Spam Email Classifier- Progress Report

TIDJANI ADENIRAN

1. Introduction

This progress report outlines the initial steps taken towards developing a spam email classification system. The task is to accurately identify unsolicited and unwanted emails ("spam") from legitimate emails ("ham") based on their textual content. This is a common problem in natural language processing and machine learning, with significant practical applications in email filtering and security. The project aims to build a reliable classifier that can process incoming emails and determine their likelihood of being spam.

2. Proposed Model and Algorithm: Multinomial Naive Bayes with TF-IDF

For this spam classification task, we propose utilizing a **Multinomial Naive Bayes (MNB)** model in conjunction with **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorization. This combination is a widely used and effective baseline approach for text classification problems, known for its simplicity and efficiency.

2.1. Model Description: Multinomial Naive Bayes

The Multinomial Naive Bayes model is a probabilistic classifier based on Bayes' Theorem. It assumes that the features (in our case, the word frequencies or TF-IDF scores) are independent of each other given the class label (spam or ham). While this independence assumption is often violated in real-world text data (words are not truly independent), the model still performs surprisingly well in practice.

The core idea is to calculate the probability of an email belonging to a specific class (spam or ham) given its features. The model learns the probability of each word appearing in spam emails and in ham emails during the training phase. When classifying a new email, it uses these learned probabilities to determine which class the email is most likely to belong to.

Mathematically, the MNB classifier aims to find the class C (spam or ham) that maximizes the posterior probability P(C | features). Using Bayes' Theorem, this can be expressed as:

P(C | features) is proportional to P(features | C) * P(C)

Where:

- P(C | features) is the posterior probability of the class given the features.
- P(features | C) is the likelihood of the features given the class.
- P(C) is the prior probability of the class.

The "Naive Bayes" assumption simplifies P(features | C) by assuming the features are independent:

P(features | C) = P(f1 | C) * P(f2 | C) * ... * P(fn | C)

Where f_i are the individual features (e.g., TF-IDF scores of words).

For Multinomial Naive Bayes, the features are typically word counts or, in our case, TF-IDF scores. The model learns P(f_i | C), the probability of observing a specific feature value (or word's TF-IDF score) given the class.

2.2. Algorithm: TF-IDF Vectorization

Before applying the MNB model, the email text needs to be converted into a numerical representation. TF-IDF vectorization achieves this by calculating a score for each word in each document (email) that reflects its importance within that document relative to its importance in the entire collection of documents (dataset).

The TF-IDF score for a term 't' in a document 'd' within a corpus 'D' is calculated as:

TF-IDF(t, d, D) = TF(t, d) * IDF(t, D)

Where:

- **Term Frequency (TF):** TF(t, d) is the number of times term 't' appears in document 'd'. Sometimes normalized by the total number of terms in the document.
- **Inverse Document Frequency (IDF):** IDF(t, D) = log((Total number of documents in D) / (Number of documents in D containing term t + 1)). The "+1" is added to avoid division by zero for terms that appear in no documents. IDF downweights terms that appear frequently across many documents (like common English words) and highlights terms that are more unique to specific documents.

The TF-IDF vectorization process creates a matrix where each row represents an email, and each column represents a unique word (or n-gram) from the vocabulary. The values in the matrix are the TF-IDF scores.

2.3. Application to Spam Classification (Concrete Example)

Let's illustrate how this applies to our spam classification problem using a simplified example:

Dataset:

- Email 1 (Ham): "Hello, how are you doing today?"
- Email 2 (Spam): "Win a FREE prize! Click here now."
- Email 3 (Ham): "I hope you are doing well."
- Email 4 (Spam): "Claim your FREE gift! Visit our site."

Step 1: Text Cleaning

- Convert to lowercase, remove punctuation, etc.
    - Email 1: "hello how are you doing today"
    - Email 2: "win a free prize click here now"
    - Email 3: "i hope you are doing well"
    - Email 4: "claim your free gift visit our site"

Step 2: TF-IDF Vectorization

- Assume a vocabulary of important words after removing stop words (like "a", "are", "you"): {"hello", "doing", "today", "win", "free", "prize", "click", "here", "now", "hope", "well", "claim", "gift", "visit", "site"}
- Calculate TF-IDF scores for each word in each email. For example, let's look at the word "free":
    - "free" appears in Email 2 and Email 4.
    - TF("free", Email 2) = 1
    - TF("free", Email 4) = 1
    - Total documents = 4
    - Documents containing "free" = 2
    - IDF("free") = log(4 / (2 + 1)) = log(4/3) approximately 0.28
    - TF-IDF("free", Email 2) = 1 * 0.28 = 0.28
    - TF-IDF("free", Email 4) = 1 * 0.28 = 0.28
- Repeat this for all words in the vocabulary to create a TF-IDF matrix.

Step 3: Multinomial Naive Bayes Training

- The MNB model learns the probabilities of the TF-IDF scores given the class.
- For "spam", the model would see that words like "free", "prize", "click", "claim", "gift" have higher TF-IDF scores in spam emails.
- For "ham", words like "hello", "doing", "hope", "well" have higher TF-IDF scores.
- The model also learns the prior probabilities of spam and ham based on the dataset's class distribution.

