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 way we approach 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 . 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 and if it is suitable to implement in a day-day application.

Methods

- **Software libraries, tools, and languages**
  - For the project, we used **Jupyter Notebook** and **Visual Studio Code** 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**
  NLP techniques preprocess human language, like English, into a computer-readable format, such as numbers. These numbers are then used in a machine-learning model to make predictions:
  - **Tokenization***: The process involves using the **'TweetTokenizer**' from **NLTK** to turn sentences into a list of words, which are then converted into numbers for computer processing.
  - **Lemmatization:** The lemmatization of a word is the process by which a word is converted to its base form, this process uses a pontificated process that analyzes the word's morphology using vocabulary from a dictionary. The library **NLTK** provides the functions to lemmatize words, and also uses the virtual dictionary '**WordNet'.**
  - **Word Stop removal:** This step consisted in removing all the stop words, from the sentences such as "the", "a" or "an", this was done to focus on the more meaningful text.
  - **Tokenization*(continued):** After the words were lemmatized and the stop words were removed, the remaining words were turned into numbers, for example. The word "hello" = 1, and "book" = 2, this would be repeated with all words. After every word was assigned a number, all sentences were rewritten with their respective word-to-number translation
  - **Padding:** After the tokenization and the lemmatization, the sentences need to be converted to lists of the same length, a padding algorithm was provided by the **TensorFlow** library
  - **Embedding:** The process of embedding is the process to give sentiment to words and thus 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.

Methods (continued)

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**

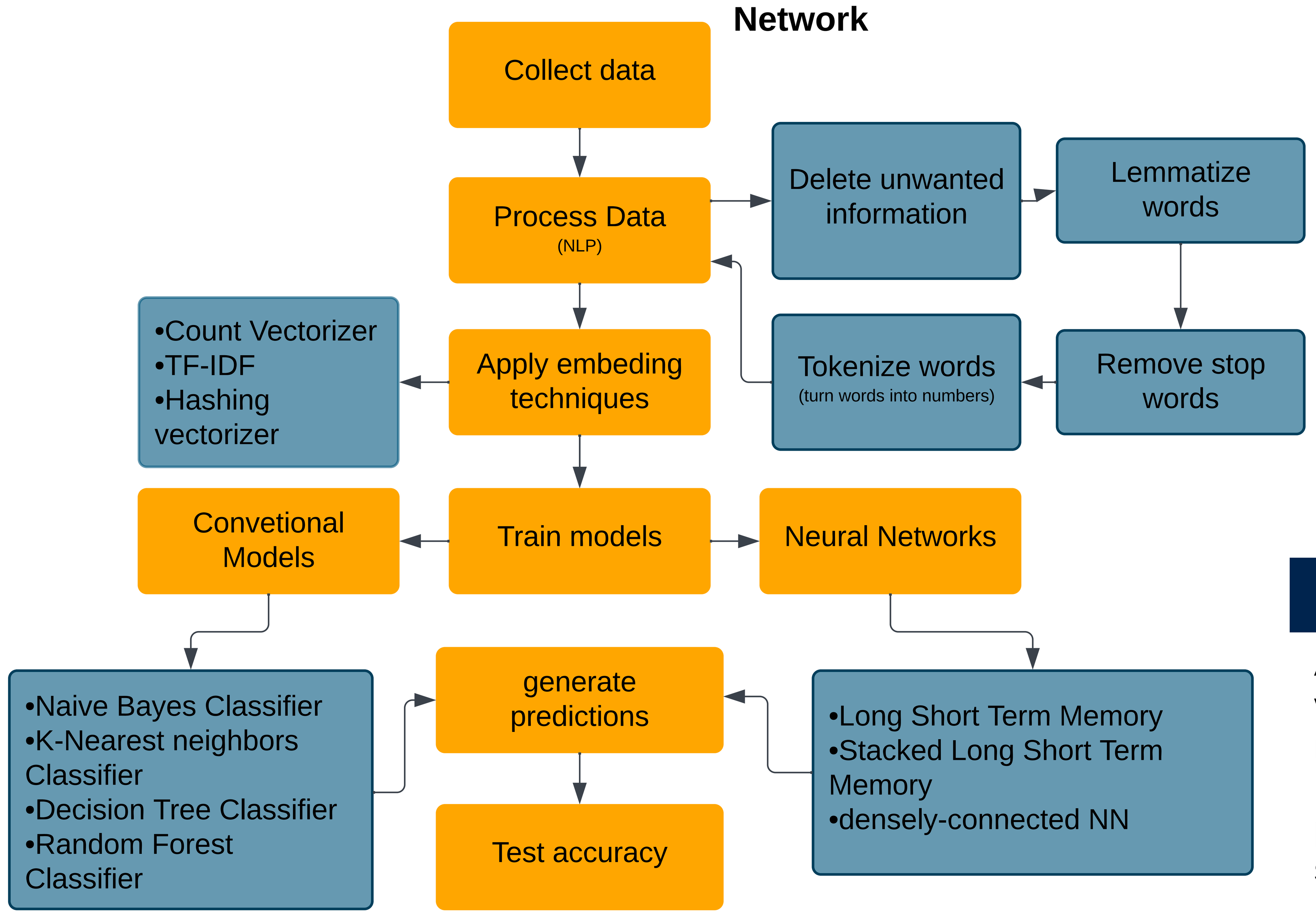

Figure 1:
The NLP process involves data collection, preprocessing, vectorization, model training (conventional models and neural networks), prediction, and accuracy testing.

