Enhancing Emotion Recognition Using POS Tagging

Hardik Soni | 20CS30023 | Assignment - 1 (NLP: CS60075)
Autumn, 2024

OVERVIEW

Language is built on grammar. The parts of speech are important because they show us how the words relate to each other. Knowing whether a word is a noun or a verb tells us a lot about likely neighboring words and about the syntactic structure around the word. Therefore,
Parts-of-speech tagging serves as a very important building block in NLP application.

What is part-of-speech (POS) tagging?

Part-of-speech refers to the category of words or the lexical terms in the language. Examples of these lexical terms in the English language would be noun, verb, adjective, adverb, pronoun, preposition, etc.

The process of assigning tags to the words of a sentence or your corpus is referred to as part of speech tagging, or POS tagging for short. Because POS tags describe the characteristics structure of lexical terms in a sentence or text, you can use it to make assumptions about the semantics.

Viterbi's Algorithm for Hidden Markov Models (HMM' s)

The Viterbi Algorithm is a dynamic programming solution for finding the most probable hidden state sequence. If we have a set of states Q and a set of observations O, we are trying to find the state sequence that maximizes P(Q|O).

By conditional probability, we can transform P(Q|O) to P(Q,O)/P(O), but there is no need in finding P(O) as P(O) does not pertain to changes in state sequences. We can just find the state sequence that maximizes P(Q,O). Let's dive into the formula for P(Q, O).

Table 5.2 from [Müller, FMP, Springer 2015]

Algorithm: VITERBI

Input: HMM specified by $\Theta = (\mathcal{A}, A, C, \mathcal{B}, B)$
Observation sequence $O = (o_1 = \beta_{k_1}, o_2 = \beta_{k_2}, \ldots, o_N = \beta_{k_N})$
Output: Optimal state sequence $S^* = (s_1^*, s_2^*, \ldots, s_N^*)$

Procedure: Initialize the $(I \times N)$ matrix $\mathbf{D}$ by $\mathbf{D}(i,1) = c_i b_{ik_1}$ for $i \in [1:I]$. Then compute in a nested loop for $n = 2, \ldots, N$ and $i = 1, \ldots, I$:

$$\begin{aligned} \mathbf{D}(i,n) &= \max_{j \in [1:I]} \big(a_{ji} \cdot \mathbf{D}(j, n-1)\big) \cdot b_{ik_n} \\ \mathbf{E}(i,n-1) &= \operatorname{argmax}_{j \in [1:I]} \big(a_{ji} \cdot \mathbf{D}(j, n-1)\big) \end{aligned}$$

Set $i_N = \operatorname{argmax}_{j \in [1:I]} \mathbf{D}(j, N)$ and compute for decreasing $n = N-1, \ldots, 1$ the maximizing indices

$$i_n = \operatorname{argmax}_{j \in [1:I]} \big(a_{j i_{n+1}} \cdot \mathbf{D}(j,n)\big) = \mathbf{E}(i_{n+1}, n).$$

The optimal state sequence $S^* = (s_1^*, \ldots, s_N^*)$ is defined by $s_n^* = \alpha_{i_n}$ for $n \in [1:N]$.

