Name: Prasun Abhishek S Singh

SUID: 418095794

­­­­IST 664: Natural Language Processing

# Final Project Detection of SPAM in email

**Introduction**:

The Project utilizes machine learning and natural language processing (NLP) to effectively differentiate between spam and non-spam (ham) emails. In addressing the widespread issue of spam emails, the project plays a crucial role in enhancing email security and optimizing user interactions with their email platforms. Employing NLP techniques enables the thorough analysis and interpretation of the complexities inherent in human language within emails. This analytical approach aids in the identification and filtering of undesired spam, ultimately improving overall communication efficiency and providing users with a more secure and streamlined email experience.

The dataset employed in this project is sourced from the Enron public email corpus, where additional spam emails were introduced to augment the existing examples. Non-spam emails are denoted as "ham." The project's core objective is to develop a robust classifier capable of accurately categorizing emails, contributing to ongoing endeavors to address the challenges presented by spam in the digital communication landscape. About the dataset it’s Enron spam dataset available at (http://www.aueb.gr/users/ion/data/enron-spam/).

STEP 1: Text Processing  
The choice of dataset for me was “EmailSpamCorpora” from the given datasets.

In the initial phase of the Email Spam Classification Project, the primary focus lies in text processing a very important step in readying the dataset for subsequent machine learning processes.

Text Processing Steps:

1. Tokenization of Text: The initial step involves breaking down email text into individual tokens. Tokenization is vital for converting raw text into a suitable format for subsequent analysis and feature extraction.

2. Pre-processing and Filtering: Depending on our selected dataset's specific characteristics, these are the pre-processing or filtering steps may be implemented which includes techniques like:

- Porter Stemming: Reducing words to their root form using the Porter stemming algorithm.

- Stopword Removal: Eliminating common words (stopwords) that do not significantly contribute to the classification task.

- Lemmatization: Reducing words to their base or dictionary form to enhance feature extraction.

Also cleaning of the dataset.  
  
STEP 2: Feature Engineering  
In pursuit of the Detection of spam in emails, this project's goal is to differentiate between spam and non-spam (ham) emails, the feature engineering process adopts the NLTK notation and utilizes Python feature functions, following the methodology demonstrated in NLTK labs. The foundational feature set commences with "bag-of-words" features, representing the presence or absence of specific words in an email. This approach transforms unstructured textual data into a formatted structure suitable for machine learning.

The NLTK Naïve Bayes classifier is employed to train and test the feature sets. Adhering to NLTK conventions, feature functions are crafted in Python, encompassing the essence of the bag-of-words representation. Functions like `document\_features`, `document\_features\_ps`, `document\_features\_wc`, and `document\_features\_lem` incorporate diverse feature extraction strategies, including Porter stemming, stopword elimination, and lemmatization. The NLTK Naïve Bayes classifier is harnessed to assess the efficacy of these features in discerning between spam and ham emails.

To comprehensively evaluate the classifier's performance, cross-validation is applied, providing a thorough examination of its precision, recall, and F-measure scores. This systematic validation process ensures the model's robustness and its ability to generalize to new data. By integrating NLTK's feature notation and classifier, the project aims to develop a well-informed and efficient spam detection model, contributing to the broader objective of enhancing email security and user experience.  
  
STEP 3: Experiments  
The experiments conducted in the project encompass the assessment of spam classification using a variety of feature sets. Primary experiments involve:

1. Base Level Experiments:

- Comparing different feature sets to establish a baseline.

- Utilizing at least two feature functions: one combining features from labs and another introducing a new approach.

- Training and testing the NLTK Naïve Bayes classifier and other methods.

- Evaluating performance metrics such as accuracy, precision, recall, and F-measure.

- Employing cross-validation for a robust evaluation.

2. Feature Function Exploration:

- Exploring advanced feature functions beyond lab examples.

- Investigating techniques like word embeddings or semantic analysis.

These experiments aim to pinpoint optimal feature sets for spam classification, improving the model's effectiveness and providing insights into email security.  
  
**Implementation:**

Project Implementation Overview:

- Data Preparation: The code processes email data from the Enron public email corpus, introducing additional spam instances for training the classifier.

- Text Handling: Tokenization, stemming, stopword removal, and lemmatization are employed for text representation.

- Feature Creation: Feature functions like `document\_features`, `document\_features\_ps`, `document\_features\_wc`, and `document\_features\_lem` are crafted to generate feature sets.

- Classifier Training: The primary model is the NLTK Naïve Bayes classifier, extensively trained and tested.

- Model Assessment: The implemented models undergo thorough evaluation, including precision, recall, F-measure, and accuracy scores.

- Model Variety: While supporting SVM and Random Forest, the focus remains on experimenting with the NLTK Naïve Bayes classifier.

Model Types:

1. NLTK Naïve Bayes Classifier:

- Central for spam detection, trained and tested with diverse feature sets.

- Evaluated using precision, recall, F-measure, and accuracy metrics.

