CSE 538 Assignment 2 – Parts of Speech tagging

Samanvitha Reddy Panyam
112025771

Viterbi Algorithm:

Viterbi algorithm is a dynamic programming problem to get the most likely sequence of tags that has the highest probability. In order to predict this, we require word emission probabilities (probability of emitting a word given a tag) and tag transition probabilities (probability of emitting a tag given certain tag) and start and end transition scores. These are given to us in the form of matrices. We construct a 2D matrix from where words in a sentence are considered as rows and possible tags are considered as columns. Any path through the trellis represents a tag sequence.

Brute force enumeration scores all the possible tag sequences, evaluate all those tag sequences and pick the sequence that gives the highest score for the sentence. This method has repeated calculations and it gives exponential time complexity.

With the Viterbi algorithm which is a dynamic programming solution saves the best path till the word seen so far. At every step, we are only going to extend that best possible sequence. The goal now is to find the best path to a node in the matrix. It can be calculated using the following formula.

$$T(i, y) = \psi x(y, i, x) + max\ y0\ \psi t(y\ 0, y) + T(i - 1, y0\ )$$

In the initial step, since we assume a sentence always starts with a start tag, we just add those start transition scores to the emission scores. Having done that, for every node in the matrix, we calculate the score using the above formula. Having calculated the scores for all the nodes, we add the end transition probabilities to the last row which is the final word in the sentence.

While calculating scores at every node, we keep track of indices to the previous node where the best possible score has occurred so that when back tracked from the node in the last row that has the highest score, we get the tag sequence required.

Features:

Features help in predicting the parts of speech tagging for words given in a sentence. Features can be based independent of the context words or can be dependent on them. Adding efficient features will increase the accuracy of correctly predicting the POS tag. Apart from the basic features, following additional features have been added to

Additional features:

Features added to identify singular NOUN and plural NOUNS:

1. If a word ends with "ness", it has been given the tag of "IS_NOUN"

   Ex: brightness, cleverness, happiness

2. If a word is in its plural form i.e., if the word ends with "es", it has been given the tag of "IS_NOUNS"

   Ex: mangoes , catches

3. If a word ends with "ment", it has been given the tag of "IS_NOUN"

   Ex: amendment, entertainment, advertisement

Features added to identify VERBS:

1. If a word ends with "ing", it has been given the tag of "IS_VERB"

   Ex: eating, sleeping, playing

2. If a word ends with "ies", it has been given the tag of "IS_VERB"

   Ex: cries, tries, flies

3. If a word ends with "ed", it has been given the tag of "IS_VERB"

   Ex: played, laughed, kicked

Features added to identify ADJECTIVE:

1. If a word ends with "est", it has been given the tag of "IS_ADJECTIVE"

   Ex: cutest, fastest, happiest

2. If a word ends with "ous", it has been given the tag of "IS_ ADJECTIVE"

   Ex: generous, nervous, fabulous

3. If a word ends with "ful", it has been given the tag of "IS_ ADJECTIVE"

   Ex: painful, helpful, awful

Features added to identify ADVERB:

1. If a word ends with "ly", it has been given the tag of "IS_ADVERB"

   Ex: happily, furiously, fortunately

Features added to identify PREPOSITION:

1. If a word is in given the list of prepositions, it has been given the tag of "IS_PREPOSITION"

   Ex: in, before, beside, at, for

Features added to identify CONJUNCTION:

1. If a word is in the given list of conjunctions, it has been given the tag of "IS_CONJUNCTION"

   Ex: and, but, or, so

Features added to identify PRONOUN:

1. If a word is in the given list of pronouns, it has been given the tag of "IS_PRONOUN"

   Ex: his, them, me

Features added to identify DETERMINANT:

1. If a word is in the given list of conjunctions, it has been given the tag of "IS_DETERMINANT"

   Ex: the, a, an, this

Features added to identify EMAIL:

1. If a word ends with ".com", it has been given the tag of "IS_EMAIL"

   Ex: xyz@gmail.com, abc@yahoo.com

Features added to identify HASHTAG:

1. If a word starts with "#", it has been given the tag of "IS_HASHTAG"

   Ex: happily, furiously, fortunately

Features added to identify URL:

1. If a word starts with "http", it has been given the tag of "IS_URL"

   Ex: https://www.google.com, https://www.linkedin.com

Features added to identify EMOJI:

1. If a word is in the given list of emojis, it has been given the tag of "IS_EMOJI"

   Ex: ☺, :P, :D

Wordnet:

I have also used wordnet that measures the semantic similarity of 2 words. With this implementation, it tries to group the words that are semantically similar and assign them to a single feature.

Lemmatizer:

Lemmatizer returns the dictionary form or base of the word. It performs morphological analysis in order to identify the lemma for each word. It identifies meaning of the word by also considering the surrounding context in the sentence. Words that have similar lemmas will be grouped together under a feature.

Brown Cluster:

I have used one of the experiment samples of CMU where they have performed brown cluster on twitter data and used that sample to identify features for the words given.

Comparison of performance for Logistic Regression before and after adding features

Basic:

Dev Evaluation	
Token-wise accuracy	84.38978240302744
Token-wise F1 (macro)	83.33422799705717
Token-wise F1 (micro)	84.38978240302745
Sentence-wise accuracy	8.928571428571429
precision	0.94
recall	0.98
f1-score	0.96
support	254

After adding the features, the token-wise accuracy has increased by 4.03% and sentence-wise accuracy has increased by 10%. Other metrics like precision, recall and f1-score have also increased.

Dev Evaluation	
Token-wise accuracy	88.41059602649007
87.28270227988523	87.28270227988523
Token-wise F1 (micro)	88.41059602649007
Sentence-wise accuracy	18.75
precision	0.95
recall	0.99
f1-score	0.97
support	254

Comparison of performance for Conditional Random Fields before and after adding features

<u>CRF:</u>

Basic

Dev Evaluation	
Token-wise accuracy	84.29517502365185
Token-wise F1 (macro)	83.21108699638205
Token-wise F1 (micro)	84.29517502365185
Sentence-wise accuracy	11.607142857142858
precision	0.95
recall	0.98
f1-score	0.97
support	254

After adding the features, the token-wise accuracy has increased approximately by 3.78% and sentence-wise accuracy has increased by 5.35%. Other metrics like precision, recall and f1-score have also increased.

Dev Evaluation	
Token-wise accuracy	88.0794701986755
Token-wise F1 (macro)	86.91823810731684
Token-wise F1 (micro)	88.0794701986755
Sentence-wise accuracy	16.964285714285715
precision	0.97
recall	0.99
f1-score	0.98
support	254

POS tag correctly being identified for sentences:

In the following examples, I have highlighted the identified parts of speech for each word in the sentences.

1.“His activities controlled abusiveness”

His : ['BIAS', 'SENT_BEGIN', u'WORD=His', u'LCASE=his', 'IS_ALNUM', 'IS_PRONOUN', u'Lemmatizer_His', 'NEXT_BIAS', u'NEXT_WORD=activities', u'NEXT_LCASE=activities', 'NEXT_IS_ALNUM', 'NEXT_IS_LOWER', 'NEXT_IS_VERB', 'NEXT_IS_NOUNS', 'NEXT_1101101000', u'NEXT_Lemmatizer_activity', 'NEXT_activity', 'NEXT_action', 'NEXT_bodily_process', 'NEXT_atural_process', 'NEXT_activeness']

activities : ['BIAS', u'WORD=activities', u'LCASE=activities', 'IS_ALNUM', 'IS_LOWER', 'IS_VERB', 'IS_NOUNS', '1101101000', u'Lemmatizer_activity', 'activity', 'action', 'bodily_process', 'atural_process', 'activeness', 'PREV_BIAS', 'PREV_SENT_BEGIN', u'PREV_WORD=His', u'PREV_LCASE=his', 'PREV_IS_ALNUM', 'PREV_IS_PRONOUN', u'PREV_Lemmatizer_His', 'NEXT_BIAS', u'NEXT_WORD=controlled', u'NEXT_LCASE=controlled', 'NEXT_IS_ALNUM', 'NEXT_IS_LOWER', 'NEXT_IS_VERB', 'NEXT_01111110110000', u'NEXT_Lemmatizer_controlled', 'NEXT_control', 'NEXT_operate', 'NEXT_manipulate', 'NEXT_', 'NEXT_master', 'NEXT_controlled']