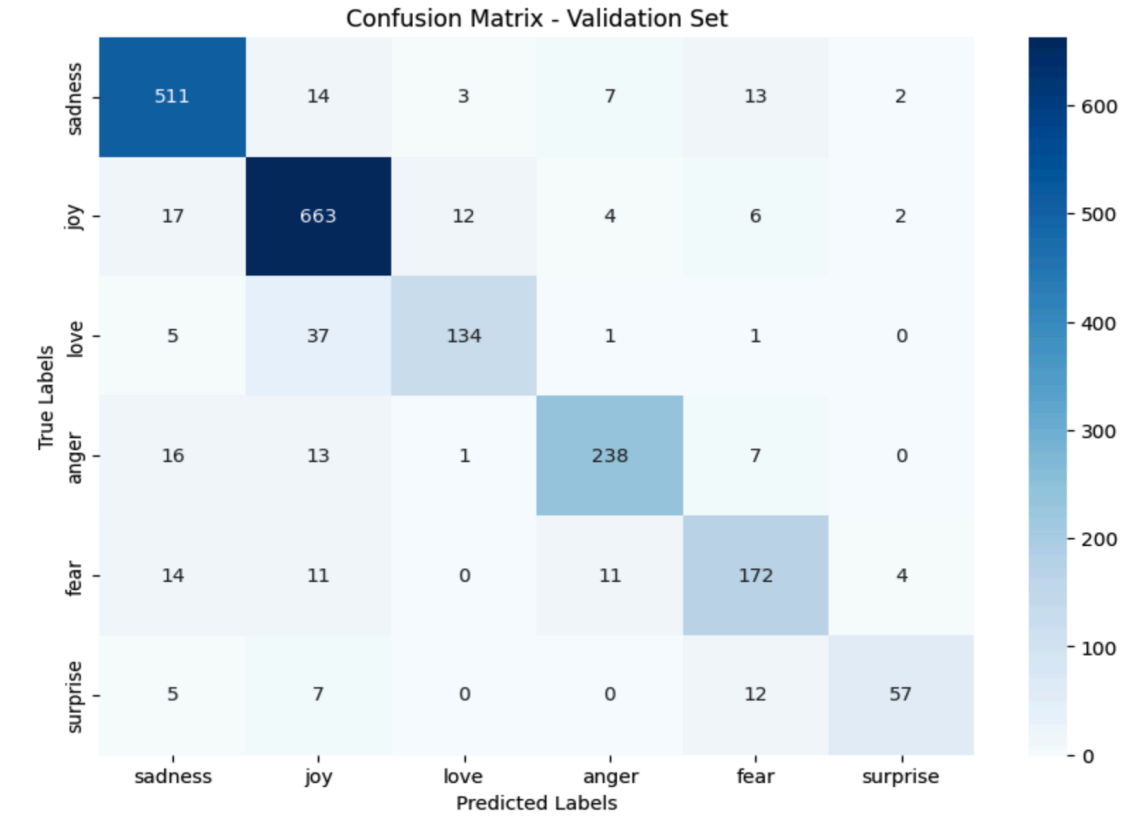
Results (Baseline Model Linear SVC)

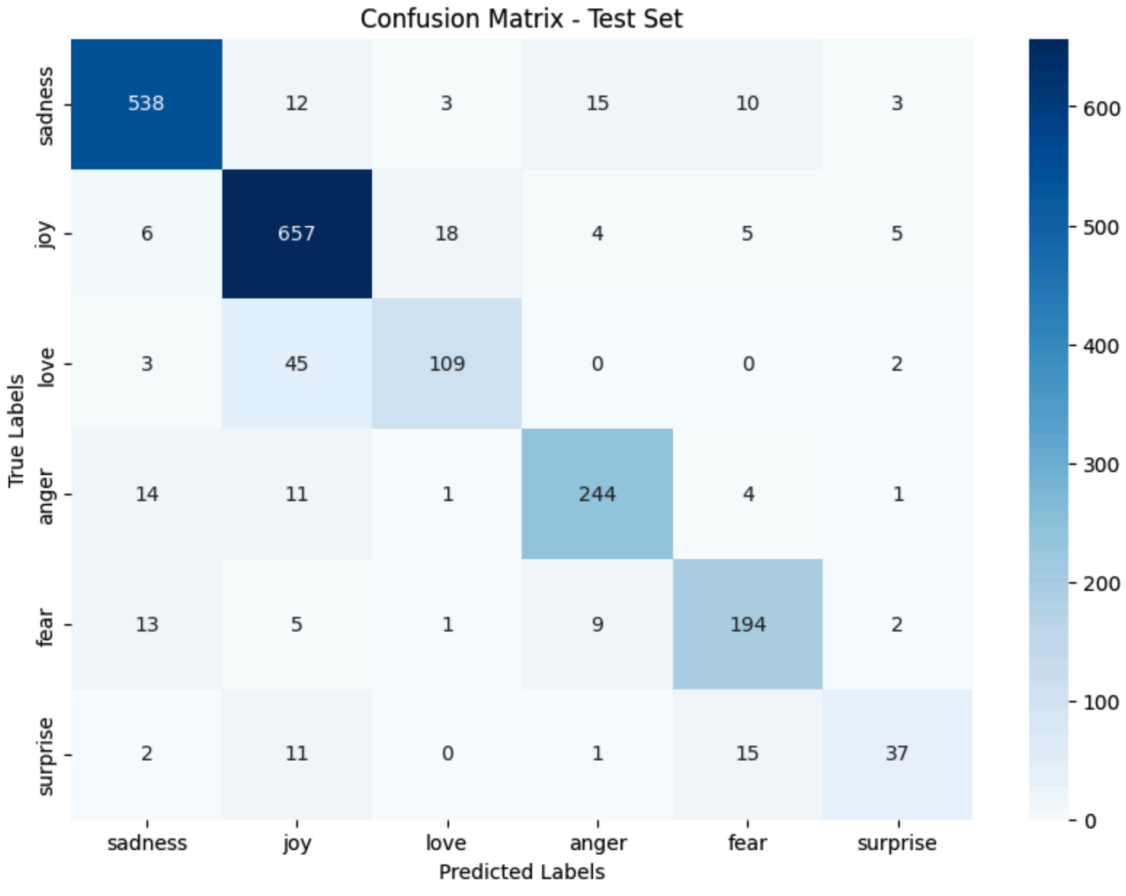
1. Validation Set's Classification Report

