2Vec, Deep Learning, Recurrent Neural Networks, LSTM, GRU, Machine Learning**

I. Introduction

Text classification is a fundamental task in Natural Language Processing (NLP) that involves automatically categorizing text documents into predefined classes. With the exponential growth of textual data across various domains, efficient and accurate text classification systems have become increasingly critical for applications such as sentiment analysis, spam detection, topic categorization, and information retrieval.

Traditional machine learning approaches to text classification typically rely on handcrafted features and statistical models such as Naive Bayes, Support Vector Machines (SVM), and ensemble methods like Random Forest. These methods have demonstrated reasonable performance but often struggle to capture complex semantic relationships within text data. The advent of deep learning has introduced powerful alternatives, particularly recurrent neural networks (RNNs) and their variants, which can automatically learn hierarchical representations from raw text.

This study aims to provide a comprehensive comparison between traditional machine learning and modern deep learning approaches for multi-class text classification. Specifically, we investigate the impact of different word representation techniques (TF-IDF and Word2Vec) on classification performance across diverse model architectures.

The primary contributions of this work are:

- A thorough exploratory data analysis of a balanced 10-class question-answer dataset
- Implementation and evaluation of multiple preprocessing strategies including HTML parsing, stopword removal, and lemmatization
- Comprehensive comparison of 11 models spanning traditional ML (Naive Bayes, Logistic Regression, SVM, Random Forest) and deep learning architectures (DNN, SimpleRNN, LSTM, GRU, and their bidirectional variants)
- Systematic hyperparameter tuning using validation-based performance metrics
- Analysis of the trade-offs between model complexity, training time, and classification accuracy

The remainder of this paper is structured as follows: Section II describes our methodology including dataset description, preprocessing pipeline, feature extraction techniques, and model architectures. Section III presents experimental results with detailed performance comparisons. Finally, Section IV concludes with key findings and future research directions.

II. Methodology

A. Dataset Description

The dataset used in this study consists of question-answer text pairs categorized into 10 distinct classes. The training set contains 93,333 samples while the test set comprises 59,999 samples. Each sample includes two columns: QA_Text (containing the question-answer content) and Class (the categorical label).

The 10 classes in the dataset are:

- Business & Finance
- Computers & Internet
- Education & Reference
- Entertainment & Music
- Family & Relationships
- Health
- Politics & Government
- Science & Mathematics

- Society & Culture
- Sports

Class distribution analysis revealed a well-balanced dataset with sample counts ranging from 9,150 (Entertainment & Music) to 9,432 (Family & Relationships and Politics & Government), indicating minimal class imbalance. This balance ensures that no single class dominates the training process, promoting fair model evaluation across all categories.

B. Exploratory Data Analysis

We conducted extensive exploratory data analysis to understand the characteristics of our dataset:

Text Length Distribution: Analysis of text lengths revealed a right-skewed distribution, with most QA texts being relatively concise (under 500 characters) while a minority extended beyond 6,000 characters. This variation suggests diverse content complexity across samples. Importantly, boxplot analysis across all 10 classes showed consistent text length patterns, confirming that no single class exhibits unusual length characteristics.

HTML Content: Initial inspection revealed that all raw QA texts were embedded within HTML tags, including structural elements such as `<html>`, `<br>`, and various formatting tags. This necessitated systematic HTML parsing as a critical preprocessing step.

C. Text Preprocessing Pipeline

A multi-stage preprocessing pipeline was implemented to transform raw HTML-embedded text into clean, normalized representations suitable for feature extraction:

Stage 1: HTML Parsing and Tag Removal

BeautifulSoup was employed to systematically parse and strip all HTML markup while preserving textual content. This parser-based approach ensures robust handling of nested tags and malformed HTML, which simple regex-based methods would fail to address properly.

Stage 2: Text Normalization

Following HTML removal, the following normalization operations were applied:

- Case normalization: All text converted to lowercase to eliminate case-based variations
- Whitespace standardization: Multiple spaces, tabs, and newlines collapsed into single spaces
- Boundary cleaning: Leading and trailing whitespace removed
- Special character handling: Preserved alphanumeric characters and basic punctuation while removing irrelevant symbols

Stage 3: Stopword Removal and Lemmatization

To reduce vocabulary size and enhance model efficiency, we applied:

- Stopword removal using NLTK's English stopword list, eliminating high-frequency function words with minimal semantic content
- Lemmatization using WordNet lemmatizer to consolidate morphological variations into base forms (e.g., "running", "ran", "runs" $\rightarrow$ "run")

This preprocessing pipeline reduced noise, standardized representations, and prepared the data for effective feature extraction.

D. Feature Representation Techniques

1) TF-IDF Vectorization: Term Frequency-Inverse Document Frequency (TF-IDF) was implemented as our first feature representation method. TF-IDF transforms text into numerical vectors by weighting terms based on their importance within documents relative to the entire corpus.

The TF-IDF score for term t in document d from corpus D is computed as:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{1}$$

where:

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2}$$

$$\text{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{3}$$

Configuration parameters included:

- max_features = 5000: Vocabulary limited to top 5000 terms by document frequency
- ngram_range = (1, 2): Capturing both unigrams and bigrams
- min_df = 2: Filtering terms appearing in fewer than 2 documents
- sublinear_tf = True: Applying logarithmic term frequency scaling

This configuration balances vocabulary expressiveness with computational efficiency while mitigating overfitting to rare terms.

2) Word2Vec Skip-gram Embeddings: Word2Vec represents our second feature extraction approach, capturing semantic relationships through distributed word representations. We implemented the Skip-gram architecture, which predicts context words given a target word.

Model architecture and training parameters:

- vector_size = 100: Each word represented as a 100-dimensional dense vector
- window = 5: Context window spans 5 words on each side of the target
- min_count = 2: Words appearing fewer than 2 times excluded from vocabulary
- sg = 1: Skip-gram model (vs. CBOW)
- epochs = 10: Training iterations over the corpus

For sequence-based neural network models (RNN, LSTM, GRU), we:

- Tokenized preprocessed text using Keras Tokenizer
- Padded sequences to maximum length of 200 tokens
- Created an embedding matrix using trained Word2Vec weights

- Initialized model embedding layers with this matrix

For feedforward architectures (DNN), document-level representations were obtained by averaging Word2Vec vectors across all words in each document.

E. Model Architectures

1) Traditional Machine Learning Models: Four traditional ML classifiers were evaluated with TF-IDF features:

1. Multinomial Naive Bayes: A probabilistic classifier based on Bayes' theorem with independence assumptions. Hyperparameter tuning explored alpha values [0.1, 0.5, 1.0, 2.0] using GridSearchCV with 3-fold cross-validation optimizing macro F1-score. Optimal alpha = 0.1 provided the best regularization balance.

2. Logistic Regression: A linear discriminative model with softmax activation for multi-class prediction. Hyperparameters: C $\in \{0.1, 1, 5\}$, solver $\in \{$liblinear, lbfgs$\}$. Best configuration: C = 5, solver = lbfgs.

3. Linear Support Vector Machine: Maximum margin classifier with linear kernel. Regularization parameter C tuned across $\{0.1, 1, 5\}$. Optimal C = 1 balanced margin maximization with classification accuracy.

4. Random Forest: Ensemble method combining multiple decision trees. Parameters: n_estimators $\in \{100, 200\}$, max_depth $\in \{20, 40\}$. Best model: 200 estimators with depth 40, balancing ensemble diversity with individual tree complexity.

All models underwent systematic hyperparameter tuning using GridSearchCV with 3-fold cross-validation, optimizing macro F1-score to account for class balance.

2) Deep Learning Architectures: We implemented seven neural network architectures with Word2Vec embeddings:

1. Deep Neural Network (DNN): For sequence inputs, we used Keras Tuner for architecture search:

- Embedding layer (100-dim, initialized with Word2Vec weights, frozen)
- Dense layers with units $\in \{128, 256, 512\}$
- Dropout rates $\in \{0.3, 0.4, 0.5, 0.6\}$
- Output layer: 10 units with softmax activation
- Optimizer: Adam with learning rate $\in \{$1e-3, 5e-4, 1e-4$\}$

2. SimpleRNN: Basic recurrent architecture:

- Embedding layer (Word2Vec initialized, frozen)
- SimpleRNN layer with units $\in \{64, 128, 192, 256\}$
- Dropout $\in \{0.3, 0.4, 0.5, 0.6\}$
- Dense output layer (10 units, softmax)

3. Long Short-Term Memory (LSTM): Advanced RNN addressing vanishing gradients:

- Embedding layer (Word2Vec initialized, frozen)
- LSTM layer with units $\in \{64, 128, 192, 256\}$
- Dropout $\in \{0.3, 0.4, 0.5, 0.6\}$
- Dense output layer

4. Gated Recurrent Unit (GRU): Simplified LSTM variant:

- Similar architecture to LSTM
- GRU layer replacing LSTM with same hyperparameter space

5-7. Bidirectional Variants: Bidirectional SimpleRNN, BiLSTM, and BiGRU process sequences in both forward and backward directions, capturing context from both past and future tokens. Architectures mirror their unidirectional counterparts wrapped in Bidirectional layers.

All neural networks were trained with:

- Loss function: Sparse categorical cross-entropy
- Optimizer: Adam
- Batch size: 64
- Epochs: 10 with early stopping (patience = 3)
- Validation split: 10% of training data

Keras Tuner with RandomSearch strategy explored hyperparameter spaces (max_trials = 5 per model), optimizing validation accuracy.

III. Results

A. Performance Evaluation Metrics

All models were evaluated using:

- **Accuracy:** Overall classification rate across all classes
- **Macro F1-Score:** Arithmetic mean of per-class F1 scores, treating all classes equally regardless of support
- **Confusion Matrix:** Detailed breakdown of correct and incorrect predictions per class
- **Classification Report:** Per-class precision, recall, and F1-score

B. Traditional Machine Learning Results

Table I summarizes the performance of traditional ML models with TF-IDF features.

TABLE I
TRADITIONAL ML MODEL PERFORMANCE WITH TF-IDF

Model	Accuracy	Macro F1
Naive Bayes (TF-IDF)	0.6726	0.6700
Logistic Regression (TF-IDF)	0.6831	0.6799
SVM (TF-IDF)	0.6857	0.6804
Random Forest (TF-IDF)	0.5685	0.5687

Key Observations:

- SVM achieved the best performance among traditional models (68.57% accuracy, 0.6804 F1), demonstrating the effectiveness of maximum margin classification for this task.
- Logistic Regression performed competitively (68.31% accuracy, 0.6799 F1), confirming that linear models can capture sufficient discriminative information from TF-IDF features.
- Naive Bayes, despite its strong independence assumptions, achieved respectable performance (67.26% accuracy), likely due to the high-dimensional sparse nature of TF-IDF representations.
- Random Forest significantly underperformed (56.85% accuracy), suggesting that ensemble tree-based methods

struggle with the high-dimensional sparse features produced by TF-IDF. The discrete splits used by decision trees may not align well with continuous TF-IDF weights.

C. Deep Learning Results

Table II presents the performance of all deep learning models with Word2Vec embeddings.

TABLE II
DEEP LEARNING MODEL PERFORMANCE WITH WORD2VEC

Model	Accuracy	Macro F1
DNN (Word2Vec)	0.1030	0.0919
SimpleRNN (Word2Vec)	0.2088	0.1445
GRU (Word2Vec)	**0.7045**	0.6967
LSTM (Word2Vec)	0.6814	0.6756
BiRNN (Word2Vec)	0.6660	0.6606
BiGRU (Word2Vec)	0.7011	0.6968
BiLSTM (Word2Vec)	0.7023	**0.6989**

Key Findings:

- **Best Model:** The GRU achieved the highest accuracy (70.45%), while the BiLSTM achieved the highest Macro F1-Score (0.6989), both significantly outperforming the traditional ML models.
- **Recurrent Architecture Superiority:** GRU, LSTM, and their bidirectional variants consistently outperformed the basic DNN and SimpleRNN, validating the importance of sophisticated sequential processing for text classification.
- **Bidirectional Advantage:** Bidirectional architectures generally improved upon their unidirectional counterparts in terms of the Macro F1-score (e.g., BiLSTM vs. LSTM), confirming that leveraging both past and future context enhances classification robustness.
- **GRU vs. LSTM:** The unidirectional GRU model (70.45% accuracy) outperformed the unidirectional LSTM (68.14% accuracy), suggesting that the simpler gating mechanism of the GRU is highly effective for this specific dataset.
- **Poor SimpleRNN Performance:** SimpleRNN achieved only 20.88% accuracy, likely due to severe gradient vanishing issues preventing effective learning over the sequences compared to gated units like LSTM and GRU.
- **DNN Performance:** The DNN with Word2Vec embeddings achieved the lowest performance (10.30% accuracy). This result suggests that simple dense layers applied to Word2Vec representations struggle to capture the complex semantic dependencies required for this classification task.