2. Multinomial Naive Bayes with Porter Stemmer:

- Applies NLTK Naïve Bayes with porter stemming.

- Trained and evaluated with this stemming feature.

3. Multinomial Naive Bayes with Word Count and Porter Stemmer:

- Explores word count and porter stemming with Naïve Bayes.

- Trained and evaluated with this combined feature set.

4. Multinomial Naive Bayes with Lemmatizer:

- Implements lemmatization for Naïve Bayes.

- Trained and evaluated using lemmatized features.

5. SVM with Porter Stemmer:

- Provides an alternative with SVM and porter stemming.

- Trained and evaluated using this SVM configuration.

6. Random Forest Classifier:

- Offers an alternative using the Random Forest algorithm.

- Trained and evaluated with the Random Forest classifier.

The project prioritizes not only training models but also rigorously evaluating them through diverse metrics, ensuring the selection of effective classifiers for precise spam detection.  
  
**Model Performance:**1. NLTK Naïve Bayes Classifier:

A close-up of a number

Description automatically generated

2. Multinomial Naive Bayes with Porter Stemmer:

A close-up of a number

Description automatically generated

3. Multinomial Naive Bayes with Word Count and Porter Stemmer:

A black text with numbers

Description automatically generated

4. Multinomial Naive Bayes with Lemmatizer:

A black text on a white background

Description automatically generated

5. SVM with Porter Stemmer:

A number with numbers and circles

Description automatically generated with medium confidence

6. Random Forest Classifier:

A number with black numbers

Description automatically generated with medium confidence  
Model-Specific Results:

1. Multinomial Naive Bayes: This model demonstrates its effectiveness as a baseline classifier for spam detection.
2. Naive Bayes with Porter Stemmer: The inclusion of the Porter Stemmer in feature processing contributes to improved classification performance.
3. Naive Bayes with Word Count and Porter Stemmer: Utilizing word count alongside the Porter Stemmer showcases the impact of feature engineering on classification outcomes.
4. Naive Bayes with Lemmatizer: The lemmatization approach aids in capturing the root forms of words, potentially enhancing the model's linguistic understanding.
5. SVM with Porter Stemmer: The Support Vector Machine with Porter Stemmer showcases the performance of a different classification algorithm, providing diverse insights into model selection.
6. Random Forest Classifier: This model demonstrates its effectiveness in handling complex feature sets, potentially offering robust spam classification capabilities.

Above all, Multinomial Naive Bayes with Lemmatizer had the highest accuracy.

**Key Considerations for future work:**  
1. Advanced Feature Engineering: Future work can focus on exploring more sophisticated feature engineering techniques, such as sentiment analysis, domain-specific features, or deep learning-based embeddings. Enhancing the richness of features can contribute to a more nuanced understanding of email content.

2. Data Processing Optimization: Investigate ways to optimize the data processing pipeline, considering scalability and efficiency. Techniques like parallel processing or distributed computing can be explored to handle larger datasets and improve overall processing speed.

3. Deep Learning Models: Incorporate deep learning models, such as recurrent neural networks (RNNs) or transformer architectures, for improved natural language understanding. Deep learning can capture intricate patterns in email text and potentially outperform traditional machine learning approaches.

4. Transfer Learning: Explore transfer learning techniques, leveraging pre-trained models on large text corpora, to enhance the model's ability to generalize across different domains or adapt quickly to new trends in spam tactics.

5. Interdisciplinary Collaboration: Foster collaboration with experts in cybersecurity and linguistics to gain deeper insights into evolving spam techniques and linguistic nuances. This interdisciplinary approach can lead to more effective feature design and model development.

6. Explainability and Interpretability: Address the challenge of making deep learning models more interpretable. Developing methods to explain model predictions can enhance user trust and facilitate the identification of potential biases or limitations in the classification system.

7. Dynamic Adaptation: Implement mechanisms for the model to dynamically adapt to changes in spam tactics over time. Continuous monitoring and updating of the model parameters based on real-time feedback can ensure its relevance and effectiveness in an ever-evolving email landscape.  
  
**Conclusion:**  
This project effectively employs machine learning methodologies to categorize emails into spam and ham groups. It assesses the efficiency of SVM, Random Forest, and Naive Bayes models in text classification, providing valuable insights. The experimentation encompasses diverse feature sets and classifiers, including Multinomial Naive Bayes with and without stemming, word count, and lemmatization. Additionally, it explores SVM with porter stemming and Random Forest classifiers.

The findings underscore the adaptability of different models and their accuracy based on feature engineering techniques. The project not only establishes a baseline through initial experiments but also investigates advanced feature functions. Utilizing precision, recall, F-measure, and accuracy metrics, coupled with cross-validation, ensures a comprehensive evaluation of model effectiveness.

In summary, this project makes a significant contribution to spam detection initiatives, presenting a versatile framework for experimentation and refinement. It sets the stage for future endeavors, emphasizing the significance of feature engineering, data processing, and the exploration of deep learning models to enhance email classification.

Code snapshots

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated