Spam Classification of SMSs using Machine Learning Techniques

Armando Orozco, Santa Ana College
Dr. Doina Bein, Department of Computer Science, California State University, Fullerton

Background

The advent of machine learning has revolutionized the approach to text classification, particularly in the Short Message Service (SMS) domain. This research explores different machine learning algorithms and techniques to analyze their effects on sentiment recognition, specifically if an SMS message is undesirable. An undesired message that could possibly be malicious is called Spam.

The goal is to apply and analyze sophisticated machine learning techniques to classify SMS content into Spam or not Spam (Ham) and analyze the effects that every single technique has including if it is suitable to implement in a day-to-day application.

Methods

- **Software Libraries, Tools, and Languages**
  - **Jupyter Notebook** and **Visual Studio Code** were used as the **IDE** and **code editor**, respectively. The primary language was **Python**, and we utilized **Pandas** for data management, **NumPy** for mathematical operations, **Matplotlib** for plotting, and **Scikit-learn** and **Tensorflow** for machine learning. Additionally, we employed NLP techniques and specific **NLP** tokenization techniques using **NLTK's 'TweetTokenizer'** and **'WordNet'** virtual dictionary.

- **Data Collection**
  - Data was collected from the **UCI machine learning repository** and a paper by Mishra, S, & Soni. The **second** dataset is simplified by removing the **'URL'**, **'PHONE'**, and **'EMAIL'** columns, as the focus is on **NLP** techniques without external information. A split of **80%-20%** was used.

- **Natural Language Processing (NLP) Techniques**
  Different steps were applied to the data those are:
  - **Tokenization**:
  - **Lemmatization**
  - **Word Stop Removal**
  - **Padding**
  - **Embedding:** The process of embedding is the process of giving sentiment to sentences, there are several techniques to give sentiment to words, and a handful of them were applied to observe their effects during the training. The techniques that were applied were:
    - **Count Vectorizer**
    - **TF-IDF (Term Frequency-Inverse Document Frequency)**
    - **Hashing vectorizer (not used with NB)**

- **Model Selection and Training**
  Once the Text Preprocessing was done, the next step was to train models. A pool of Machine-learning models and some deep-learning techniques were chosen during this research.

ML Algorithms:
- **Naive Bayes Classifier**
- **K-Nearest neighbors Classifier**
- **Decision Tree Classifier**
- **Random Forest Classifier**

DL algorithms:
- **Long Short-Term Memory**
- **Stacked Long Short-Term Memory**
- **Densely Connected Neural Network**

Methods (continued)