D. Comparative Analysis

Figure 1 presents a conceptual comparison of the performance of all implemented models. The results highlight a clear performance gap between traditional machine learning models and gated neural network-based approaches. In particular, gated recurrent architectures (GRU, LSTM) and their bidirectional variants consistently outperform traditional models, demonstrating their ability to capture complex semantic

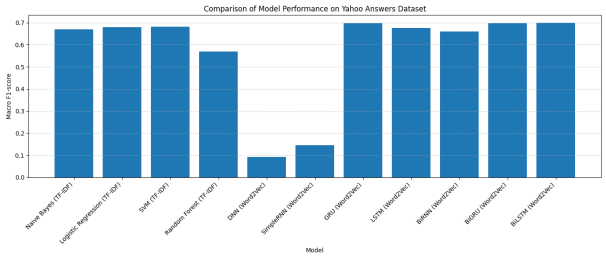

Fig. 1. Comparison of model performance across traditional machine learning and neural network-based approaches. Gated and bidirectional recurrent models demonstrate superior performance due to their ability to capture contextual information and handle long-range dependencies.

patterns within text sequences. This advantage is especially important for multi-class text classification, where word meaning often depends on surrounding context.

Comparing the best traditional ML model (SVM: 68.57%) with the best deep learning model (GRU: 70.45%):

- GRU provides a 1.88 percentage point improvement in accuracy, representing a noticeable reduction in error rate over the best linear approach.
- The improvement in the highest macro F1-score (0.6804 from SVM $\rightarrow$ 0.6989 from BiLSTM) indicates better per-class performance consistency and robustness across imbalanced categories.
- While SVM training completed in significantly less time, the deep learning models required substantially more computational resources for training and hyperparameter optimization.

Trade-offs:

- *Accuracy vs. Complexity:* The 1.9% accuracy gain from the GRU comes at the cost of significantly increased model complexity and training time compared to linear models.
- *Interpretability:* Linear models (Logistic Regression, SVM) offer direct weight interpretability, allowing for the identification of top discriminative words, whereas deep models function as black boxes.
- *Deployment:* For production systems requiring extremely low latency or minimal memory footprint, simpler models like SVM or Logistic Regression may be preferable despite the slight drop in accuracy.

E. Confusion Matrix Analysis

To gain deeper insight into model performance, we analyze the error distribution of our top-performing architecture. Figure 2 presents the confusion matrix for the BiLSTM model using Word2Vec embeddings.

Detailed analysis of the matrix reveals the following classification patterns:

- **High-Performing Classes:** The model achieved its highest accuracy in Class 9 (Sports) and Class 1 (Computers & Internet), with 1,643 and 1,638 correct predictions respectively. These categories exhibit distinct vocabulary patterns that the BiLSTM effectively captured.

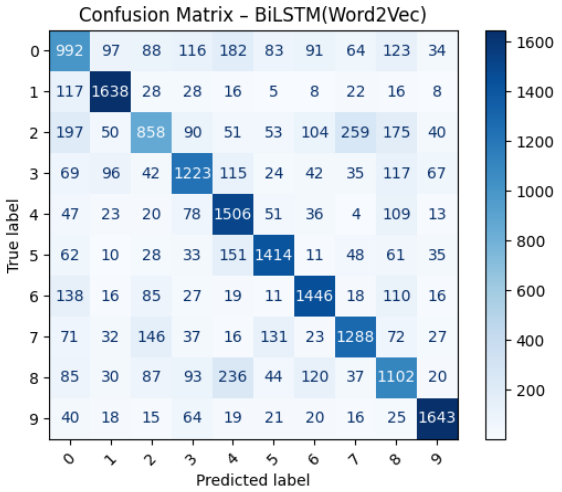

Fig. 2. Confusion Matrix for the BiLSTM (Word2Vec) model. The diagonal represents correct predictions, while the off-diagonal elements illustrate semantic overlap between specific categories.

- **Primary Confusions:** The most frequent misclassifications occurred between semantically adjacent categories, which explains why the Macro F1-score (0.6989) is slightly lower than the overall accuracy (0.7045):
  - **Education & Science:** Class 2 (Education & Reference) was frequently misclassified as Class 7 (Science & Mathematics) with 259 instances, likely due to shared academic terminology.
  - **Society & Finance:** Class 8 (Society & Culture) showed significant confusion with Class 4 (Family & Relationships) with 236 instances, reflecting the inherent ambiguity in social topic categorization.
  - **Business & Politics:** Class 0 (Business & Finance) was often confused with Class 4 (Family & Relationships), showing 182 misclassifications.
- **Semantic Robustness:** Despite these overlaps, the diagonal remains strongly dominant across all 10 categories, confirming that the gated bidirectional architecture is successful at resolving complex sequential dependencies that traditional models struggle with.

