Assignment 1: Enhancing Emotion Recognition Using POS Tagging

Prasanna Paithankar (21CS30065)
Department of Computer Science, Indian Institute of Technology Kharagpur
paithankarprasanna@kgpian.iitkgp.ac.in

Abstract

This document contains the implementation of a POS tagger using the Viterbi Algorithm from scratch. The POS tagger is trained on the Treebank dataset from the NLTK library. The POS tagger is then used to enhance the emotion recognition system. The emotion recognition system is trained on the Twitter messages dataset from the Hugging Face API. The system uses Tfidf for creating sentence embeddings and a classical classifier (Naive Bayes or SVM) for emotion recognition. The system is then improved by using the POS tagger to tag the dataset and augment it before sentence embedding step. The classifier is then trained again for emotion recognition using the augmented dataset. The system is evaluated on the test set of the Twitter messages dataset. The improve recognizer performs slightly better than the vanilla recognizer as presented in the classification reports and confusion matrices. The time taken for training is also presented.

Assignment problem statement: https://sites.google.com/view/nlp-cs-iit-kgp/assignments

1 POS Tagging by Viterbi Algorithm

A bottom-up dynamic version of the Viterbi algorithm has been implemented for POS tagging on the Treebank dataset from scratch.

Input: A sentence $S = w_1, w_2, w_3, ..., w_n$
Output: The most probable sequence of POS tags $T = t_1, t_2, t_3, ..., t_n$ for the sentence S

Initialization: $\pi(0, < s >, < s >) = 1$
for $k = 1$ *to* n **do**
  for *each POS tag* u **do**
    for *each POS tag* v **do**
      $\pi(k, u, v) = \max_w(\pi(k-1, w, u) \times P(u|w) \times P(w|v))$
      $(k, u, v) = \arg\max_w(\pi(k-1, w, u) \times P(u|w) \times P(w|v))$
    end
  end
end
$T_n = \arg\max_{u,v}(\pi(n, u, v) \times P(< /s > |u) \times P(u|v))$
$T_{n-1} = (n, T_n, < /s >)$
for $k = n - 2$ *to* 1 **do**
  $T_k = (k + 2, T_{k+1}, T_{k+2})$
end

Algorithm 1: Viterbi Algorithm

1.1 Computing Transition and Emission Probabilities

The transition and emission probabilities are computed using the Treebank dataset. The transition probability $P(u|v)$ is the probability of a POS tag u given the previous POS tag v. The emission probability $P(w|v)$ is the probability of a word w given the POS tag v. The probabilities are computed as follows:

$$P(u|v) = \frac{C(v,u)}{C(v)} \tag{1}$$

$$P(w|v) = \frac{C(v,w)}{C(v)} \tag{2}$$

where $C(v,u)$ is the count of the POS tag u following the POS tag v, $C(v,w)$ is the count of the word w with the POS tag v, and $C(v)$ is the count of the POS tag v.

In the final implementation, we apply smoothing to the probabilities to handle unseen words and POS tags.

$$P(u|v) = \frac{C(v,u) + \epsilon}{C(v) + \epsilon \times |V|} \tag{3}$$

$$P(w|v) = \frac{C(v,w) + \epsilon}{C(v) + \epsilon \times |V|} \tag{4}$$

where $|V|$ is the size of the vocabulary and ϵ is a smoothing hyper-parameter.

1.2 Results

We evaluate the implemented POS tagger on the Treebank dataset and compare the results with the NLTK POS tagger. Time benchmarked on 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 2401 Mhz, 4 Core(s), 8 Logical Processor(s).

POS Tagger	**Accuracy**	**Time (s)**
viterbi	0.97162	104
nltk.pos_tag	0.89298	3

Table 1: POS Tagging Results

Some sample POS tagging results are presented in the Appendix.

2 Vanilla Emotion Recognizer

The Twitter messages corpus is loaded using the Hugging Face API. The dataset is split into training, validation, and test sets. The corpus has 6 labels: joy, sadness, anger, fear, love, and surprise. We perform a multi-class classification on the training set and evaluate the model on the test set.

2.1 Data Preprocessing and Vectorization

The text data is preprocessed by lemmatizing the words. Lemmatization is implemented as WordNet Lemmatizer from the NLTK library. The text data is then vectorized using Tfidf to create sentence embeddings.

2.2 Naive Bayes Classifier

Naive Bayes classifier is implemented as sklearn.naive_bayes.MultinomialNB. The classifier is trained on the training set and evaluated on the test set.

Training time: 0s

The results are presented in Table 2 and Figure 1.

