Yasser ISSAM, Christophe SAURY, Jung YSEOK

Introduction to Machine Learning Final Report

Problem Description:

In the era of information overload, effective organization and categorization of textual data are crucial for efficient information retrieval and comprehension. The problem at hand revolves around the classification of newsgroup articles into predefined categories, aiming to develop a robust model capable of automatically assigning topics to incoming texts. This project focuses on the application of machine learning techniques, specifically the Naive Bayes algorithm, to tackle the inherent challenges of text classification.

The selected dataset for this endeavor is the 20 newsgroups dataset, a diverse collection of documents covering a wide array of topics. This dataset poses a suitable challenge for a text classification model, given its variety of categories ranging from technology and science to religion and politics. The primary goal is to build a model that can accurately categorize new articles into these predefined groups, facilitating the efficient organization and retrieval of information within the dataset.

This problem is particularly relevant in contexts where large volumes of unstructured textual data are encountered, such as news aggregation platforms, discussion forums, and content recommendation systems. Successful text classification not only aids in organizing content but also enhances the user experience by providing targeted and relevant information.

By addressing the challenges inherent in the text classification of newsgroup articles, this project aims to improve our understanding of text classification methods. The insights gained from this endeavor can find applications in various domains where automated content categorization is essential for knowledge management and user engagement.

Description of the news group input:

The fetch_20newsgroups function from scikit-learn provides a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups. Each newsgroup represents a specific category or topic, covering diverse subjects such as technology, sports, politics, and more. The dataset is commonly used for text classification and natural language processing tasks. There are a total of 3'135'253 words in the dataset. The dataset includes words that are misspelled.

The structure of the dataset allows for supervised machine learning tasks, making it a suitable choice for training and evaluating text classification models. The articles within each newsgroup exhibit significant variability in writing style, content, and vocabulary, making it a challenging yet realistic benchmark for text categorization algorithms.

