Enhancing Sentiment Analysis Using POS Tagging

(Assignment-1)

Name: Priyanshi Dixit
Roll Number: 20CS30069

Objective

This study analyses a text classification job including the combination of Term Frequency-Inverse Document Frequency (TF-IDF) features and Part-of-Speech (POS) tag characteristics. The main objective was to assess how well this approach would help a Support Vector Machine (SVM) classifier perform better on a text classification test.

Framework

Data Preparation:

Using a fixed random seed (random_state=42) to ensure consistent splitting, we divide the dataset into training and testing subsets, assigning 20% of the data for testing. The training subset is then further divided into training and validation sets, with validation making up 20% of the remaining data and the same random seed being used for repeatability.

Extraction of Features:

TF-IDF Vectorization: To turn the text data into numerical vectors, we employ the TF-IDF vectorization approach. The most important terms in the corpus are represented by up to 10,000 characteristics.

POS tagging: We use the Viterbi algorithm for the text input in order to conduct POS tagging, which extracts grammatical data in addition to the semantic data gathered by TF-IDF.

Feature Integration:

We combine the TF-IDF-based embeddings with the encoded POS tag features.
For the same, we have a plan in place. It creates feature-rich representations that incorporate both syntactic and semantic information by first encoding the POS tags into binary features before concatenating them with TF-IDF vectors. With this method, the text categorization model is better able to take into account both grammatical structure and semantic content.

Combining Strategy Explanation

1. Combining POS Tags with TF-IDF Features:

The code concatenates all_pos_tags, a list of all the POS tags that were obtained from the training, validation, and test sets, in this phase. The dataset's documents' POS tags will all be included in this list.

2.MultiLabelBinarizer for Encoding:

A MultiLabelBinarizer object called mlb is initialized by the code. A scikit-learn tool called MultiLabelBinarizer is used to transform several binary labels (in this case, POS tags) into a binary matrix format that is appropriate for machine learning.

On mlb, fit_transform is used to encode the POS tags. Each POS tag for every document is converted into a binary representation by this technique. Each distinct POS tag is converted into a binary feature, and each document is represented as a binary vector that indicates whether each POS tag is present or not.

3. Splitting Encoded POS Tags into Sets:

The code divides the encoded POS tags back into training, validation, and test sets after encoding all of the POS tags. The machine learning model will be trained on these sets and evaluated using them.
The encoded POS tags for the training set are contained in the file pos_tags_train_encoded.
The encoded POS tags for the validation set are contained in the file pos_tags_val_encoded.
The encoded POS tags for the test set are contained in the file pos_tags_test_encoded.

4. Combining TF-IDF-Based Embeddings and POS Tag Features:

The algorithm then creates feature matrices for the training, validation, and test sets by combining the TF-IDF-based embeddings and the encoded POS tag features.
The combined feature matrix for the training set is called X_train_combined. It includes both the encoded POS tag features and the TF-IDF-based features (which were transformed to a NumPy array using toarray()).
The combined feature matrix for the validation set is called X_val_combined.
The combined feature matrix for the test set is called X_test_combined.

The generated feature matrices contain both semantic content information (from TF-IDF) and syntactic structure information (from POS tags) as a result of merging the two features in this manner. This might provide the text data a better representation and possibly enhance how well machine learning models function.

Why did we choose this strategy?

This strategy was chosen because of the below-mentioned benefits:

- This approach combines the depth of textual data gathered by TF-IDF-based embeddings with the grammatical and structural data offered by POS tags. It makes use of the text's syntactic and semantic components, enhancing its comprehensiveness and educational value for various NLP tasks.
    - Use Case: It is especially helpful in activities like sentiment analysis, text categorization, and named entity identification where comprehending both the meaning and grammatical structure of the text is crucial.

- The approach may be tailored and adaptable. You may pick the encoding method for POS tags, the number of features for the TF-IDF, and the classifier (in this case, SVM). It is appropriate for a variety of NLP issues because of its versatility.
    - Use Case: Because different datasets and jobs may call for various setups, the ability to fine-tune parameters is essential in real NLP applications.

Observations

We train an SVM classifier with a linear kernel on the integrated features using the training data. We then evaluate the model's performance on the validation and test sets. Model evaluation includes calculating accuracy scores and generating a detailed classification report.

Validation Findings

Since the entire dataset takes a large training time, taking in time constraints, we had **256 training entities and 64 validation entities set**, the SVM classifier has an accuracy of about

`SVM Validation Accuracy: 0.7344`

Precision, recall, F1-score, and support are just a few of the performance metrics the classification report gives for the model for each class.

Test Findings:

Since the entire dataset takes a large training time, taking in time constraints, we had **256 training entities and 80 testing entities set,**

The SVM classifier's accuracy on the test set was around

`SVM Test Accuracy: 0.6000`

The test set's categorization report gives various parameters for each class

Comparison with the baseline model

The following findings were reached while comparing the POS-tag-enhanced model to the baseline model, which does not make use of POS tags:

Model Baseline (Without POS Tags):

The baseline model was not trained using POS tags but just on TF-IDF-based embeddings.
The model showed its classification performance in the validation and test sets.
A thorough understanding of the model's performance in the lack of syntactic information was supplied by the model's precision, recall, and F1 scores for each class in the form of detailed classification reports.

```
Again take into consideration
len(X_train) = 256
len(X_val) = 64
len(X_test) = 80
```

We have taken these so that the execution time is less. We would get better result for training for a longer period of time

BASELINE MODEL WITHOUT TAGS

SVM Validation Accuracy: 0.6719
Classification Report for Validation:

	precision	recall	f1-score	support
neg	0.65	0.59	0.62	29
pos	0.68	0.74	0.71	35
accuracy			0.67	64
macro avg	0.67	0.66	0.67	64
weighted avg	0.67	0.67	0.67	64

SVM Test Accuracy: 0.6000
Classification Report for Testing:

	precision	recall	f1-score	support
neg	0.71	0.49	0.58	45
pos	0.53	0.74	0.62	35
accuracy			0.60	80
macro avg	0.62	0.62	0.60	80
weighted avg	0.63	0.60	0.60	80

Model with POS-Tag Enhancement:

To capture both semantic and syntactic properties in text input, the POS-tag-enhanced model combined POS tags and TF-IDF-based embeddings.
To assess the model's effectiveness, classification reports for the validation and test sets were created.
These reports showed the accuracy, recall, and F1 scores for each class, showing how the model's performance was affected by the introduction of grammatical information.

Again take into consideration
len(X_train) = 256
len(X_val) = 64
len(X_test) = 80
We have taken these so that the execution time is less. We would get better result for training for a longer period of time

POS-TAG ENHANCED MODEL

SVM Validation Accuracy: 0.7344
Classification Report for Validation:

	precision	recall	f1-score	support
neg	0.69	0.76	0.72	29
pos	0.78	0.71	0.75	35
accuracy			0.73	64
macro avg	0.73	0.74	0.73	64
weighted avg	0.74	0.73	0.73	64

```
SVM Test Accuracy: 0.6000
Classification Report for Testing:
              precision    recall  f1-score   support

         neg       0.67      0.58      0.62        45
         pos       0.54      0.63      0.58        35

    accuracy                           0.60        80
   macro avg       0.60      0.60      0.60        80
weighted avg       0.61      0.60      0.60        80
```

Conclusion

In conclusion, this experiment demonstrates the potential advantages of utilizing syntactic data from POS tagging to improve sentiment analysis models. It is especially important to comprehend subtle sentiment expressions in text, therefore the combination of semantic and syntactic data shows promise for enhancing the precision and depth of sentiment categorization.