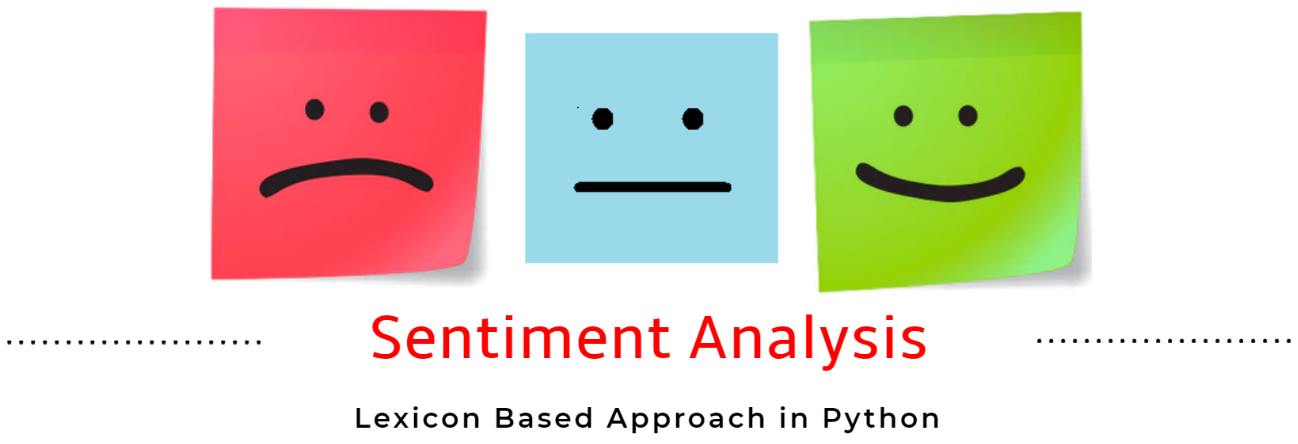
Natural Language Processing

Sentiment Analysis

Sentiment Analysis through POS Tagging: Analyzing Emotions in Text



Introduction

This report unveils a robust sentiment analysis system comprising a custom POS tagger and a TF-IDF based sentiment classifier with a Naive Bayes model. The fusion of these components maximizes sentiment understanding by incorporating both syntactic insights and discriminative features.

POS Tagging with Viterbi Algorithm

Our custom POS tagger, developed on the accessible Universal Tagset, employs the Viterbi algorithm—a smart tool that helps determine the most likely sequence of word labels in a sentence. It does this by considering the likelihood of a word belonging to a particular category (like noun, verb, etc.).