This dataset is widely utilized in the research community for benchmarking and comparing the performance of different text classification models. Its diversity and size make it an excellent resource for developing and testing robust text categorization techniques.

```
categories = ['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

Theory and method chosen:

Text classification is a fundamental problem in natural language processing (NLP) that involves assigning predefined categories or labels to textual documents. In the context of this project, the chosen method to address the text classification task is the Naive Bayes algorithm.

Naive Bayes Algorithm:

The Naive Bayes algorithm is a probabilistic machine learning algorithm based on Bayes' theorem. It is widely used for text classification tasks due to its simplicity, efficiency, and effectiveness, especially in scenarios with a relatively small amount of data. Despite its "naive" assumption of independence between features, it often performs surprisingly well in practice.

Bayes' Theorem:

Bayes' theorem is a fundamental concept in probability theory that describes the probability of an event based on prior knowledge of conditions related to the event. The theorem is expressed as follows:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

In the context of text classification:

P(A|B) is the probability of a document belonging to category A given its features B.

P(B|A) is the likelihood of observing features B given that the document belongs to category A.

P(A) is the prior probability of a document belonging to category A.

P(B) is the probability of observing features B.

Application in Text Classification:

In text classification, the features (words or tokens) of a document are used to determine the probability of the document belonging to each predefined category. The "naive" assumption in Naive Bayes is that the presence or absence of a particular

feature is independent of the presence or absence of any other feature, given the category. The features of our documents here are the words that compose it.

Implementation:

The implementation of the text classification model using the Naive Bayes algorithm involves several key components:

Data Preprocessing:

The fetch_20newsgroups function from scikit-learn is utilized to obtain the 20 newsgroups dataset, which comprises approximately 20,000 newsgroup documents spread across 20 different categories. The dataset is then split into training and testing sets, ensuring the model is trained on one subset and evaluated on another.

Vocabulary Construction:

The TextCategorizer class contains a method construct_vocabulary that processes the training data to extract unique words and build a vocabulary. The vocabulary is a set of all unique words present in the training set and is crucial for training the model.

Training the Naive Bayes Model:

The train_model method of the TextCategorizer class is responsible for training the Naive Bayes model. For each category, the method calculates word counts using Laplace smoothing to handle unseen words. It computes probabilities for each word given a specific category and stores the information in the category_info dictionary. It does so by dividing the word counts of each word per category by the total amount of words.

Computing Log Probabilities:

The compute_log_probabilities method in the TextCategorizer class converts the word count probabilities into log probabilities. This is done to avoid numerical underflow when dealing with small probabilities.

Making Predictions:

The make_predictions method of the TextCategorizer class is responsible for predicting the categories of new articles in the test set. It uses the Naive Bayes algorithm to calculate log probabilities for each category and selects the category with the highest probability as the predicted label.

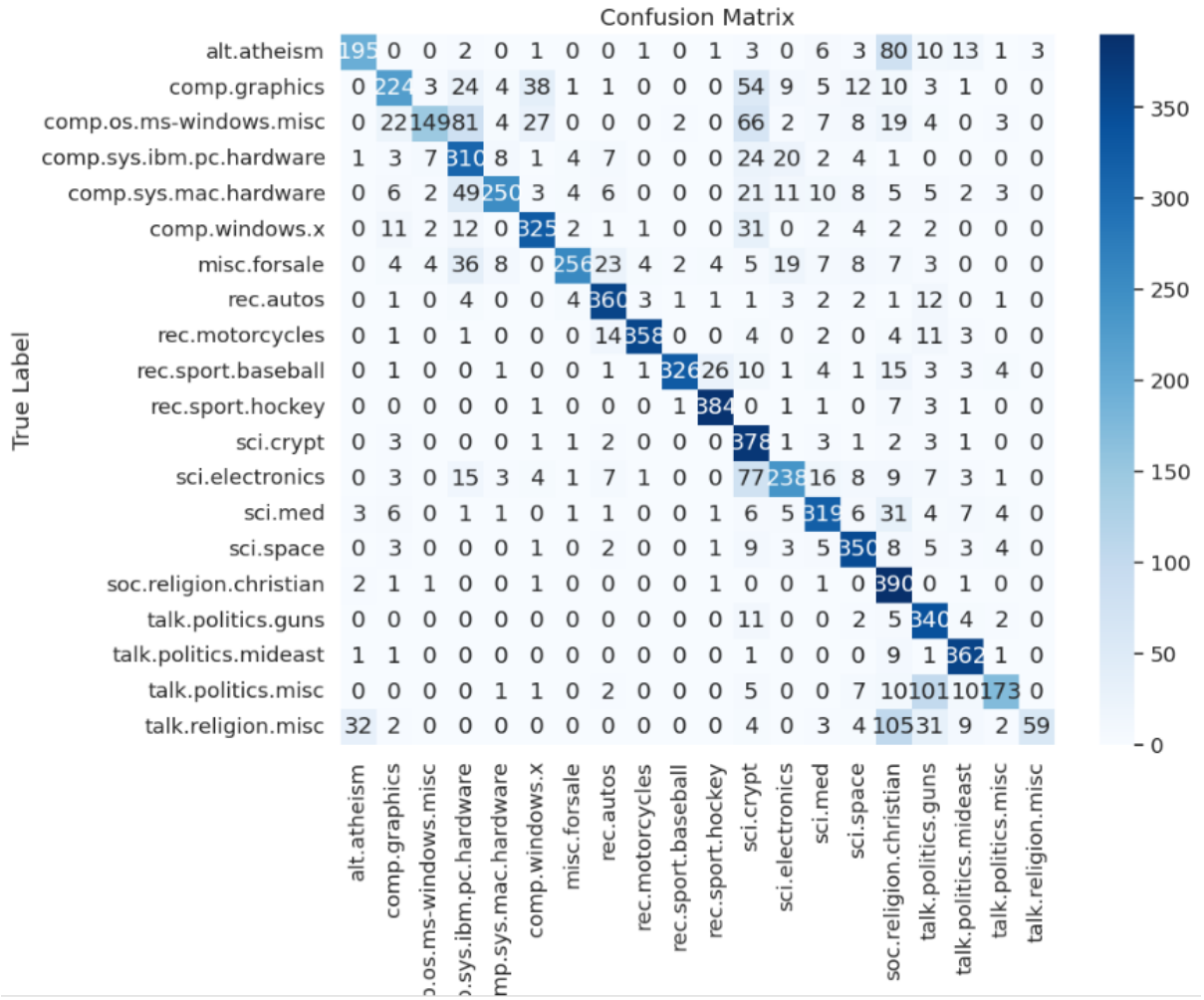
```
# Make Predictions
true_labels, predicted_labels = text_categorizer.make_predictions(test_data.data)
```

