Project X
SMART-MAIL-GUARD

Amal Verma, Kevin Shah, Rupak R. Gupta

Contents

Overview

SmartMailGuard is a system designed to categorize emails using Naïve Bayes, LSTM, and other Transformer architectures.

Using these different models and algorithms we can compare and grade their effectiveness on datasets of varying sizes and on the type of classification: Binary (Spam/Not-Spam) or Multiclass.

Acknowledgements

We would like to thank the Community of Coders (CoC) and Project X to provide us with such an opportunity to learn in the field of deep learning.

We would also like to thank the mentors for guiding us at every step of this long project and all the co-mentees that contributed in the project.

Naive Bayes

Naive Bayes is a simple and efficient **probabilistic classifier** based on Bayes' theorem, which assumes that all features are **independent** of each other given the class label. Despite this naive assumption, it works surprisingly well in many real-world scenarios, especially in **text classification** tasks.

The model calculates the probability of a class C given features X using Bayes' theorem:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where:

- P(C∣X) is the **posterior probability**.
- P(X∣C) is the **likelihood**.
- P(C) is the **prior probability** of the class.
- P(X) is the probability of the data.

There are different types of Naive Bayes classifiers, such as **Gaussian**, **Multinomial**, and **Bernoulli** Naive Bayes, each suited for different types of data (continuous, categorical, or binary).

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are designed to handle sequential data by incorporating memory of previous inputs. Unlike traditional feedforward networks, RNNs maintain a hidden state that is passed along through time steps. This makes them well-suited for tasks where order matters, like time series analysis or natural language processing.

In RNNs, the hidden state hth_tht at time step t is computed as:

$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$

This allows RNNs to process sequences of arbitrary length, but they can struggle with long-term dependencies due to issues like the **vanishing gradient problem**. Variants like **LSTM** and **GRU** solve this by introducing gating mechanisms to better handle long-range dependencies.

Long Short-Term Memory (LSTMs)

Long Short-Term Memory (LSTMs) are a type of Recurrent Neural Network (RNN) designed to address the **vanishing gradient problem**, which makes it difficult for standard RNNs to learn long-term dependencies. LSTMs introduce a series of **gates** (input, forget, and output) that control the flow of information, allowing the network to retain or forget information as needed.

The core idea is the **cell state** Ct, which carries information across different time steps. The gates regulate the flow of data:

- The **forget gate** decides what information to discard.
- The **input gate** controls what new information is added.
- The **output gate** determines the output based on the current cell state.

This gated structure makes LSTMs highly effective for tasks like **language modeling**, **speech recognition**, and **time series forecasting**, where long-range dependencies are important.

Multinomial Naive Bayes

Multinomial Naive Bayes is a variant of the Naive Bayes classifier, particularly effective for **text classification** tasks, where the features represent **counts** or **frequencies** of words. It assumes that the features (e.g., word occurrences) follow a multinomial distribution.

The classifier calculates the probability of each class based on the frequency of words in a document. The likelihood $P(x_i|C)$ is estimated as the probability of word xi occurring in class C, given by:

$$P(x_i|C) = \frac{\text{count}(x_i, C) + 1}{\sum_x \text{count}(x, C) + |V|}$$

Here, **Laplace smoothing** is applied to handle words that may not appear in training data. This model is widely used for **spam detection**, **sentiment analysis**, and other text-related tasks.

Bidirectional Encoder Representations from Transformers (BERT)

BERT is a state-of-the-art transformer-based model introduced by Google, specifically designed for various natural language processing (NLP) tasks. It processes text in a **bidirectional** manner, analyzing the context of each word by considering both its preceding and following words. This bidirectional capability allows BERT to capture complex relationships in language, improving its understanding of meaning and context.

BERT is pre-trained on two primary tasks:

- **Masked Language Modeling (MLM)**: Random words in sentences are masked, and the model learns to predict these masked words based on the surrounding context.
- **Next Sentence Prediction (NSP)**: This task involves determining whether one sentence logically follows another, which aids in understanding sentence relationships.

After pre-training, BERT can be fine-tuned for specific applications, such as:

- **Text Classification**: Effective for tasks like sentiment analysis and spam detection.
- **Question Answering**: Identifies answers from given passages, achieving high accuracy in QA tasks.
- **Named Entity Recognition (NER)**: Efficiently identifies and classifies key entities in text.

BERT's unique architecture and training approach have set new benchmarks in several NLP tasks, making it a foundational model in the field.

Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are supervised learning algorithms used for classification and regression tasks. They work by finding the **optimal hyperplane** that separates different classes in the feature space while maximizing the **margin** between the closest points of each class, known as **support vectors**.

SVMs can handle both linear and non-linear classification using the **kernel trick**, which transforms the input data into a higher-dimensional space to find an optimal separation. They are widely used in applications like text classification, image recognition, and bioinformatics due to their effectiveness in high-dimensional spaces.

Decision Trees

Decision Trees are a supervised learning algorithm used for classification and regression tasks. They create a tree-like model where:

- **Internal Nodes**: Represent features.
- **Branches**: Represent decision rules.
- **Leaf Nodes**: Represent outcomes.

The tree is constructed by splitting the dataset based on feature values to create homogeneous subsets. Decision trees are easy to interpret and can handle non-linear relationships and missing values. However, they can be prone to overfitting, which can be mitigated through techniques like **pruning**.

Random Forest

Random Forest is an ensemble learning method used for classification and regression tasks. It constructs multiple decision trees using random subsets of the data, which enhances predictive accuracy and reduces overfitting. The model uses bootstrapped samples (random sampling with replacement) to create each tree.

Predictions are made by aggregating the results from all trees: in classification tasks, the final prediction is based on the majority vote, while in regression tasks, it takes the average of the predictions. Random Forest also provides insights into feature importance, helping to identify the most influential variables. It is widely valued for its accuracy and effectiveness in handling complex datasets.

Future scope

1. Increasing the size of dataset

2. Explore more advanced transformer models (like RoBERTa)
3. Building a chrome extension
4. Including more categories to multi-intent classification
5. Multi-Language Support

References and Resources

1. Course on *Deep Learning Specialization* by *DeepLearning.AI*
2. *Long Short-Term Memory*
3. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
4. *Attention is all you need*
5. Kaggle datasets
6. HuggingFace Transformer Models