Label	Precision	Recall	F1-score	Support
sadness	0.68	0.92	0.78	581
joy	0.62	0.99	0.76	695
love	1.00	0.03	0.05	159
anger	0.94	0.19	0.31	275
fear	1.00	0.19	0.32	224
surprise	0.00	0.00	0.00	66
accuracy			0.66	2000
macro avg	0.71	0.38	0.37	2000
weighted avg	0.73	0.66	0.57	2000

Table 2: Naive Bayes Classifier Results

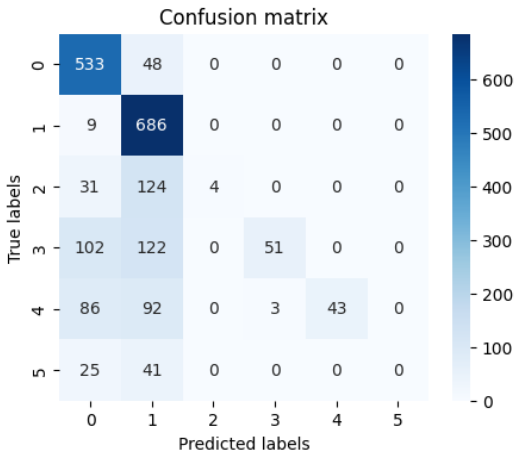

Figure 1: Confusion Matrix for Naive Bayes Classifier

2.3 SVM Classifier

SVM classifier is implemented as sklearn.svm.SVC. The classifier is trained on the training set and evaluated on the test set.

Training time: 125s

The results are presented in Table 3 and Figure 2.

Label	Precision	Recall	F1-score	Support
sadness	0.89	0.93	0.91	581
joy	0.82	0.96	0.88	695
love	0.87	0.54	0.67	159
anger	0.90	0.79	0.84	275
fear	0.86	0.78	0.82	224
surprise	0.85	0.44	0.58	66
accuracy			0.86	2000
macro avg	0.86	0.74	0.78	2000
weighted avg	0.86	0.86	0.85	2000

Table 3: SVM Classifier Results

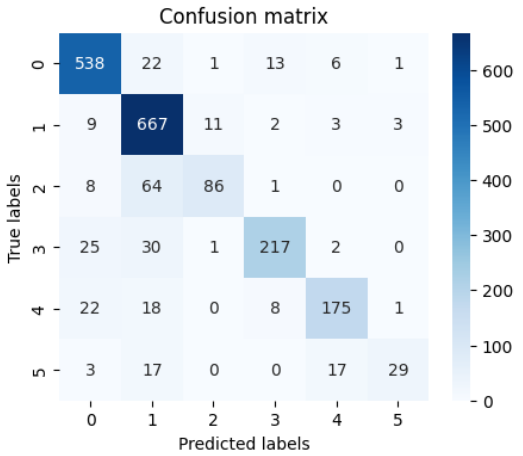

Figure 2: Confusion Matrix for SVM Classifier

3 Improved Emotion Recognizer

The corpus is augmented using the POS tagger implemented in the previous section. We present the pipeline for the same and two augmentation strategies.

3.1 Pipeline

Input: A sentence $S = w_1, w_2, w_3, ..., w_n$
Input: POS tag classes $C = c_1, c_2, c_3, ..., c_m$
Output: The most probable emotion label E for the sentence S

POS tagging: $T = t_1, t_2, t_3, ..., t_n$
Filtering: $S' = w_i$ if $t_i \in C$
Tfidf Vectorization: $X = \text{Tfidf}(S')$
Train the model: $M = \text{Classifier}(X)$
Predict the emotion label: $E = M.predict(X)$

Algorithm 2: Improved Emotion Recognizer Pipeline

3.2 Augmentation

The POS tagger is used to tag the Twitter messages dataset. We manually select the POS tag classes that are relevant for emotion recognition. The POS tag classes are used to filter the words in the dataset. The filtered words are then used to create sentence embeddings using Tfidf.

We use the following POS tag classes for augmentation: '<s>', '-LRB-', 'NN', 'JJR', 'NNP', 'VBZ', 'RB', "''", '</s>', 'VBN', 'RBS', 'VB', 'NNPS', '-RRB-', 'NNS', 'PDT', 'JJ', 'VBP', 'RP', 'VBG', 'RBR', 'FW', 'VBD', 'WDT', 'WRB', '-NONE-', 'WP', 'POS', 'JJS', '``', 'WP$', 'UH'

3.3 Alternative Augmentation Strategy

(This has not been implemented in the final code)

We consider words which have ambiguous POS tags. We take such words and replace them with a synonym using nltk.corpus.wordnet.synsets. We then use the POS tagger to tag the augmented dataset after removing stopwords.

3.3.1 Naive Bayes Classifier

Naive Bayes classifier is trained on the augmented dataset and evaluated on the test set.

Training time: 0s

The results are presented in Table 4 and Figure 3.