controlled : ['BIAS', u'WORD=controlled', u'LCASE=controlled', 'IS_ALNUM', 'IS_LOWER', 'IS_VERB', '01111110110000', u'Lemmatizer_controlled', 'control', 'operate', 'manipulate', '', 'master', 'controlled', 'PREV_BIAS', u'PREV_WORD=activities', u'PREV_LCASE=activities', 'PREV_IS_ALNUM', 'PREV_IS_LOWER', 'PREV_IS_VERB', 'PREV_IS_NOUNS', 'PREV_1101101000', u'PREV_Lemmatizer_activity', 'PREV_activity', 'PREV_action', 'PREV_bodily_process', 'PREV_atural_process', 'PREV_activeness', 'NEXT_BIAS', 'NEXT_SENT_END', u'NEXT_WORD=abusiveness', u'NEXT_LCASE=abusiveness', 'NEXT_IS_ALNUM', 'NEXT_IS_LOWER', 'NEXT_IS_NOUN', u'NEXT_Lemmatizer_abusiveness']

abusiveness : ['BIAS', 'SENT_END', u'WORD=abusiveness', u'LCASE=abusiveness', 'IS_ALNUM', 'IS_LOWER', 'IS_NOUN', u'Lemmatizer_abusiveness', 'PREV_BIAS', u'PREV_WORD=controlled', u'PREV_LCASE=controlled', 'PREV_IS_ALNUM', 'PREV_IS_LOWER', 'PREV_IS_VERB', 'PREV_01111110110000', u'PREV_Lemmatizer_controlled', 'PREV_control', 'PREV_operate', 'PREV_manipulate', 'PREV_', 'PREV_master', 'PREV_controlled']

2.“They abruptly stopped talking”

They : ['BIAS', 'SENT_BEGIN', u'WORD=They', u'LCASE=they', 'IS_ALNUM', 'IS_PRONOUN', u'Lemmatizer_They', 'NEXT_BIAS', u'NEXT_WORD=abruptly', u'NEXT_LCASE=abruptly', 'NEXT_IS_ALNUM', 'NEXT_IS_LOWER', 'NEXT_IS_ADVERB', u'NEXT_Lemmatizer_abruptly', 'NEXT_abruptly']

abruptly : ['BIAS', u'WORD=abruptly', u'LCASE=abruptly', 'IS_ALNUM', 'IS_LOWER', 'IS_ADVERB', u'Lemmatizer_abruptly', 'abruptly', 'PREV_BIAS', 'PREV_SENT_BEGIN', u'PREV_WORD=They', u'PREV_LCASE=they', 'PREV_IS_ALNUM', 'PREV_IS_PRONOUN', u'PREV_Lemmatizer_They', 'NEXT_BIAS', u'NEXT_WORD=stopped', u'NEXT_LCASE=stopped', 'NEXT_IS_ALNUM', 'NEXT_IS_LOWER', 'NEXT_IS_VERB', 'NEXT_01111110110010', u'NEXT_Lemmatizer_stopped', 'NEXT_op', 'NEXT_discontinue', 'NEXT_break', 'NEXT_check', 'NEXT_intercept', 'NEXT_d', 'NEXT_barricade', 'NEXT_hold_on', 'NEXT_opped']

