VITERBI IMPLEMENTATION

- Initialized a path matrix (NxL) for storing the back pointers for the scores.
- Initialized a best_scores matrix (NxL) to store the score of the best path to a word-tag combination.
- Stored the first row of the best_scores matrix as a sum of emission_scores for the first token and start transition scores for all labels.
- Computed best_scores for all tokens starting from word 1 to N, for all labels by invoking get_best_score function
- For current token - w and current label – l, over all labels temp_label, get_best_Score function computes the maximum possible sum of best_score[w-1][temp_label] and transition score from temp_label to l.
- Stored the temp_label_index in path[w][l] corresponding to the maximum sum of best_score[w-1][temp_label] and transition score from temp_label to l.
- Computed the maximum_final_score value and final_column_tag index by computing the maximum value of sum of best_scores[N-1][label] and end_scores[label] over all values of label L
- In function get_best_path, I backtracked from final_column_tag index till 0th index in the path matrix to store the best path backwards in string final_path.
- Reversed the final_path value to get the path from start.
- Asserted that length of path should be equal to number of tokens as each token is associated with a label.
- Asserted that max_final_Score is updated and is not the initial value of negative infinity.

ISSUES

- As the best_scores matrix had LxN dimension as per taught in class, was little tough to visualize it as NxL, but realized that it behaves the same. Replaced column wise operations as per taught in class with row-wise.
- Tried to track the path from the start but realized that I cannot. The final_tag_index is the start point to backtrack.

FEATURE DESCRIPTION

1.Words beginning with @

To check whether a word begins with a '@' symbol. Users are quite often mentioned in tweets and twitter handles begin with an '@' symbol. This feature clusters all the mentions of users together thus helping the model realize this is a particular type of word. Ex. @nickjonas123

2.Words beginning with #:

A **hashtag** is a label used on social media sites that makes it easier to find information with a theme or specific content. When it comes to Twitter and its tweets, Hashtags are of utmost importance. Thus, this feature checks whether a word begins with a '#' symbol and tags it with a feature named IS_HASHTAG. Ex. #HappyNewYears

3.URLs:

It is a very common trend to share links to good articles, blog posts related to latest news, inventions etc via tweets. Hence, words that begin with 'http', 'https', 'www' and end with '.com', '.in', '.ly' or urls redirecting to a specific page like 'tinyurl.com/12345' have been categorized and clustered with a feature named IS_URL.

4.Stemming:

Majority of the words used on twitter or in english language are affixes of a root word. For example, *argue*, *argued*, *argues*, *arguing*, and *argus* reduce to the stem *argu*. It is of utmost importance to categorize such words as they eventually belong to a family of a root word. Stemming is the particular case of tokenization which reduces inflected forms to a single base form or stem. Words are given feature as STEMMED_WORD='<stem word>' using the Porter Stemmer Implementation of NLTK Library.

5.Lemmatization:

Lemmatization is a process of reducing words into their lemma or dictionary. It takes into account the meaning of the word in the sentence. For eg: *beautiful* and *beautifully* are lemmatised to *beautiful* and *beautifully* respectively without changing the meaning of the words. But, *good*, *better* and *best* are lemmatised to *good* since all the words have similar meaning. It is extremely important to categorize words having a similar dictionary meaning as they mean the same thing. Words are given feature as LEMMATIZED_WORD='<lemma>' using the WordNetLemmatizer Implementation of NLTK Library.

6.Phonetic Similarity:

Clustered words if sound of two or more words share same phonetic features. Since tweets contains words like 'b4' for before, '2mrw', '2morow', '2moroww' for tomorrow, they are grouped together as they mean the same thing. Word is given a feature as its phonetic representation using the double metaphone library of NLTK.

7.Suffixes and Prefixes:

Suffixes are one the most important factor when it comes to decide whether a word is a verb or a noun or an adjective etc. For example, if a words having suffixes as *'ion', 'ence', 'ness'* can belong to the Noun class with a high probability. Also, words having suffixes as 'ing','ise','fy' are mostly verbs. This is an important property which helps the model classify a word into its respective POS tag. Words are given features as IS_VERB, IS_ADVERB, IS_NOUN, IS_ADJECTIVE based on popular suffixes which the respective grammar classes have. A prefix is placed at the beginning of a word to modify or change its meaning. Though the meaning changes, the root meaning remains the same and hence words having prefixes such as *'pre', 'mono', 'trans', 'tri'* are tagged with a feature named 'HAS_PREFIX'

8.Abbreviation:

Since the number of characters in a tweet are limited, It is difficult for a user to express all his feelings/thoughts. Hence, almost all the users use abbreviations to express their emotions in short words. Most common twitter abbreviations are *LOL, LMAO, RT* etc. Thus, abbreviation is a key feature in twitter tweets. Therefore, it is important to cluster them as they have a similar pattern and hence words are tagged with the feature, IS_ABBREVIATION in my feature_gen.py.

9.Singular Noun:

Clustering of singular and plural words is important as they both share the same meaning, just different quantities. If a word is in plural form of some other word, its singular form is added as a feature by the name of SINGULAR=<Singular word>. For example, *Animal* will be added as a feature to the word *Animals*. But no feature will be added to the word Animal as it is already singular.

10. Number as a Word:

Clustering is done to group words having representation of a number in numeric and verbal form. Since 1 and One mean the same thing, One is added as a feature of 1. This is done using the number_to_word function of inflect package.

11. Exclamation, Periods:

Users are prone to using symbols like exclamations and periods to express their thoughts on twitter. These are not proper words and can be classified as noise. Hence such symbolic words are clustered and tagged with features as IS_EXCLAMATION and IS_PERIOD respectively. Ex. Happy!!!!!!