Label	Precision	Recall	F1-score	Support
sadness	0.68	0.93	0.79	581
joy	0.64	0.98	0.78	695
love	1.00	0.04	0.08	159
anger	0.94	0.27	0.41	275
fear	0.92	0.25	0.39	224
surprise	0.00	0.00	0.00	66
accuracy			0.68	2000
macro avg	0.70	0.41	0.41	2000
weighted avg	0.73	0.68	0.61	2000

Table 4: Naive Bayes Classifier Results on Augmented Dataset

3.3.2 SVM Classifier

SVM classifier is trained on the augmented dataset and evaluated on the test set.

Training time: 108s

The results are presented in Table 5 and Figure 4.

4 Comparison

The results of the vanilla and improved emotion recognizers are compared in the Table 6.

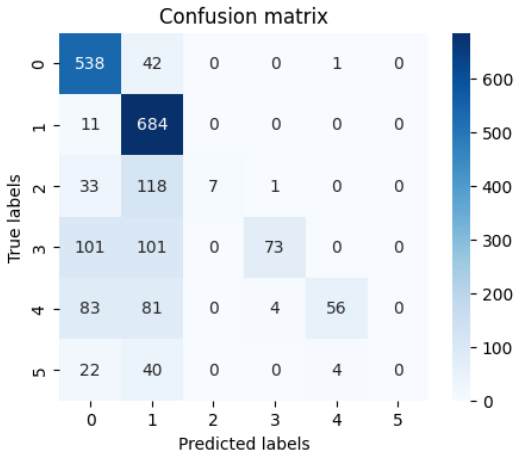

Figure 3: Confusion Matrix for Naive Bayes Classifier on Augmented Dataset

Label	**Precision**	**Recall**	**F1-score**	**Support**
sadness	0.90	0.92	0.91	581
joy	0.81	0.96	0.88	695
love	0.87	0.53	0.66	159
anger	0.90	0.79	0.84	275
fear	0.87	0.79	0.82	224
surprise	0.85	0.50	0.63	66
accuracy			0.86	2000
macro avg	0.87	0.75	0.79	2000
weighted avg	0.86	0.86	0.85	2000

Table 5: SVM Classifier Results on Augmented Dataset

5 Conclusion

We have implemented a POS tagger using the Viterbi Algorithm from scratch. The POS tagger is trained on the Treebank dataset and evaluated (97%) giving better results than nltk.pos_tag (89%). We then implement an emotion recognition system using the Twitter messages dataset. The system uses Tfidf for creating sentence embeddings and a classical classifier (Naive Bayes or SVM) for emotion recognition. The system is then improved by augmenting the dataset using the POS tagger. The classifier is trained again for emotion recognition using the augmented dataset. The system is evaluated on the test set of the Twitter messages dataset. We find slight improvement in the performance of the improved recognizer as compared to the vanilla recognizer.

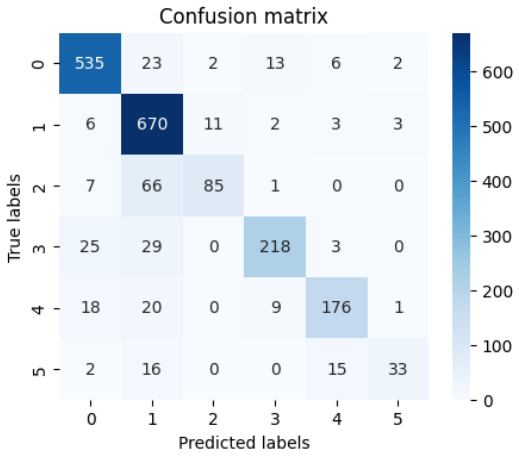

Figure 4: Confusion Matrix for SVM Classifier on Augmented Dataset

Classifier	**Accuracy**	**Macro average F1-score**	**Training time (s)**
Naive Bayes (Vanilla)	0.66	0.37	0
Naive Bayes (Improved)	0.68	0.41	0
SVM (Vanilla)	0.86	0.78	125
SVM (Improved)	0.86	0.79	108

Table 6: Comparison of Vanilla and Improved Emotion Recognizers

A Viterbi Tagging Samples

Format: [(Word, Actual tag, Predicted tag), ...]

1. [('Pierre', 'NNP', 'NNP'), ('Vinken', 'NNP', 'NNP'), (',', ',', ','), ('61', 'CD', 'CD'), ('years', 'NNS', 'NNS'), ('old', 'JJ', 'JJ'), (',', ',', ','), ('will', 'MD', 'MD'), ('join', 'VB', 'VB'), ('the', 'DT', 'DT'), ('board', 'NN', 'NN'), ('as', 'IN', 'IN'), ('a', 'DT', 'DT'), ('nonexecutive', 'JJ', 'JJ'), ('director', 'NN', 'NN'), ('Nov.', 'NNP', 'NNP'), ('29', 'CD', 'CD'), ('.', '.', '.')]