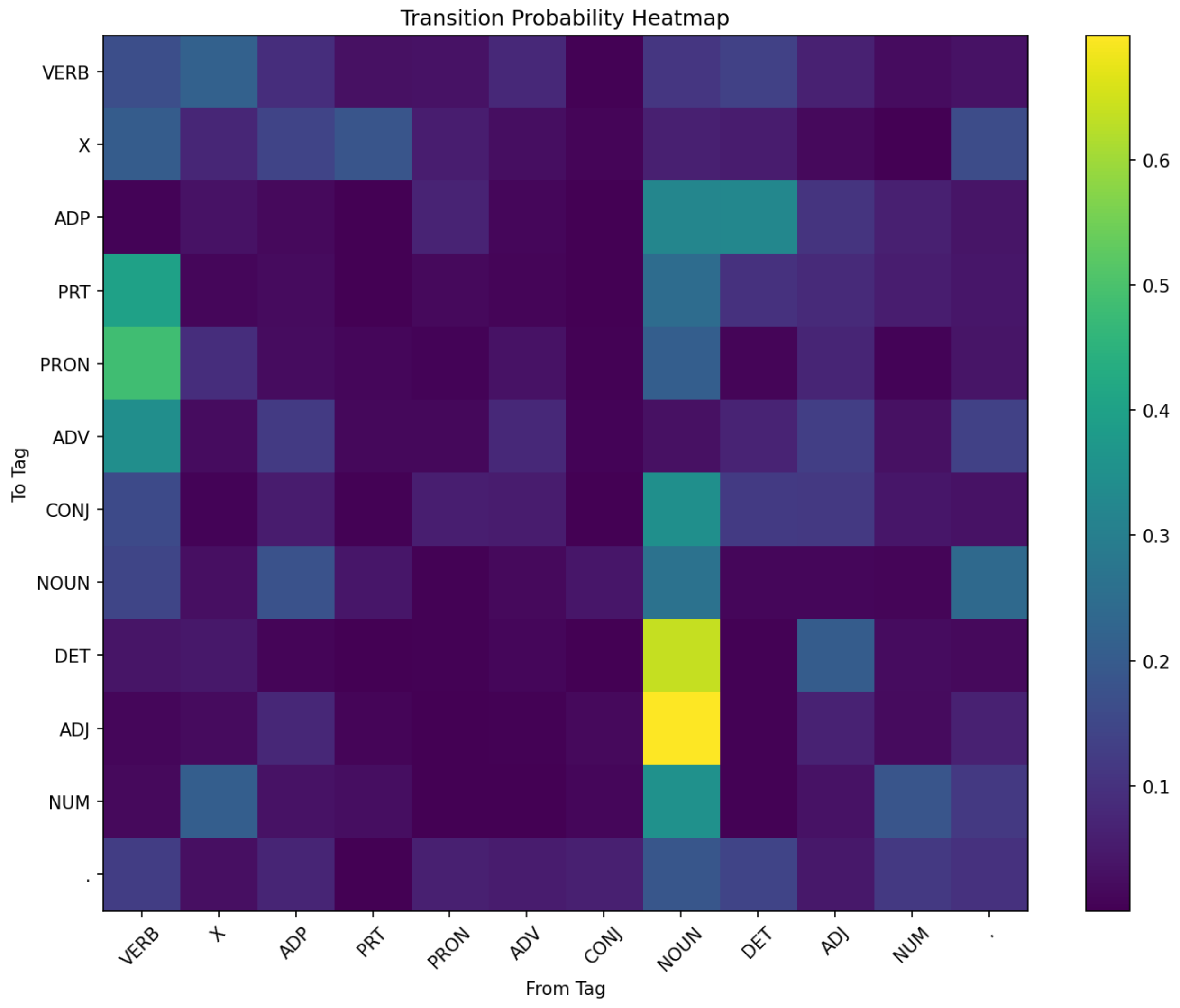
In simple terms, the tagger uses probabilities to make educated guesses about the words in a sentence. It looks at how likely it is for one word category to follow another, and also how likely a certain word is to belong to a specific category.

Additionally, the tagger takes into account the probability of a sentence starting with a particular word category. This way, it gets off to a good start when figuring out the tags.

Our tagger shows reliable accuracy, consistently ranging between 86% to 90%. This means it's quite good at understanding different kinds of sentences. In this section, we'll explain the Viterbi algorithm in a way that's easy to understand, and show how it plays a crucial role in our tagger.

Transition Probability Analysis

In this section, we examine transition probabilities between parts-of-speech tags derived from the Treebank corpus. The heatmap below illustrates these probabilities, shedding light on linguistic patterns captured by our model.



Conclusion on Viterbi-based POS Tagging

In conclusion, the implementation of the Viterbi algorithm in our custom POS tagger has proven to be highly effective. By considering transition probabilities, emission probabilities, and start probabilities, the tagger accurately assigns part-of-speech labels to words in a sentence. The heat distribution graphs further illustrate the robustness of our model.

Moving forward, let's transition to the next topic: the integration of TF-IDF and Naive Bayes classification for sentiment analysis. This dynamic combination capitalizes on both syntactic insights and discriminative features to enhance the accuracy of sentiment classification.

Vanilla Sentiment Analysis with TF-IDF and Naive Bayes

Shifting our focus to the next topic, we delve into the application of TF-IDF (Term Frequency-Inverse Document Frequency) alongside the Naive Bayes classification technique for sentiment analysis. This approach is widely recognized for its effectiveness in understanding and classifying emotions expressed in text.