stopped : ['BIAS', u'WORD=stopped', u'LCASE=stopped', 'IS_ALNUM', 'IS_LOWER', 'IS_VERB', '01111110110010', u'Lemmatizer_stopped', 'op', 'discontinue', 'break', 'check', 'intercept', 'd', 'barricade', 'hold_on', 'opped', 'PREV_BIAS', u'PREV_WORD=abruptly', u'PREV_LCASE=abruptly', 'PREV_IS_ALNUM', 'PREV_IS_LOWER', 'PREV_IS_ADVERB', u'PREV_Lemmatizer_abruptly', 'PREV_abruptly', 'NEXT_BIAS', 'NEXT_SENT_END', u'NEXT_WORD=talking', u'NEXT_LCASE=talking', 'NEXT_IS_ALNUM', 'NEXT_IS_LOWER', 'NEXT_IS_VERB', 'NEXT_011111101011111', u'NEXT_Lemmatizer_talking', 'NEXT_alk', 'NEXT_peak', 'NEXT_pill', 'NEXT_pill_the_beans', 'NEXT_lecture']

talking : ['BIAS', 'SENT_END', u'WORD=talking', u'LCASE=talking', 'IS_ALNUM', 'IS_LOWER', 'IS_VERB', '011111101011111', u'Lemmatizer_talking', 'alk', 'peak', 'pill', 'pill_the_beans', 'lecture', 'PREV_BIAS', u'PREV_WORD=stopped', u'PREV_LCASE=stopped', 'PREV_IS_ALNUM', 'PREV_IS_LOWER', 'PREV_IS_VERB', 'PREV_01111110110010', u'PREV_Lemmatizer_stopped', 'PREV_op', 'PREV_discontinue', 'PREV_break', 'PREV_check', 'PREV_intercept', 'PREV_d', 'PREV_barricade', 'PREV_hold_on', 'PREV_opped']

Sentences where CRF is better than MEMM:

“His gameplay has greatly improved”

In this sentence, ‘His’ is a possessive adjective. LR classifies this as a pronoun but when CRF is used, it is tagged as an adjective which is correct. So, for these kind of sentences where the words are contextually dependent, CRF performs better and efficiently identifies the tag sequences.

Comparison of Logistic Regression and CRF taggers:

In 'Logistic Regression' model, tagging a word with a POS tag is independent of the context words. So, it is usually faster compared to the Conditional Random Fields model which is dependent on the neighboring words. CRF calculates the probabilities for every state by saving the best possible tag sequence to arrive to that state. Basically, CRFs are the sequential version of logistic regression

	Accuracy types	Logistic Regression	CRF Tagger
Basic features	Token-wise accuracy	84.38978240302744	84.29517502365185
	Sentence-wise accuracy	8.928571428571429	11.607142857142858
Additional features	Token-wise accuracy	88.41059602649007	88.0794701986755
	Sentence-wise accuracy	16.964285714285715	16.964285714285715

For the base case where the number of iterations is 25, LR gives better accuracy where as when the number of iterations increased to 35, CRF gives better accuracy

Modifying the hyper-parameters:

Following are the observations made by changing the number of iterations.

Base case:
Number of iterations = 25

Dev Evaluation	
Token-wise accuracy	88.0794701986755
Token-wise F1 (macro)	86.91823810731684
Token-wise F1 (micro)	88.0794701986755
Sentence-wise accuracy	16.964285714285715

Number of iterations = 30

Dev Evaluation	
Token-wise accuracy	88.36329233680227
Token-wise F1 (macro)	87.41632113576611
Token-wise F1 (micro)	88.36329233680227
Sentence-wise accuracy	17.857142857142858

Number of iterations = 35

Dev Evaluation	
Token-wise accuracy	88.41059602649007
Token-wise F1 (macro)	87.62930119605788
Token-wise F1 (micro)	88.41059602649007
Sentence-wise accuracy	17.857142857142858

Number of iterations = 15

Dev Evaluation	
Token-wise accuracy	88.1267738883633
Token-wise F1 (macro)	86.79695422319588
Token-wise F1 (micro)	88.1267738883633
Sentence-wise accuracy	15.178571428571427

Number of iterations = 20

Dev Evaluation	
Token-wise accuracy	88.31598864711448
Token-wise F1 (macro)	87.11496919041748
Token-wise F1 (micro)	88.31598864711448
Sentence-wise accuracy	17.857142857142858