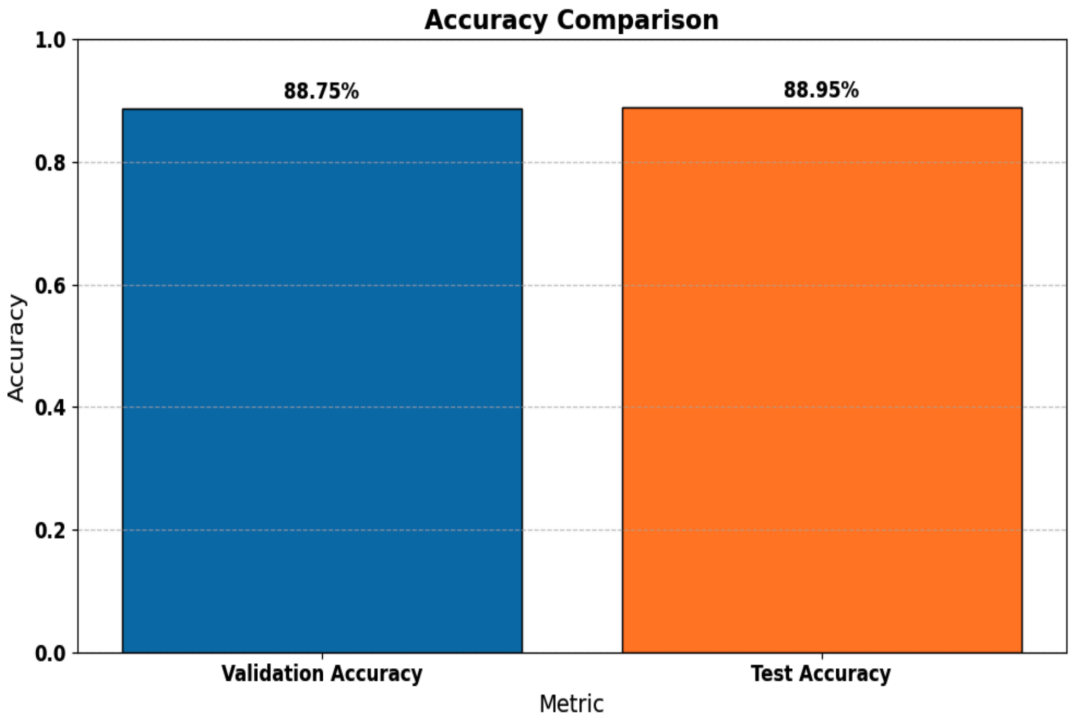
```
Validating Classification on Validation Set.....
Validation Time: 2.96 seconds
Validation Accuracy: 88.75  %.

Validation Report:
              precision    recall  f1-score   support

     sadness       0.90      0.93      0.91       550
         joy       0.89      0.94      0.92       704
        love       0.89      0.75      0.82       178
       anger       0.91      0.87      0.89       275
        fear       0.82      0.81      0.81       212
    surprise       0.88      0.70      0.78        81

    accuracy                           0.89      2000
   macro avg       0.88      0.83      0.85      2000
weighted avg       0.89      0.89      0.89      2000
```

2. Test Set's Classification Report

```
Testing Classification on Test Set.....
Testing Time: 2.95 seconds
Test Accuracy: 0.8895

Test Report:
              precision    recall  f1-score   support

     sadness       0.93      0.93      0.93       581
         joy       0.89      0.95      0.92       695
        love       0.83      0.69      0.75       159
       anger       0.89      0.89      0.89       275
        fear       0.85      0.87      0.86       224
    surprise       0.74      0.56      0.64        66

    accuracy                           0.89      2000
   macro avg       0.86      0.81      0.83      2000
weighted avg       0.89      0.89      0.89      2000
```

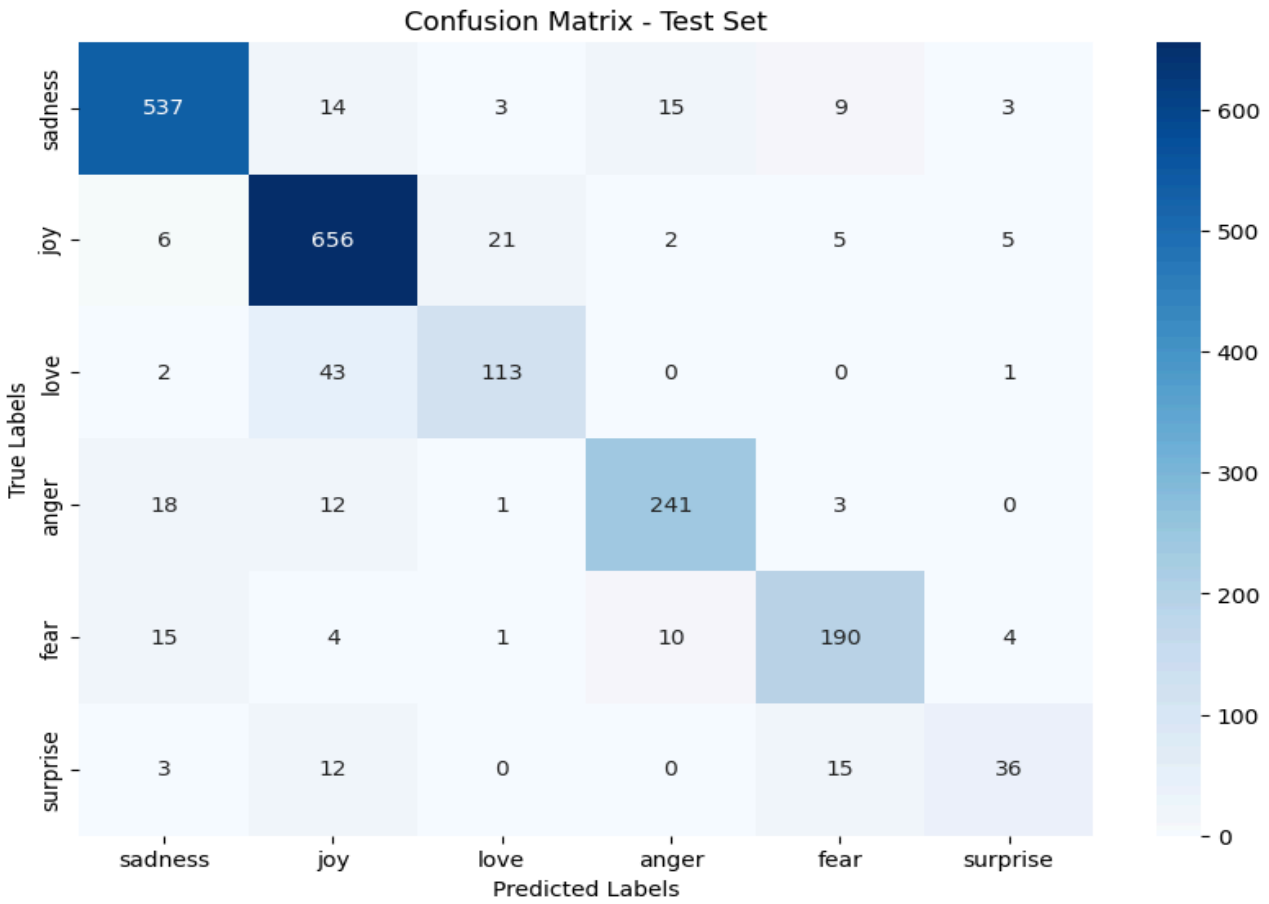

Confusion Matrix - Validation Set
True Labels
Predicted Labels
sadness
joy
love
anger
fear
surprise
511 14 3 7 13 2
17 663 12 4 6 2
5 37 134 1 1 0
16 13 1 238 7 0
14 11 0 11 172 4
5 7 0 0 12 57
0
100
200
300
400
500
600