2. [('Mr.', 'NNP', 'NNP'), ('Vinken', 'NNP', 'NNP'), ('is', 'VBZ', 'VBZ'), ('chairman', 'NN', 'NN'), ('of', 'IN', 'IN'), ('Elsevier', 'NNP', 'NNP'), ('N.V.', 'NNP', 'NNP'), (',', ',', ','), ('the', 'DT', 'DT'), ('Dutch', 'NNP', 'JJ'), ('publishing', 'VBG', 'NN'), ('group', 'NN', 'NN'), ('.', '.', '.')]

3. [('Rudolph', 'NNP', 'NNP'), ('Agnew', 'NNP', 'NNP'), (',', ',', ','), ('55', 'CD', 'CD'), ('years', 'NNS', 'NNS'), ('old', 'JJ', 'JJ'), ('and', 'CC', 'CC'), ('former', 'JJ', 'JJ'), ('chairman', 'NN', 'NN'), ('of', 'IN', 'IN'), ('Consolidated', 'NNP', 'NNP'), ('Gold', 'NNP', 'NNP'), ('Fields', 'NNP', 'NNP'), ('PLC', 'NNP', 'NNP'), (',', ',', ','), ('was', 'VBD', 'VBD'), ('named', 'VBN', 'VBN'), ('*-1', '-NONE-', '-NONE-'), ('a', 'DT', 'DT'), ('nonexecutive', 'JJ', 'JJ'), ('director', 'NN', 'NN'), ('of', 'IN', 'IN'), ('this', 'DT', 'DT'), ('British', 'JJ', 'JJ'), ('industrial', 'JJ', 'JJ'), ('conglomerate', 'NN', 'NN'), ('.', '.', '.')]

4. [('A', 'DT', 'DT'), ('form', 'NN', 'NN'), ('of', 'IN', 'IN'), ('asbestos', 'NN', 'NN'), ('once', 'RB', 'RB'), ('used', 'VBN', 'VBN'), ('*', '-NONE-', '-NONE-'), ('*', '-NONE-', '-NONE-'), ('to', 'TO', 'TO'), ('make', 'VB', 'VB'), ('Kent', 'NNP', 'NNP'), ('cigarette', 'NN', 'NN'), ('filters', 'NNS', 'NNS'), ('has', 'VBZ', 'VBZ'), ('caused', 'VBN', 'VBN'), ('a', 'DT', 'DT'), ('high', 'JJ', 'JJ'), ('percentage', 'NN', 'NN'), ('of', 'IN', 'IN'), ('cancer', 'NN', 'NN'), ('deaths', 'NNS', 'NNS'), ('among', 'IN', 'IN'), ('a', 'DT', 'DT'), ('group', 'NN', 'NN'), ('of', 'IN', 'IN'), ('workers', 'NNS', 'NNS'), ('exposed', 'VBN', 'VBN'), ('*', '-NONE-', '-NONE-'), ('to', 'TO', 'TO'), ('it', 'PRP', 'PRP'), ('more', 'RBR', 'JJR'), ('than', 'IN', 'IN'), ('30', 'CD', 'CD'), ('years', 'NNS', 'NNS'), ('ago', 'IN', 'RB'), (',', ',', ','), ('researchers', 'NNS', 'NNS'), ('reported', 'VBD', 'VBD'), ('0', '-NONE-', '-NONE-'), ('*T*-1', '-NONE-', '-NONE-'), ('.', '.', '.')]

5. [('The', 'DT', 'DT'), ('asbestos', 'NN', 'NN'), ('fiber', 'NN', 'NN'), (',', ',', ','), ('crocidolite', 'NN', 'NN'), (',', ',', ','), ('is', 'VBZ', 'VBZ'), ('unusually', 'RB', 'RB'), ('resilient', 'JJ', 'JJ'), ('once', 'IN', 'IN'), ('it', 'PRP', 'PRP'), ('enters', 'VBZ', 'VBZ'), ('the', 'DT', 'DT'), ('lungs', 'NNS', 'NNS'), (',', ',', ','), ('with', 'IN', 'IN'), ('even', 'RB', 'RB'), ('brief', 'JJ', 'JJ'), ('exposures', 'NNS', 'NNS'), ('to', 'TO', 'TO'), ('it', 'PRP', 'PRP'), ('causing', 'VBG', 'VBG'), ('symptoms', 'NNS', 'NNS'), ('that', 'WDT', 'WDT'), ('*T*-1', '-NONE-', '-NONE-'), ('show', 'VBP', 'VBP'), ('up', 'RP', 'RP'), ('decades', 'NNS', 'NNS'), ('later', 'JJ', 'JJ'), (',', ',', ','), ('researchers', 'NNS', 'NNS'), ('said', 'VBD', 'VBD'), ('0', '-NONE-', '-NONE-'), ('*T*-2', '-NONE-', '-NONE-'), ('.', '.', '.')]