12. Quantifiers:

Words like all, many, few, lot are words which quantify a noun. Ex. *There are many apples*. Here, ‘many’ behaves as an adjective to the noun ‘apple’. They are tagged with feature named IS_QUANTIFIER.

FEATURE COMPARISON

Below are the results of Dev evaluation for feature comparison on the Logistic Regression Model on addition of each of the above mentioned feature:

Feature	**Token-wise accuracy**	**Token-wise F1 (macro)**	**Token-wise F1 (micro)**	**Sentence Wise Accuracy**
Basic	84.38978240302744	83.3342279970571	84.3897824030274	8.9285714285714
Metaphones	84.91012298959319	83.79692081904301	84.9101229895931	10.71428571428571
Stemming and Lemmatization	85.61967833491012	84.41082716197673	85.6196783349101	11.60714285714285
Singular Noun of Plurals	85.61967833491012	84.37496750621976	85.6196783349101	11.60714285714285
Word Representation of Numbers	85.66698202459791	84.63664859901523	85.6669820245979	11.60714285714285
# and @ Check	86.04541154210028	84.8666923344556	86.0454115421003	12.57895358395983
Abbreviation and Quantifier Check	86.04541154210028	84.67505492657406	86.0454115421003	13.39285714285714
Suffixes And Prefix Check	86.75496688741721	85.56314421782075	86.7549668874172	13.39285714285714
URL Check	86.802270577105	85.63494741462394	86.802270577105	14.28571428571428
! and . Check	87.03878902554399	85.83153159283303	87.038789025544	15.17857142857142

Observations:

Metaphones increase the accuracy by a good amount as meta phones are widely used in tweets. Stemming and Lemmatization increase the accuracy significantly. As twitter tweets mostly have hashtags and mentions, incorporating that feature increased the accuracy significantly from 85.66 to 86.04. Also, checking whether a word is adverb, verb, noun, adjective increases the accuracy to a great amount of 86.75 from 86.04. This feature helps the model to a great extent to tag it’s parts of speech. Finally, as twitter tweets express wide range of emotions in a limited texts, use of exclamations and periods is quite obvious and hence that feature improves the accuracy by a good number too.

COMPARISON of MEMM AND CRF

With given basic features, Learning rate - 1.0

MEMM	CRF
Token-wise accuracy 84.38978240302744 Token-wise F1 (macro) 83.33422799705717 Token-wise F1 (micro) 84.38978240302745 Sentence-wise accuracy 8.928571428571429	Token-wise accuracy 84.29517502365185 Token-wise F1 (macro) 83.21108699638205 Token-wise F1 (micro) 84.29517502365185 Sentence-wise accuracy 11.607142857142858

After adding new Features, Learning Rate - 1.0

MEMM	CRF
Token-wise accuracy 87.03878902554399 Token-wise F1 (macro) 85.83153159283303 Token-wise F1 (micro) 87.038789025544 Sentence-wise accuracy 15.178571428571427	Token-wise accuracy 85.99810785241249 Token-wise F1 (macro) 84.44998147992932 Token-wise F1 (micro) 85.99810785241247 Sentence-wise accuracy 11.607142857142858

Precision and Recall Results for twitter_dev.lr.pred:

accuracy: 87.04%; precision: 87.04%; recall: 87.04%; FB1: 87.04

POS	**Precision**	**Recall**
.	95.09%	99.21%
ADJ	77.19%	44.44%
ADP	89.61%	91.39%
ADV	90.20%	71.32%
CONJ	100.00%	92.86%
DET	97.58%	93.08%
NOUN	77.60%	90.40%
NUM	85.71%	70.59%
PRON	96.88%	95.88%
PRT	89.09%	85.96%
VERB	84.59%	86.46%
X	87.56%	81.42%

Observations : As we can see from the above table, precision is maximum for conjunctions. Pronouns also have a higher precision and recall value. Adverbs have a higher precision value and comparatively lower recall value. By definition of recall, it means less less relevant adverbs are selected but higher selected adverbs are relevant.

Precision and Recall Results for twitter_dev.crf.pred:

accuracy: 86.00%; precision: 86.00%; recall: 86.00%; FB1: 86.00

POS	**Precision**	**Recall**
.	96.18%	99.21%
ADJ	62.50%	55.56%
ADP	85.09%	90.73%
ADV	81.31%	67.44%
CONJ	100.00%	92.86%
DET	96.72%	90.77%
NOUN	82.39%	84.97%
NUM	75.00%	70.59%
PRON	95.00%	97.94%
PRT	86.21%	87.72%
VERB	83.20%	86.19%
X	84.00%	80.33%

Observations:

As we can see from the above table, precision is maximum for conjunctions. Pronouns also have a higher precision and recall value. Adverbs have a higher precision value and comparatively lower recall value as LR Model. Numbers have lower precision than LR model. Due to the labelling bias issue in MEMM Models, they tend to give a high accuracy considering only local optima. CRF without the label bias problem and considering the global optima gives a better value. CRF model performs better than MEMM on sentences like *"Games with footballs are fun"* which have plural noun due to the problem of label bias.

HYPER-PARAMETER TUNING

After adding new Features, Learning Rate - 0.1

MEMM	CRF
Token-wise accuracy 87.03878902554399 Token-wise F1 (macro) 85.83153159283303 Token-wise F1 (micro) 87.038789025544 Sentence-wise accuracy 15.178571428571427	Token-wise accuracy 86.23462630085147 Token-wise F1 (macro) 85.10258618545858 Token-wise F1 (micro) 86.23462630085147 Sentence-wise accuracy 13.392857142857142

The CRF token wise accuracy increases with a decrease in learning rate as the model learns more.

REFERENCES: https://en.wikipedia.org/wiki/Viterbi_algorithm#Pseudocode