```
Sample 7475 - Predicted: talk.politics.mideast, True: talk.politics.mideast
Sample 7476 - Predicted: sci.space, True: sci.space
Sample 7477 - Predicted: comp.sys.mac.hardware, True: comp.sys.mac.hardware
Sample 7478 - Predicted: rec.autos, True: rec.autos
Sample 7479 - Predicted: sci.crypt, True: misc.forsale
Sample 7480 - Predicted: comp.sys.ibm.pc.hardware, True: misc.forsale
Sample 7481 - Predicted: comp.sys.ibm.pc.hardware, True: comp.sys.ibm.pc.hardware
Sample 7482 - Predicted: rec.sport.hockey, True: rec.sport.hockey
Sample 7483 - Predicted: misc.forsale, True: misc.forsale
Sample 7484 - Predicted: soc.religion.christian, True: alt.atheism
Sample 7485 - Predicted: rec.sport.baseball, True: rec.sport.baseball
Sample 7486 - Predicted: talk.politics.guns, True: talk.politics.guns
Sample 7487 - Predicted: rec.sport.hockey, True: rec.sport.hockey
Sample 7488 - Predicted: rec.sport.baseball, True: rec.sport.baseball
Sample 7489 - Predicted: sci.crypt, True: comp.graphics
```

Confusion Matrix Visualization:

The visualize_confusion_matrix method creates a confusion matrix using the confusion_matrix function from scikit-learn. This matrix helps in understanding the model's performance by displaying the distribution of predicted and true labels across different categories.

```
# Visualize Confusion Matrix
text_categorizer.visualize_confusion_matrix(true_labels, predicted_labels)
```



Encapsulation in a Class:

The TextCategorizer class encapsulates the entire functionality, promoting code modularity and reusability. The class design allows for easy experimentation with different datasets and provides a clear structure for text classification tasks.

Application of Laplace Smoothing:

Laplace smoothing is applied in the training phase (update_category_information method) to handle words that might not appear in the training set. This ensures that the model does not assign zero probability to unseen words, contributing to better generalization.

```
for document in data_for_category:
    words = set(document.lower().split())
    for word in words:
        total_words += 1
        word_count_category[word] += 1

for word in word_count_category:
    word_count_category[word] += 1

total_words_with_smoothing = len(self.unique_words_list) + total_words
for word in word_count_category:
    self.category_info[category]["word_count"][word] = word_count_category[word] / total_words_with_smoothing
```

We add +1 to the count of every word for the application of Laplace smoothing.

By breaking down the implementation into these components, the code achieves clarity, and the TextCategorizer class serves as a comprehensive tool for text classification tasks. This modular design facilitates future enhancements, such as incorporating different algorithms or addressing specific challenges related to text data.

Experimental results:

The model's performance is assessed using the test set from the 20 newsgroups dataset. The accuracy of the model on the test data is calculated, providing a quantitative measure of its effectiveness in correctly classifying new articles.

We obtain the following accuracy :

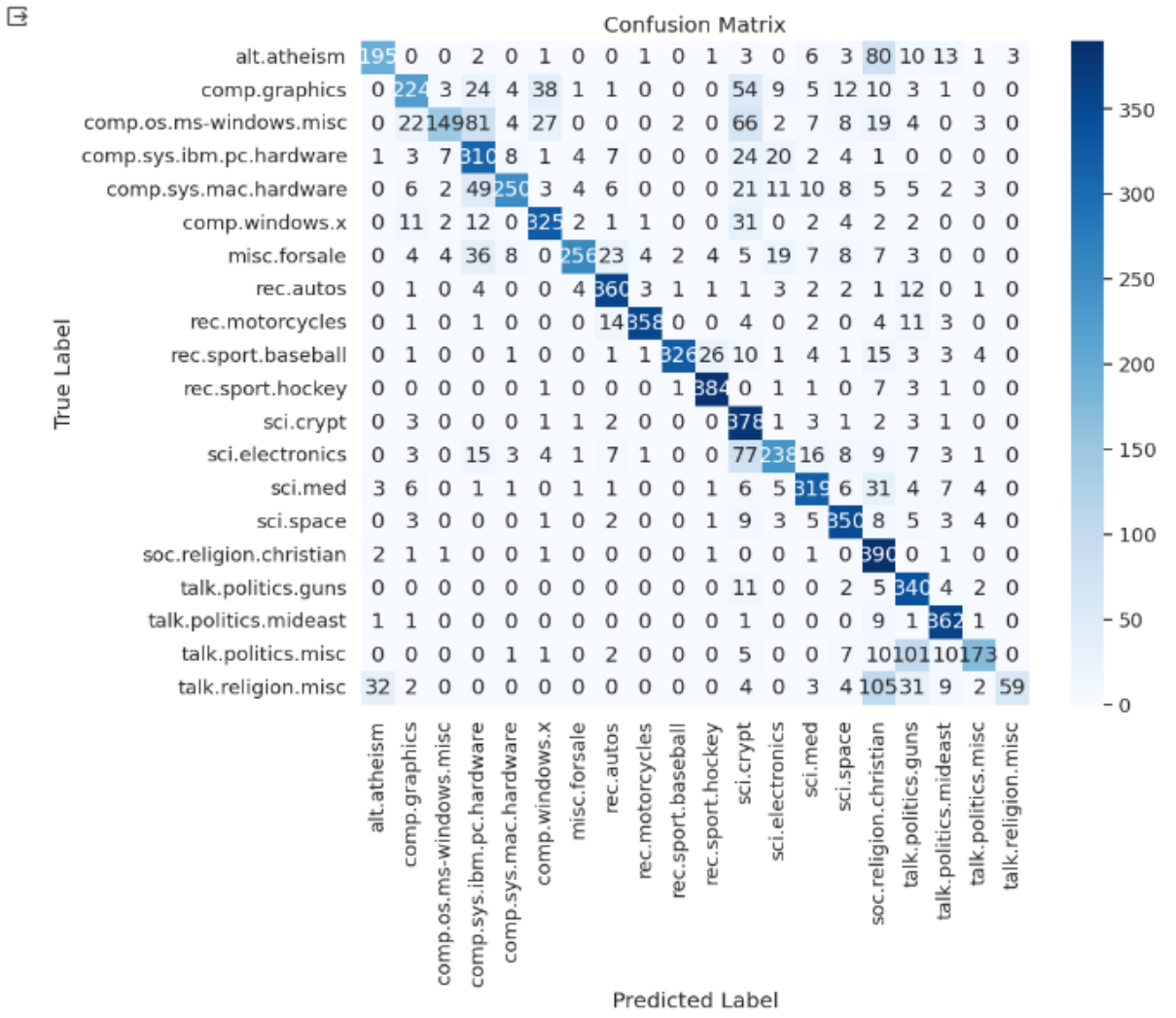
```
# Compute and Print Accuracy
accuracy = text_categorizer.compute_accuracy(true_labels, predicted_labels)
print("Test Accuracy(%):", accuracy)
```