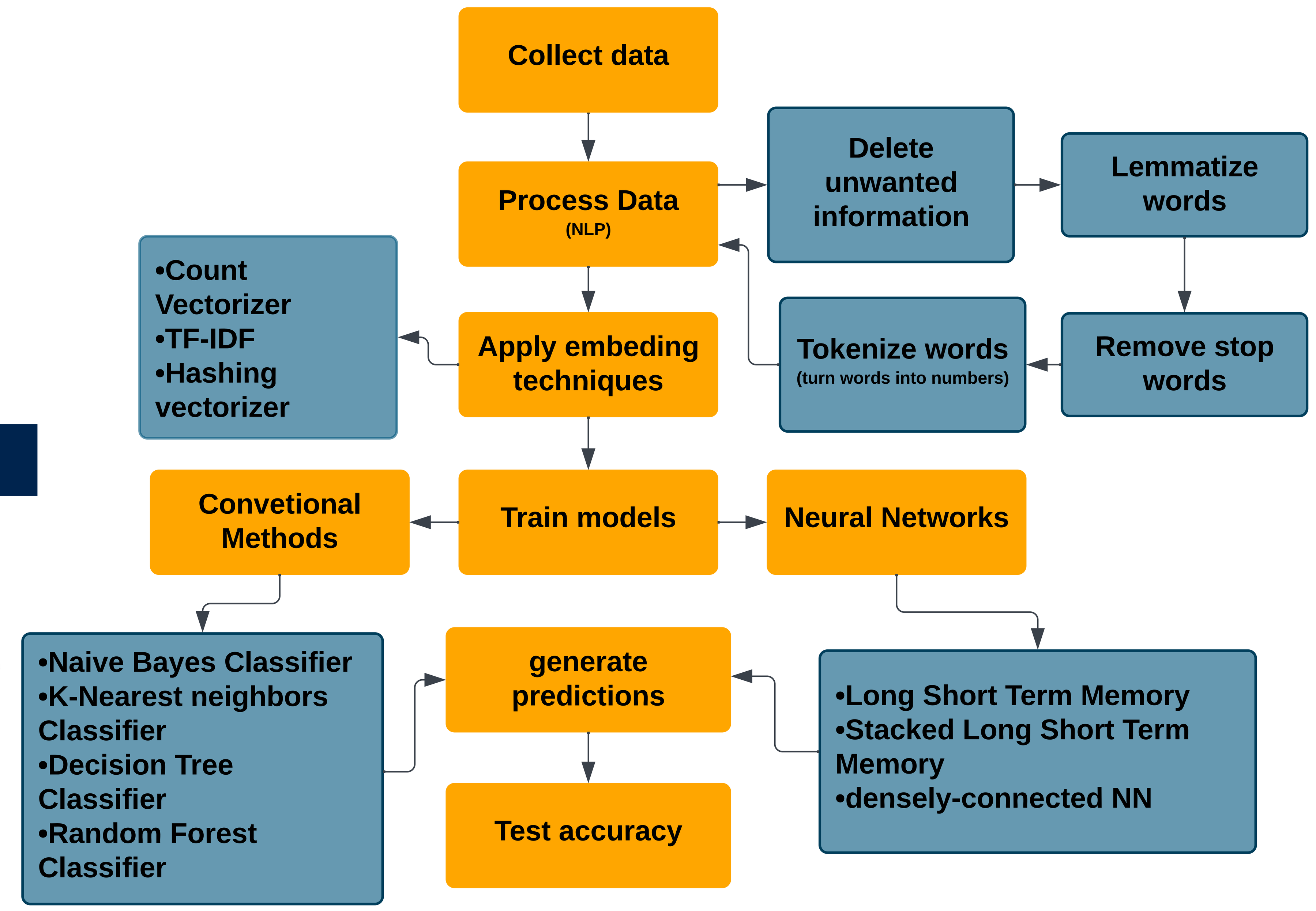

Figure 1:
The NLP process involves data collection, preprocessing, vectorization, model training (conventional models and neural networks), prediction, and accuracy testing.

