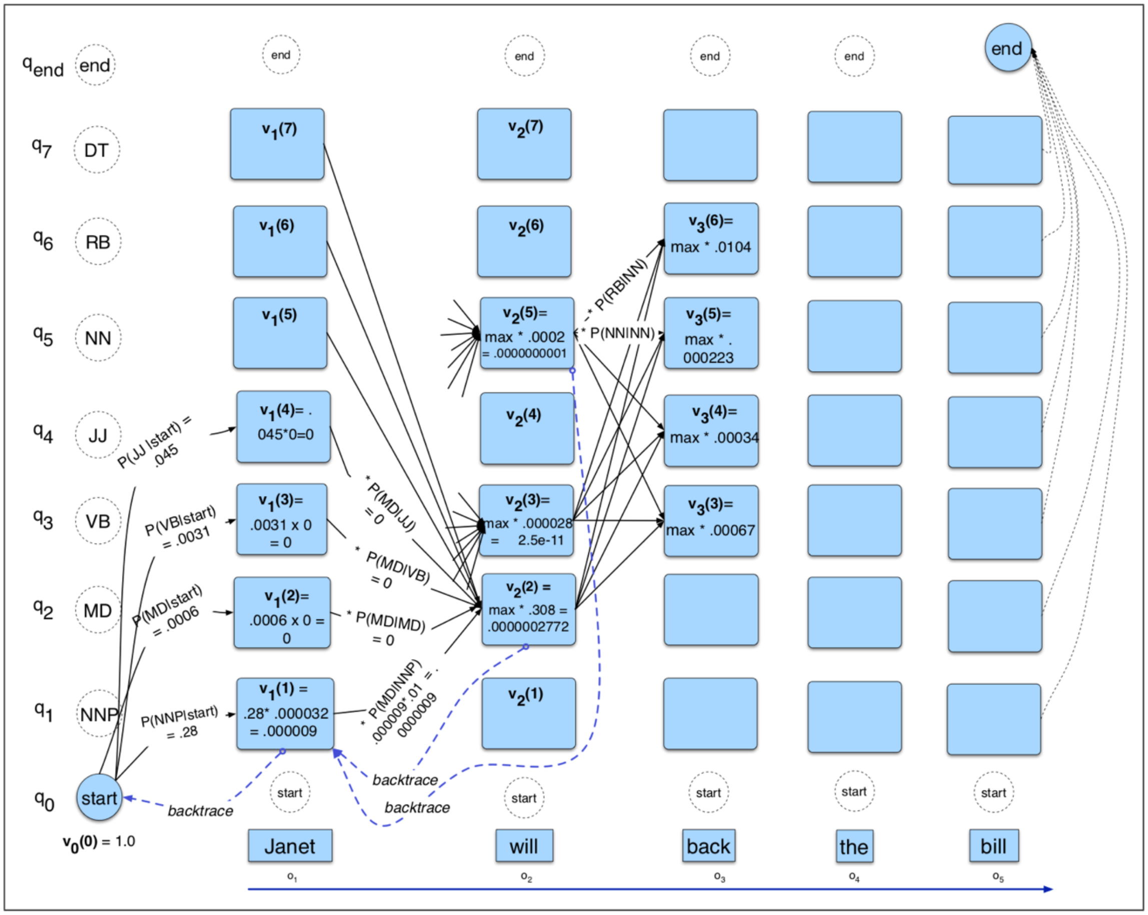
Assignment 2: Part of Speech Tagging

Niranjan Balsubramanian and Jun Kang - CSE 628 Spring 2018
Name: Aishwarya Danoji
SBU ID: 111493647

1) Viterbi Algorithm Implementation:



Pictorial view of each step in Viterbi algorithm-here q0-q7 are POS tags, v_y(i) is same as T (i, y) that maintains the score of the best sequence from 1 . . . i such that $y_i = y$.

Steps:

-First sets up a probability matrix, with N (length of the sentence) columns and m (number of tags considered) rows.

-Let us consider following sentence for POS tagging:

"I love to play volleyball"

POS tag - PRON ADJ DET VERB NOUN

Here N=5 and m=no. of tags in the vocabulary= 12 (in our case)

We will be using following equation:

$$T(i, y) = \psi_x(y, i, x) + \max \psi_t(y', y) + T(i-1, y')$$

Here,
$T(i, y)$ - the score of the best sequence from 1 . . . i such that $y_i = y$.
$\psi_x(y, i, x)$- emission probability or the state observation likelihood of the observation word x given the POS tag y.
$\psi_t(y', y)$ - transition probability of moving from tag y' to y.
$T(i\text{-}1, y')$- the previous Viterbi path probability from the previous time step.

- We begin in the first column by setting the Viterbi value in each cell to the product of the transition probability (into it from the start state) and the observation probability (of the first word.

-Then we move on, column by column; for every state in column 1, we compute the probability of moving into each state in column 2, and so on. For each state *y* at time *t*, we compute the value T (i, y) by taking the maximum over the extensions of all the paths that lead to the current cell.

- After the cells are filled in, backtracking from the *end* state, we should be able to reconstruct the correct POS tagged sequence for the given sentence.