Step 4: Classifying a New Email

- New Email: "Claim your prize now!"
- **Cleaning:** "claim your prize now"
- **Vectorization:** Calculate TF-IDF scores for the words in this new email based on the vocabulary and IDF values learned from the training data.
- **Classification:** The MNB model calculates:
    - P("claim your prize now" | Spam)
    - P("claim your prize now" | Ham)
- Based on the learned probabilities from the training data, words like "claim", "prize", and "now" are more strongly associated with the "Spam" class. Therefore, P("claim your prize now" | Spam) is likely to be higher than P("claim your prize now" | Ham).
- The model would then classify this email as "Spam".

Variables, Factors, and States:

- **Variables/Features:** The TF-IDF scores of the words (or n-grams) present in the email text. We also included the email length as an additional feature.
- **Factors:** The presence and frequency of specific words and their importance (as captured by TF-IDF) are the key factors influencing the classification. The prior probability of the classes (spam vs. ham) is also a factor.
- **States:** The two possible classification outcomes are "Spam" and "Ham".

3. Implementation Status

The baseline implementation of the spam classification system using Multinomial Naive Bayes and TF-IDF vectorization has been completed. The current implementation includes the following:

- **Data Loading:** Loading the "Book1.csv" dataset into a pandas DataFrame. Handling different encodings (latin-1) was necessary.
- **Data Cleaning:** Implemented steps for handling missing values, removing duplicates, converting text to lowercase, removing HTML tags, punctuation, and non-ASCII characters.
- **Data Preparation:** Utilized TfidfVectorizer from scikit-learn to convert the cleaned text into a TF-IDF matrix. Included the email length as an additional numerical feature and combined it with the TF-IDF features using hstack.
- **Train/Test Split:** Split the data into 80% for training and 20% for testing using train_test_split.
- **Model Training:** Trained a MultinomialNB model on the training data.
- **Model Optimization (Preliminary):** A basic loop was implemented to evaluate the model's accuracy with different alpha values (0.01, 0.1, 0.5, 1.0, 2.0) to identify a potentially better smoothing parameter. The code then uses the best-performing alpha (found to be 0.5 in preliminary tests) for the final model training and evaluation.
- **Model Evaluation:** Calculated and displayed the accuracy, classification report (including precision and recall), and confusion matrix on the test set.
- **User Input Feature:** Added a function and a loop to allow users to input email text and get a real-time classification from the trained model.

The implementation is functional and represents a solid baseline. It is not yet fully optimized and lacks some advanced features that could potentially improve performance (e.g., more sophisticated feature engineering, hyperparameter tuning for the TF-IDF vectorizer).

4. Initial Experimental Results

Initial experiments were conducted using the implemented baseline model. The evaluation was performed on the 20% held-out test set. The results are as follows:

- **Best Alpha Value Found:** 0.5
- **Accuracy on Test Set (with alpha=0.5):** 0.9857
- **Precision (Spam):** 1.0000
- **Recall (Spam):** 0.90

Confusion Matrix:

	Predicted Ham	**Predicted Spam**
Actual Ham	950	5
Actual Spam	14	146

Interpretation of Initial Results:

The initial results are promising. The model achieved an accuracy of **0.9830,** indicating that a high percentage of emails in the test set were correctly classified. The high precision for spam **0.9669** suggests that when the model predicts an email is spam, it is very likely to be true spam, which is important to avoid incorrectly filtering legitimate emails.

However, the recall for spam **0.9125** is lower than precision. This means that the model is missing some of the actual spam emails (resulting in false negatives). The confusion matrix clearly shows the trade-off: a low number of false positives (151) but a higher number of false negatives (964). This suggests that the model is more conservative in classifying emails as spam.

The inclusion of email length as a feature might have contributed positively, but further analysis is needed to confirm its impact. The preliminary alpha optimization helped identify a better smoothing parameter compared to the default.

5. Next Steps

Based on the initial results and analysis, the following steps are planned for the next phase of the project:

- **Further Feature Engineering:** Explore additional text-based features that could help differentiate spam from ham, such as the presence of specific spam trigger words, the use of excessive capitalization or exclamation marks, or domain-specific features.

Advanced Hyperparameter Tuning: Perform more comprehensive hyperparameter tuning for both the `TfidfVectorizer` (e.g., `min_df`, `max_df`, `ngram_range`) and the `MultinomialNB` model (`alpha`) using techniques like GridSearchCV or RandomizedSearchCV to find the optimal combination of parameters.

- **Handling Class Imbalance:** Investigate techniques to address the class imbalance in the dataset, such as oversampling the minority class (spam) or undersampling the majority class (ham). This could potentially improve the model's ability to detect spam.
- **Explore Other Models:** While MNB is a strong baseline, explore other classification algorithms like Logistic Regression or Support Vector Machines (SVMs) to see if they yield better performance.
- **Error Analysis:** Conduct a more in-depth analysis of the false negatives (spam emails misclassified as ham) to understand the characteristics of these emails and identify potential areas for improvement in preprocessing or feature engineering.

6. Conclusion

The initial implementation of the spam email classifier using Multinomial Naive Bayes and TF-IDF has provided a solid baseline and promising initial results. The model demonstrates good precision in identifying spam but needs improvement in recall to capture more spam emails. The next steps will focus on enhancing feature engineering, optimizing hyperparameters, addressing class imbalance, and exploring alternative models to further improve the classifier's performance. The current implementation also successfully incorporates the user input feature, allowing for interactive classification.