Results

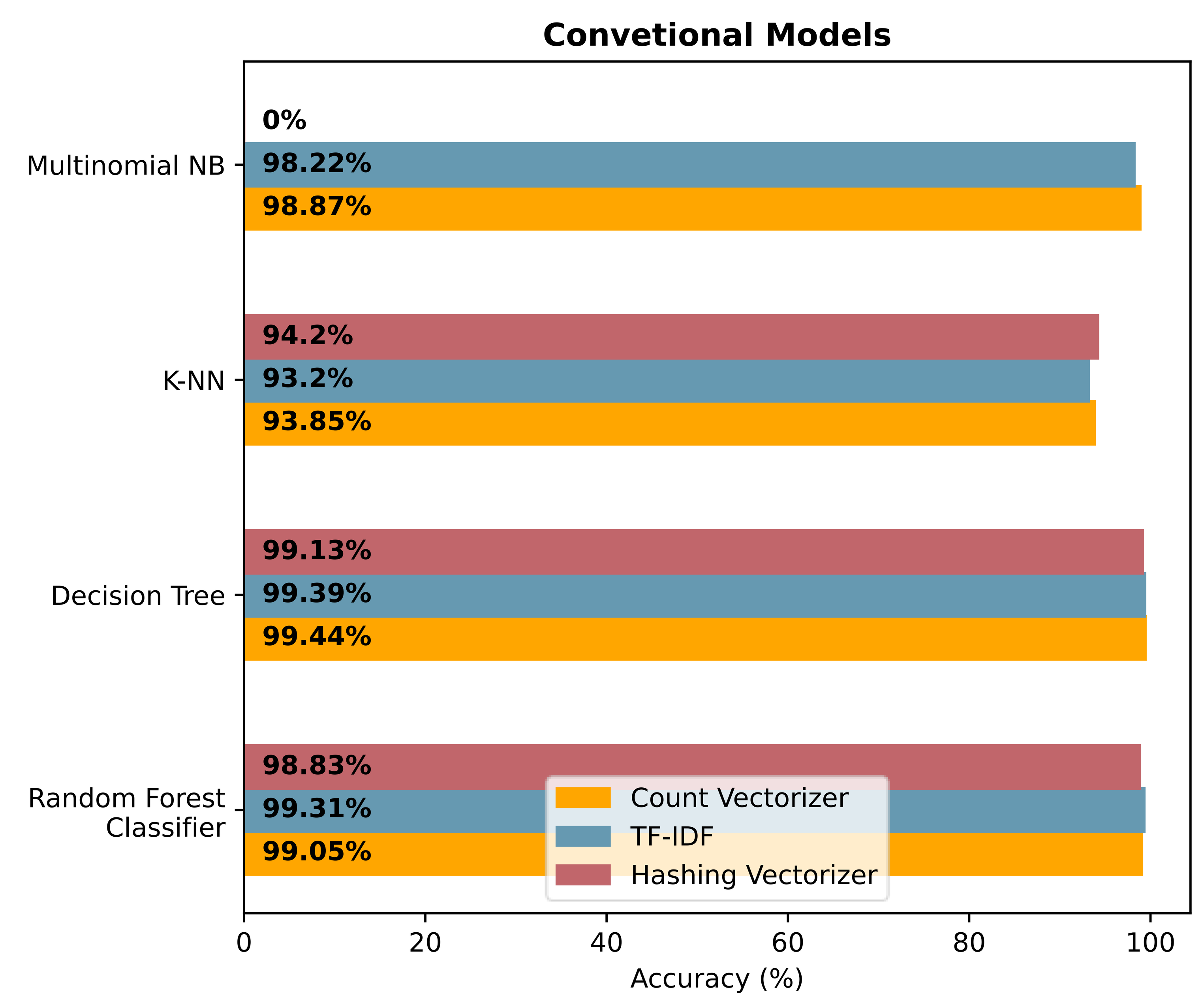

Figure 2:
Comparison of accuracy percentages of conventional machine learning algorithms with their respective vectorization or embedding techniques

Results (continued)

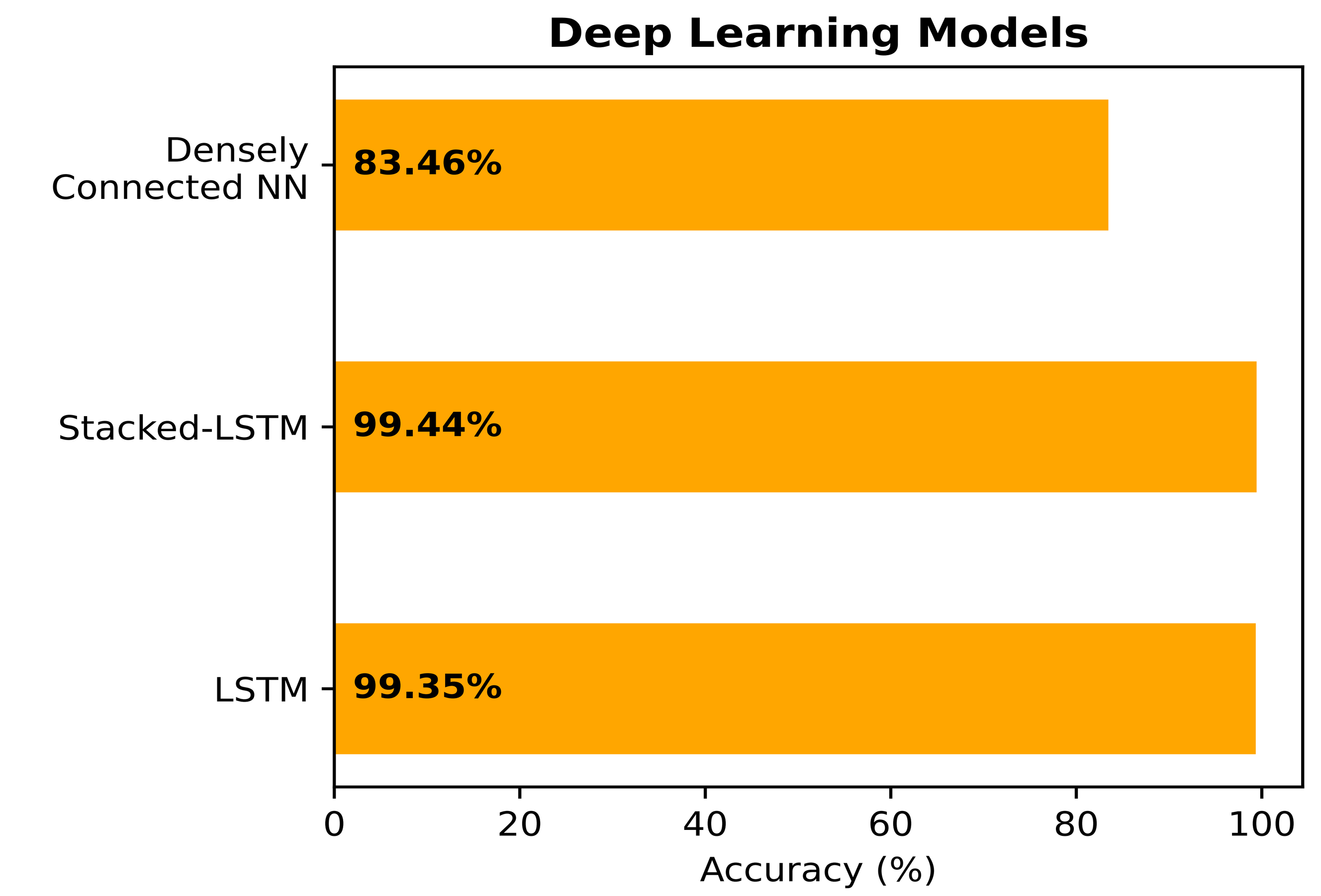

Figure 3:
Comparison of accuracy percentages of Deep Learning algorithms

Conclusion & Future Work

After conducting multiple experiments as part of our research, we employed various methods to assess the accuracy of different models used to classify messages as spam or ham. The results indicated that models such as Naive Bayes, k-nearest neighbors, and the Densely connected neural network did not perform up to the desired standard. However, the random forest classifier showed relatively lower proficiency when using a specific embedding technique. In comparison with the other models, it is possible to say that the embedding technique impacted the random forest. Nevertheless, all models exhibited similar accuracy levels when using any embedding technique. 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 models that performed the best were the Stacked LSTM and the decision tree in conjunction with the Count Vectorizer technique. These results suggest that an LSTM could be the better option for more complex problems. However, the decision tree model demonstrated the same accuracy as the Stacked LSTM and also boasted faster compilation time and simpler implementation. This suggests that in simpler problems, such as the one discussed in this research, a conventional model could be more practical for real-world implementation. 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 analyze its effects.

References

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.

Acknowledgements

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