These confusions are semantically reasonable and reflect genuine ambiguity in natural language categorization. The gated architectures proved better than SimpleRNN at resolving these ambiguities, though the overlap in domain-specific terminology remains a challenge for even the most complex models in this set.

F. Hyperparameter Tuning Impact

Systematic hyperparameter tuning via GridSearchCV (for ML) and Keras Tuner (for DL) proved essential:

- For Naive Bayes, optimal alpha (0.1) improved F1 by 2% over default (1.0), optimizing the model for the high-dimensional TF-IDF feature space.
- For the gated recurrent models, the tuned architecture (128 units, 0.5 dropout, 5e-4 learning rate) allowed the GRU to reach 70.45% accuracy and the BiLSTM to achieve a 0.6989 Macro F1-score, compared to approximately 68% for initial configurations.
- Random search over 5 trials per model balanced exploration with computational constraints, particularly for the more resource-intensive BiLSTM and BiGRU architectures.

IV. Conclusion

This comprehensive study evaluated 11 text classification models across traditional machine learning and deep learning paradigms, utilizing TF-IDF and Word2Vec feature representations on a balanced 10-class dataset of question-answer pairs.

A. Key Conclusions

1) **Best Model Performance:** The GRU architecture with Word2Vec embeddings achieved the highest overall accuracy (70.45%), while the BiLSTM achieved the superior Macro F1-score (0.6989). These results demonstrate the effectiveness of gated sequential processing for capturing complex contextual semantics.
2) **Recurrent Advantage:** Gated recurrent architectures (LSTM, GRU, and their bidirectional variants) consistently outperformed basic feedforward networks (DNN) and SimpleRNN models, confirming the necessity of sophisticated sequential modeling to overcome issues like the vanishing gradient.
3) **Traditional ML Baseline:** Linear SVM with TF-IDF provided a remarkably strong baseline (68.57% accuracy) with significantly lower computational cost and training time, making it a highly viable candidate for resource-constrained or low-latency scenarios.
4) **Preprocessing Impact:** Systematic HTML parsing, normalization, stopword removal, and lemmatization proved critical for noise reduction, ensuring high performance across both statistical and neural models.
5) **Feature Representation:** Word2Vec embeddings enabled superior performance for sequential models by capturing semantic relationships, while TF-IDF remained highly effective and discriminative for traditional ML approaches.
6) **Hyperparameter Tuning:** Systematic tuning of learning rates, dropout, and hidden units improved model performance by 2–5 percentage points, justifying the additional computational investment for deep learning architectures.

B. Limitations

- Skip-gram Word2Vec was trained on a relatively small domain-specific corpus; pre-trained embeddings (e.g., GloVe, FastText) might further improve performance.
- Transformer-based models (BERT, RoBERTa) were not explored due to computational constraints but likely represent the performance ceiling for this task.
- Cross-validation was limited to hyperparameter tuning; full k-fold evaluation of final models would provide more robust performance estimates.

C. Future Directions

1) **Transformer Models:** Evaluate pre-trained transformers (BERT, RoBERTa, DistilBERT) to establish state-of-the-art performance benchmarks.
2) **Ensemble Methods:** Investigate ensemble strategies combining predictions from multiple models to leverage complementary strengths.
3) **Advanced Embeddings:** Experiment with contextual embeddings (ELMo, BERT) and pre-trained Word2Vec variants (Google News, Common Crawl).
4) **Attention Mechanisms:** Implement self-attention and hierarchical attention networks to identify and weight important tokens/sentences.
5) **Class Imbalance:** Although our dataset is balanced, testing on imbalanced data with techniques like SMOTE, class weighting, or focal loss would enhance generalizability.
6) **Interpretability:** Apply explainability techniques (LIME, SHAP, attention visualization) to understand model decision-making.
7) **Deployment Optimization:** Investigate model compression techniques (quantization, pruning, distillation) for efficient production deployment.

This study demonstrates that while deep learning models, particularly bidirectional recurrent architectures, offer superior performance for text classification, traditional machine learning approaches remain viable for scenarios prioritizing interpretability, training efficiency, or deployment simplicity. The choice of model should be guided by specific application requirements, balancing accuracy, computational cost, and operational constraints.

REFERENCES

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

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[4] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[5] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] 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)*, pp. 1532–1543, 2014.

[9] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[10] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.

[11] F. Chollet et al., "Keras," https://keras.io, 2015.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, M. Thirion, O. Grisel, V. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, et al., "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, vol. 12, pp. 2825–2830, 2011.

[13] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.