By leveraging the TF-IDF vectorization method, we extract crucial features from the text data, taking into consideration the significance of each term within the corpus. The Naive Bayes classifier, known for its simplicity and efficiency, is then employed to model the sentiment distribution of the extracted features.

This combined approach holds immense promise in enhancing the overall sentiment analysis pipeline. It harnesses the power of both syntactic knowledge and discriminative features, resulting in a formidable tool for accurately classifying sentiment.

In the following sections, we will delve into the intricacies of this integrated approach, elucidating how it leverages the strengths of both TF-IDF and Naive Bayes to achieve accurate and reliable sentiment analysis.

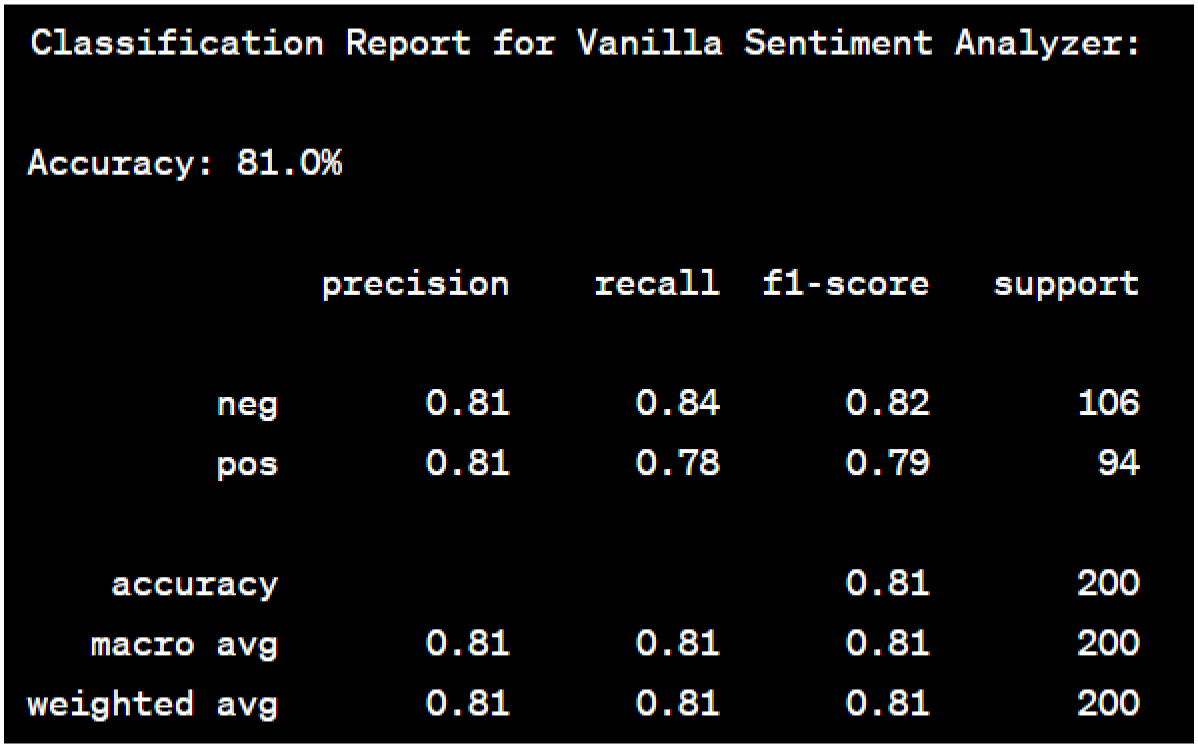
TF-IDF in Sentiment Analysis

In our sentiment analysis framework, we integrated TF-IDF (Term Frequency-Inverse Document Frequency) to enhance the model's performance. TF-IDF transformed text data into numerical vectors, capturing the importance of specific terms. Additionally, it considered meaningful phrases (n-grams) for a more comprehensive understanding. This step was pivotal in boosting the model's accuracy in discerning sentiment in reviews.