Results

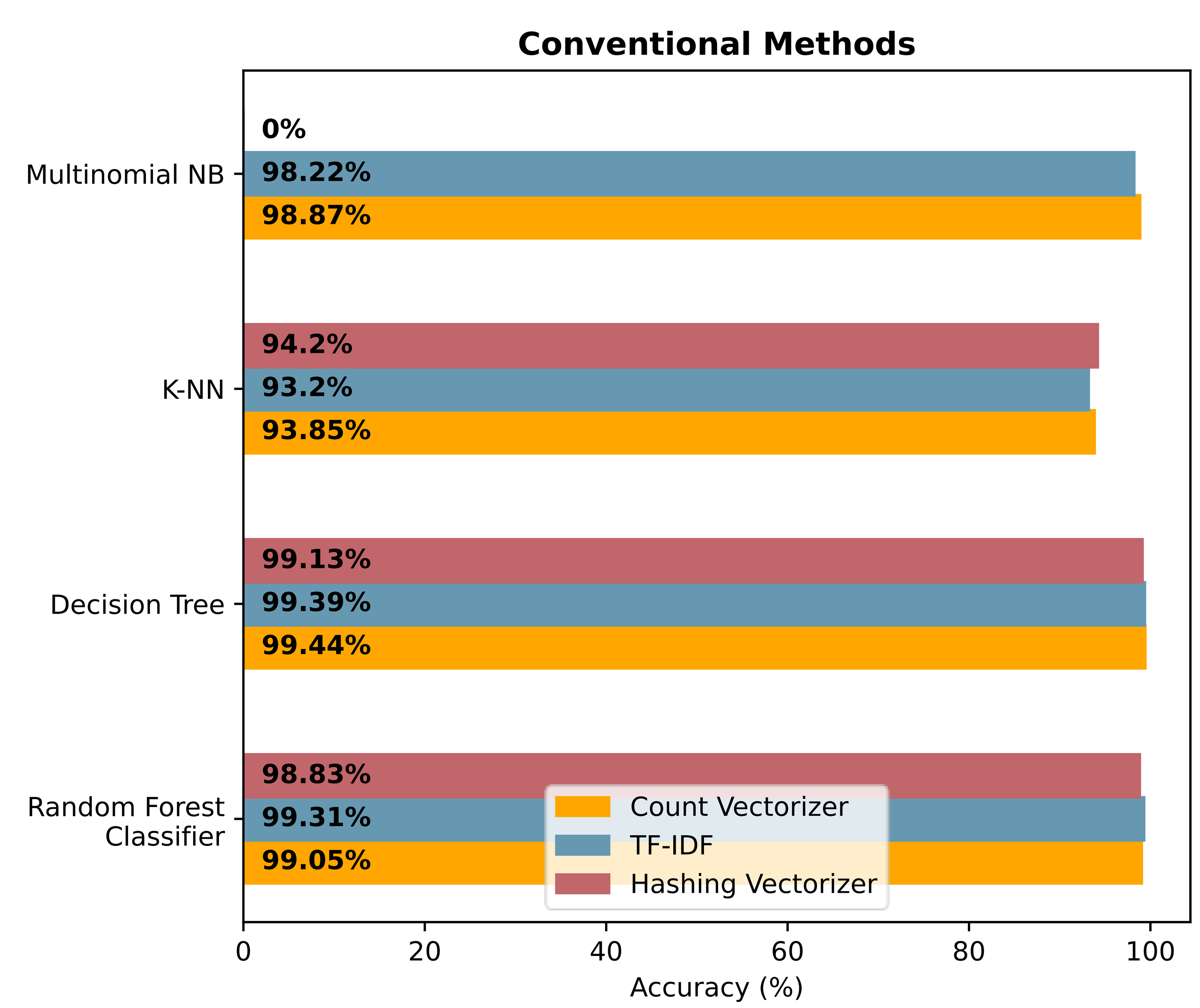

Figure 2:
Comparison of accuracy percentages of conventional machine learning algorithms with their respective vectorization or embedding techniques

Results (continued)