```
Test Accuracy(%): 76.28783855549655
```

The model works quite well given that the accuracy is at 76%.

We use the confusion matrix to visualize the distribution of predicted and true labels across different categories. This matrix helps us identify potential areas of improvement by telling us for which categories the model is more or less accurate. It highlights the model's strengths and limitations which can help us improve the model.

This is the obtained confusion matrix from applying our model to the news groups texts from the scilearn library.



We can see from this confusion matrix that the model works well for most categories that have very unique vocabulary. The model has a harder time being accurate when applied to a category that shares similar vocabulary to another category.
We can see that the model is the least accurate when applied to the "talk.religion.misc" category, this can be explained by the fact that that category shares vocabulary with the "soc.religion.christian" and "alt.atheism" categories. Same goes for the categories linked to politics and the hardware related categories.

Extending the vocabulary list may help make the model more accurate.

Discussion:

The Naive Bayes algorithm demonstrates its efficacy in the context of text classification. Despite its "naive" assumption of independence between features, the algorithm performs well, particularly in scenarios with limited data. The Laplace smoothing technique proves valuable in handling unseen words and contributes to the model's generalization ability.

The diversity of the 20 newsgroups dataset poses a realistic challenge for the text classification model. The model must discern patterns across various topics, writing styles, and vocabularies. The insights gained from this project contribute to our understanding of text classification methods and their applicability in real-world scenarios.

Consideration is given to the impact of misspelled words in the dataset. While misspellings can pose challenges, the model's ability to handle them effectively showcases its robustness in the face of noisy data.

Conclusion:

In conclusion, this project addresses the critical problem of text classification using the Naive Bayes algorithm. The implementation demonstrates the algorithm's suitability for categorizing newsgroup articles into predefined topics. The model exhibits promising accuracy, showcasing its potential applicability in scenarios involving large volumes of unstructured textual data.

The project highlights the importance of effective text classification in managing and organizing information. The insights gained from this endeavor contribute to the broader field of natural language processing and machine learning, offering valuable lessons for future research and application in diverse domains.

The TextCategorizer class encapsulates the implemented solution, providing a modular and reusable framework for text classification tasks. Future work may involve exploring other algorithms, fine-tuning parameters, and considering more sophisticated approaches for handling misspelled words. Overall, this project serves as a foundation for continued exploration and improvement in the field of text classification and machine learning.