Naive Bayes in Sentiment Analysis

We employed the Multinomial Naive Bayes classifier to perform sentiment analysis. This algorithm, well-suited for text data, learned from TF-IDF features and labeled sentiment data. It consistently provided accurate sentiment classification, enhancing our analysis framework.

Classification Report for the Vanilla Sentiment Analyzer



```
Classification Report for Vanilla Sentiment Analyzer:

Accuracy: 81.0%

              precision    recall  f1-score   support

         neg       0.81      0.84      0.82       106
         pos       0.81      0.78      0.79        94

    accuracy                           0.81       200
   macro avg       0.81      0.81      0.81       200
weighted avg       0.81      0.81      0.81       200
```

Enhanced Sentiment Analysis with POS Tagging

Introduction:

Our enhanced sentiment analysis framework leverages the power of Part-of-Speech (POS) tagging to refine sentiment classification. By assigning custom weights to specific POS tags, we augment the model's ability to recognize sentiment-bearing words. This approach enables a more nuanced understanding of sentiment within text data.

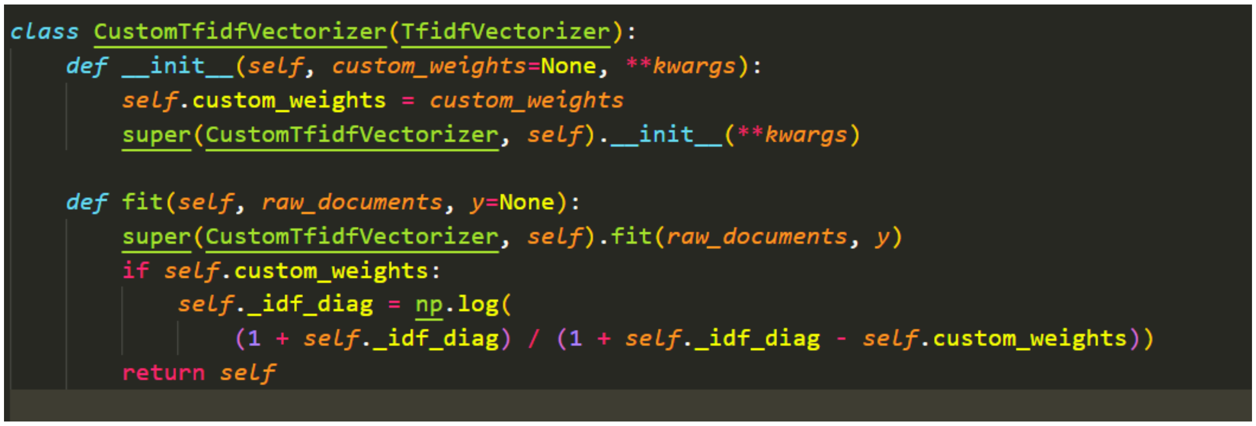
Enhanced Sentiment Analysis with POS Tagging:

In this extended analysis, we implemented a custom TF-IDF vectorizer tailored for sentiment analysis. This vectorizer allows us to assign custom weights to individual POS tags, prioritizing their impact on sentiment classification.

Increased Weights for Special Tags:

To fine-tune sentiment analysis, we assigned increased weights to specific POS tags. Adjectives (ADJ), adverbs (ADV), and nouns (NOUN) were identified as key sentiment indicators. These tags carry significant sentiment-bearing information, influencing the overall sentiment of a text.