Number of iterations = 40

Dev Evaluation	
Token-wise accuracy	88.31598864711448
Token-wise F1 (macro)	87.49060442208084
Token-wise F1 (micro)	88.31598864711448
Sentence-wise accuracy	16.964285714285715

Number of iterations = 45

Dev Evaluation	
Token-wise accuracy	88.22138126773889
Token-wise F1 (macro)	87.19797162515319
Token-wise F1 (micro)	88.22138126773889
Sentence-wise accuracy	16.964285714285715

Number of iterations = 50

Dev Evaluation	
Token-wise accuracy	88.0794701986755
Token-wise F1 (macro)	87.04192674298073
Token-wise F1 (micro)	88.0794701986755
Sentence-wise accuracy	16.964285714285715

Number of iterations = 55

Dev Evaluation	
Token-wise accuracy	87.9848628192999
Token-wise F1 (macro)	86.82280954801514
Token-wise F1 (micro)	87.9848628192999
Sentence-wise accuracy	15.178571428571427

Number of iterations = 60

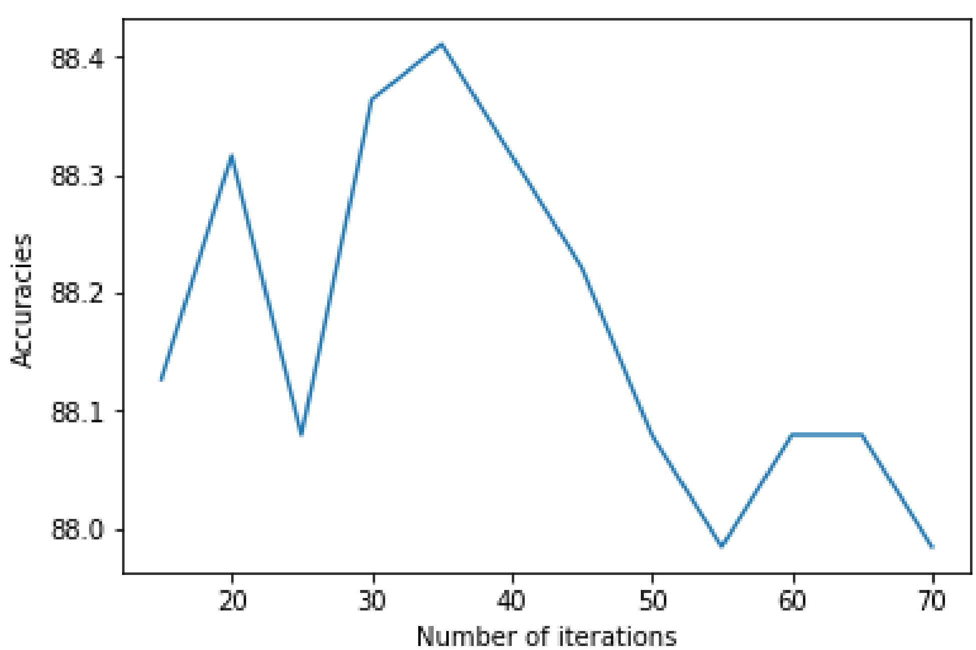
Dev Evaluation	
Token-wise accuracy	88.0794701986755
Token-wise F1 (macro)	86.94426496862974
Token-wise F1 (micro)	88.0794701986755
Sentence-wise accuracy	15.178571428571427

Number of iterations = 65

Dev Evaluation	
Token-wise accuracy	88.0794701986755
Token-wise F1 (macro)	86.98109846635114
Token-wise F1 (micro)	88.0794701986755
Sentence-wise accuracy	15.178571428571427

Number of iterations = 70

Dev Evaluation	
Token-wise accuracy	87.9848628192999
Token-wise F1 (macro)	86.93984803988862
Token-wise F1 (micro)	87.9848628192999
Sentence-wise accuracy	15.178571428571427



Best accuracy method is CRF Tagger with a token accuracy 88.41% at 35 iterations.