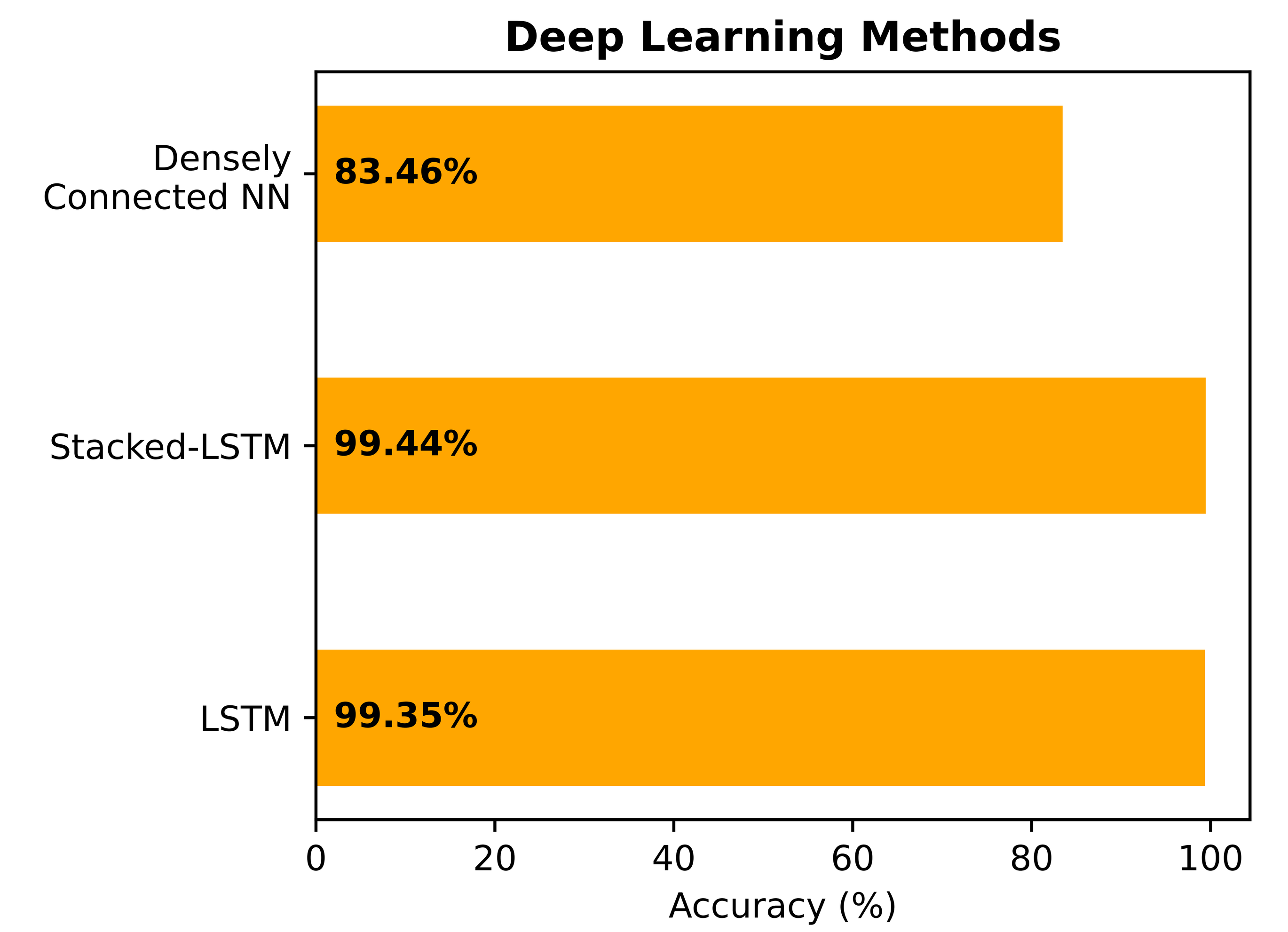

Figure 3:
Comparison of accuracy percentages of Deep Learning algorithms

Conclusion & Future Work

After conducting experiments for our research, we used different methods to evaluate models for classifying messages as spam or ham. The **Naive Bayes**, **K-Nearest Neighbors**, and **Densely Connected Neural Network** models did **not meet** the **desired standard.**

In the case of the Densely connected neural network, its accuracy was lower compared to the LSTM (Long Short-Term Memory) and the Stacked LSTM. This suggests that LSTM may be more effective in classifying text and that using only a Dense layer is insufficient to yield satisfactory results, although it still achieved acceptable accuracy due to its simplicity.

The best models were the **Stacked LSTM** and the **decision tree** with Count Vectorizer. This suggests that **LSTM** may be **better** for **complex problems**, but the **decision tree** showed similar **accuracy**, **faster compilation time**, and simpler implementation, making it more **practical** for simpler real-world problems.

Moving forward, the next steps for further improvement will involve implementing this model in a real-world application, such as a messaging service, a web app, or even an email service, and analyzing its effects.

References & Acknowledgements

1. Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc
2. Almeida,Tiago and Hidalgo,Jos. (2012). SMS Spam Collection. UCI Machine Learning Repository. https://doi.org/10.24432/C5CC84.
3. Mishra, S., & Soni, D. (2020). Smishing Detector: A security model to detect smishing through SMS content analysis and URL behavior analysis. Future Generation Computer Systems, 108, 803-815.

Project supported by Project RAISER, U.S. Department of Education HSI-STEM award P031C210118.