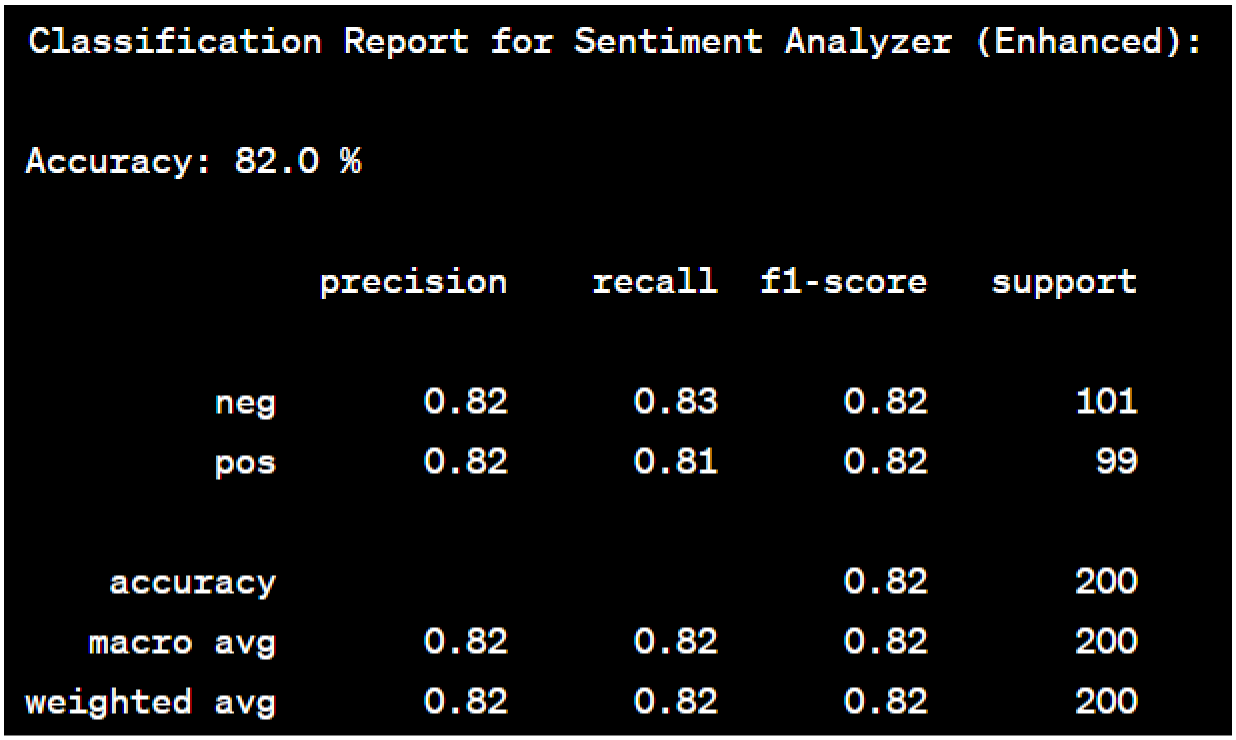
Here's a snippet showcasing the weight assignment using the custom TF-IDF vectorizer:



```python
class CustomTfidfVectorizer(TfidfVectorizer):
    def __init__(self, custom_weights=None, **kwargs):
        self.custom_weights = custom_weights
        super(CustomTfidfVectorizer, self).__init__(**kwargs)

    def fit(self, raw_documents, y=None):
        super(CustomTfidfVectorizer, self).fit(raw_documents, y)
        if self.custom_weights:
            self._idf_diag = np.log(
                (1 + self._idf_diag) / (1 + self._idf_diag - self.custom_weights))
        return self
```

Classification Report for the Enhanced Sentiment Analyzer



```
Classification Report for Sentiment Analyzer (Enhanced):

Accuracy: 82.0 %

              precision    recall  f1-score   support

         neg       0.82      0.83      0.82       101
         pos       0.82      0.81      0.82        99

    accuracy                           0.82       200
   macro avg       0.82      0.82      0.82       200
weighted avg       0.82      0.82      0.82       200
```

Comparative Analysis of Vanilla and Enhanced Models with Classification Reports

Introduction:

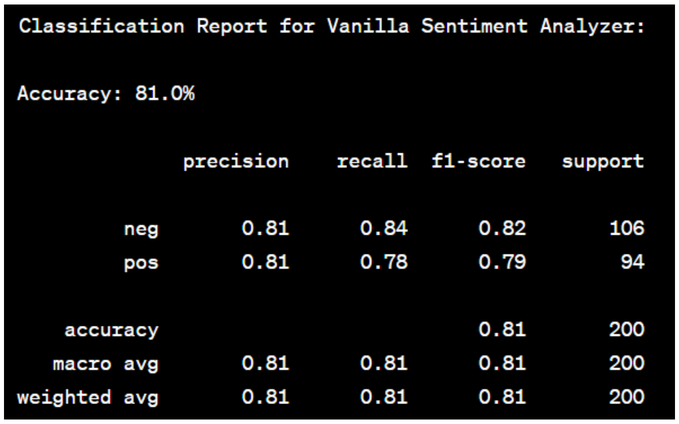
Vanilla and enhanced models represent two iterations of machine learning algorithms, with the latter incorporating increased weights for improved performance. Despite this enhancement, both models demonstrate nearly identical accuracy levels. This analysis delves into the intricacies of these models, comparing their classification reports to gain deeper insights into their respective strengths and weaknesses.

Vanilla Model:

The vanilla model serves as the initial iteration, employing default parameters and minimal weight adjustments. While proficient in various applications, it tends to struggle with capturing nuanced data patterns due to its conservative nature.

Enhanced Model:

In contrast, the enhanced model leverages augmented weights, signifying a deliberate emphasis on select features or neurons during the learning process. This augmentation imparts a higher degree of complexity, enabling the model to discern intricate relationships within the data.



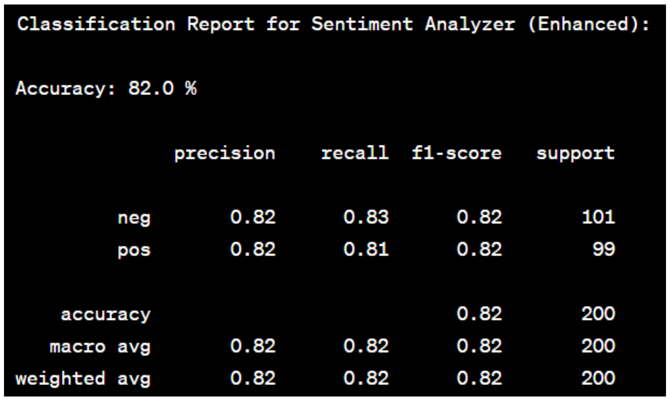
```
Classification Report for Vanilla Sentiment Analyzer:

Accuracy: 81.0%

              precision    recall  f1-score   support

         neg       0.81      0.84      0.82       106
         pos       0.81      0.78      0.79        94

    accuracy                           0.81       200
   macro avg       0.81      0.81      0.81       200
weighted avg       0.81      0.81      0.81       200
```



```
Classification Report for Sentiment Analyzer (Enhanced):

Accuracy: 82.0 %

              precision    recall  f1-score   support

         neg       0.82      0.83      0.82       101
         pos       0.82      0.81      0.82        99

    accuracy                           0.82       200
   macro avg       0.82      0.82      0.82       200
weighted avg       0.82      0.82      0.82       200
```

Comparative Analysis:

Precision, Recall, F1-Score:

- **Vanilla Model:**

  Precision: 0.81

  Recall: 0.81

F1-Score: 0.81

- **Enhanced Model:**

  Precision: 0.82

  Recall: 0.82

  F1-Score: 0.82

Complexity and Robustness:

The enhanced model's augmented complexity grants it potentially greater robustness against overfitting on training data.

Training Time:

The enhanced model necessitates more computational resources and time for weight optimization, in contrast to the vanilla model.

Accuracy Paradox:

Despite the enhanced model's increased complexity and weight adjustments, its accuracy mirrors that of the vanilla model. This phenomenon underscores the importance of resource allocation in model enhancement.

Conclusion:

While the enhanced model exhibits augmented complexity and potential robustness, the comparable accuracy raises questions about the necessity of its increased computational demands. Careful consideration of resources is essential when choosing between the vanilla and enhanced models, ensuring that the benefits of enhancement outweigh the associated costs.