Confusion Matrix - Test Set
True Labels
Predicted Labels
sadness
joy
love
anger
fear
surprise
538 12 3 15 10 3
6 657 18 4 5 5
3 45 109 0 0 2
14 11 1 244 4 1
13 5 1 9 194 2
2 11 0 1 15 37
0
100
200
300
400
500
600



3. (Training + Validation Data) for Training:- Test Set's Classification Report

```
Testing Classification on Test Set.....
Testing Time: 3.62 seconds
Test Accuracy: 88.64999999999999 %.

Test Report:
              precision    recall  f1-score   support

     sadness       0.92      0.92      0.92       581
         joy       0.89      0.94      0.91       695
        love       0.81      0.71      0.76       159
       anger       0.90      0.88      0.89       275
        fear       0.86      0.85      0.85       224
    surprise       0.73      0.55      0.63        66

    accuracy                           0.89      2000
   macro avg       0.85      0.81      0.83      2000
weighted avg       0.88      0.89      0.88      2000
```



Incorporating POS Tags into TF-IDF Vectorization

Objective:

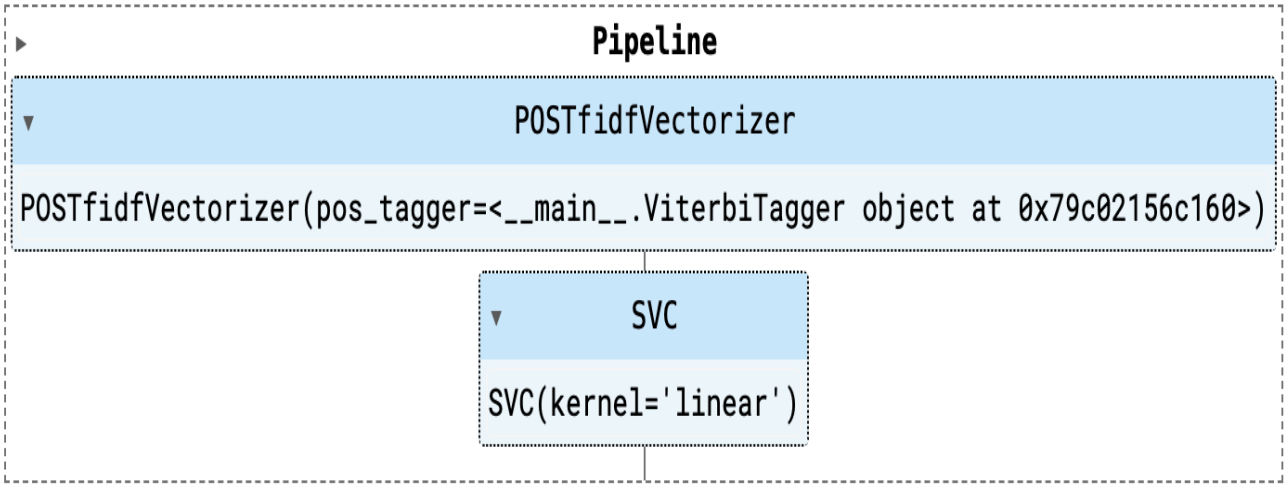
The goal is to enhance the feature representation of text data by integrating Part-of-Speech (POS) tags into the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization process. This approach aims to capture both the lexical and grammatical aspects of the text, potentially improving the performance of the machine learning model.

Implementation Details:

1. **POS Tagging and Concatenation :** The initial step involves tokenizing and POS-tagging the text. Instead of treating words as isolated entities, each word is combined with its corresponding POS tag. For example, the word "happy" with the POS tag "ADJ" (adjective) is transformed into "happy/ADJ". This concatenation provides a more nuanced representation of the word by incorporating its grammatical role.

2. **TF-IDF Vectorization :** After creating the word-POS pairs, we apply the TF-IDF vectorization process. This method calculates the TF-IDF values for each concatenated token (e.g., "happy/ADJ"), allowing the model to learn not only the importance of the words but also their grammatical significance in the context of the corpus.

Out[19]:



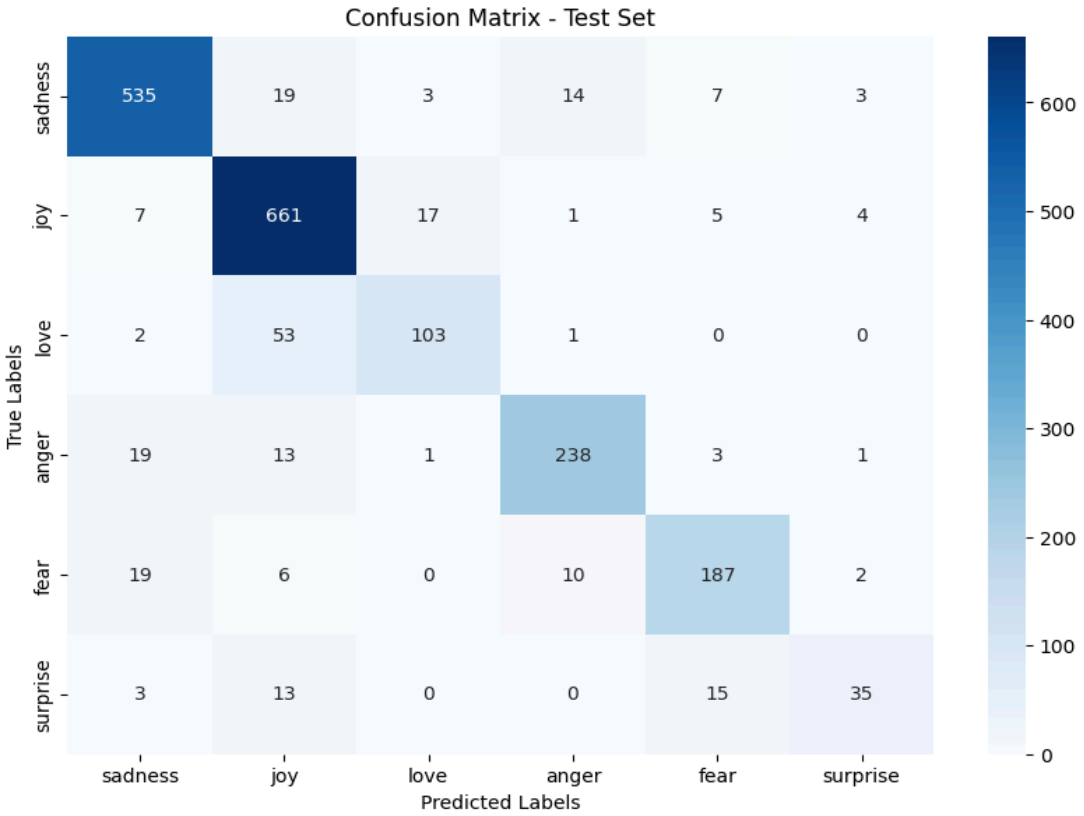
Alternate Approaches:

1. **POS-Tagged Sentence Vectors** : As an alternative, one could compute TF/IDF vectors for sentences that include POS tags and append these vectors as additional features to the original TF/IDF vectors of the words. This approach adds another layer of information but may not capture the precise relationship between words and their tags as effectively as the word-POS concatenation method.
2. **POS-Weighted TF/IDF :** A more sophisticated approach involves computing TF/IDF vectors that are weighted based on POS tags. This method emphasizes the significance of different POS tags in the context of the corpus, potentially improving model performance by better aligning the feature representation with the grammatical structure of the text.

Evaluation:

The effectiveness of incorporating POS tags into the TF-IDF vectorization process should be evaluated by comparing the performance of models trained with and without POS-tagged features. Key metrics for assessment include accuracy, precision, recall, and F1-score. Analyzing confusion matrices and other performance metrics will provide insights into whether the inclusion of POS tags enhances the model's ability to classify or predict based on textual data.

```
Accuracy: 87.950%
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.92      0.92       581
           1       0.86      0.95      0.91       695
           2       0.83      0.65      0.73       159
           3       0.90      0.87      0.88       275
           4       0.86      0.83      0.85       224
           5       0.78      0.53      0.63        66

    accuracy                           0.88      2000
   macro avg       0.86      0.79      0.82      2000
weighted avg       0.88      0.88      0.88      2000
```



Conclusion:

Integrating POS tags into the TF-IDF vectorization process offers a promising approach to enrich the feature representation of text data. By capturing both lexical and grammatical aspects, this method aims to improve the model's performance and provide a more comprehensive understanding of the text. Further experimentation and evaluation will determine the most effective way to incorporate POS tags and assess their impact on model performance.