2) Description of Features added

Feature code	**Description**	**Example words**	**Feature name**
`if word.isdigit() or w2n.word_to_num(word).isdigit()`	To check if the string of word or the word itself is a number. Used word2number in python	One, twenty, hundred	**IS_DIGIT**
`if wordnet.synsets(word)`	To check if the word is a valid word in vocabulary . Used nltk.corpus in python	Non-valid words: Okay U	**IS_VALID**
`v=['ed','ing','s','d','ies','es']` `if word[-2:] in v`	To check if the word ends with common verb endings	Baking, baked, bakes	**HAS_VCONJ**
`if word[-2:] is "er"` `or word[0:2] is "un"`	To check if the word ends with 'er' or starts with 'un', common adjective extensions	unclean, undesirable unkind bigger smarter better	**HAS_ADJCONJ**
`adv=['ly','ful']` `if word[-2:] in adv`	To check if word ends with common adverb endings like	perfectly sadly	**HAS_AVCONJ**

	-ly . It’s a quick way to transform adjective into adverb		
`if word[0].isupper()`	To check if the word starts with Capital letter, most Noun starts with capital letter	Emma Germany Morgan Stanley	**CAP_START**
`if "#" in word or "@" in word`	To check if the word contains # or @ in it.	@ves.ac.in #hello	**HAS_#@**

`conj=['and','if','as','but',//`
`'or','while','as']`
`if word in conj`
	To check if the word is a conjunction		**IS_CONJ**

`if "'" in word`	to check if the word contains ‘	‘s, n’t	**HAS_’**
`if word[-1:] is "s"`	To check if the word is singular or plural, Nouns are usually plural	Words Girls Boys	**IS_PLURAL**
`if word in string.punctuation`	To check if the word is a punctuation. Used function from string	! , : ;	**IS_PUNCT**
`for pf in token2features(sent, i-2, add_neighs = False)` `for pf in token2features(sent, i+2, add_neighs = False)`	To include more neighboring words to make a tag decision		**PPREV_** **NNEXT_**
`if len(sent)>10`	To check if the length of the sentence whose POS tag is to be found is greater than 10		**IS_LONG**
`det=['the’,'a','an']` `if word in det`	To check if the word is among the common determinants in english		**IS_DET**
`if '-' in word`	To check if the word contains -	Un-neceassary	**HAS_-**

Other features tried that did not work:

- **Browns clustering –**

It is a hard hierarchical agglomerative clustering problem. It is typically applied to text, grouping words into clusters that are assumed to be semantically related by virtue of their having been embedded in similar contexts. Thus, it chooses the POS tag for a word based on the internal makeup of the word. The algorithm gives you a tree and you need to truncate it at some level to get clusters.

It did not work because:
- the word itself cannot be used to form a cluster. As same word can sometimes get different tags based on the context.
-Also, for this to works the words in our training set must be valid English words and also the vocabulary size show be large enough. Both these conditions are not satisfied in the twitter data set that we have.

- **Stemming:**

It is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Thus, stemming usually refers to crude heuristic process that chops off the ends of the words in the hope of achieving this goal correctly most of the time and often includes the removal of derivational suffix like er, ing, ly, s etc.

Eg -run-verb/noun is the stem word for running-Verb, runs, runner-Noun etc.
This did not word because if we replace running and runner with run in the dataset the POS tags obtained for the original word will be invalid.

- **Lemmatization-**

It usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove influential endings only and to return the base or dictionary form of a word, which is known as lemma. Even this does not work because of the same reason as mentioned above.

-

3) Comparison of the added features against the basic features

Feature No:	**Feature Set**	**Logistic Regression O/P**	**CRF O/P**	**Comment**
1	**Basic Features:** SENT_BEGIN SENT_END IS_ALNUM IS_NUMERIC IS_UPPER IS_LOWER IS_DIGIT PREV_ NEXT_	Token-wise accuracy 84.389782403 Token-wise F1 (macro) 83.3342279971 Token-wise F1 (micro) 84.389782403 Sentence-wise accuracy 8.92857142857	Token-wise accuracy 84.2951750237 Token-wise F1 (macro) 83.2110869964 Token-wise F1 (micro) 84.2951750237 Sentence-wise accuracy 11.6071428571	This accuracy can be further improved and is considered as minimum accuracy with basic features

2	**Basic Features All newly added features**	Token-wise accuracy 83.7748344371 Token-wise F1 (macro) 82.7307699242 Token-wise F1 (micro) 83.7748344371 Sentence-wise accuracy 14.2857142857	Token-wise accuracy 82.4976348155 Token-wise F1 (macro) 81.7448909809 Token-wise F1 (micro) 82.4976348155 Sentence-wise accuracy 9.82142857143	Overfitting hence less accuracy, code is very hard coded
3	**Basic features +verb:** HAS_VCONJ	Token-wise accuracy 84.247871334 Token-wise F1 (macro) 83.2274347489 Token-wise F1 (micro) 84.247871334 Sentence-wise accuracy 9.82142857143	Token-wise accuracy 84.389782403 Token-wise F1 (macro) 83.6793451427 Token-wise F1 (micro) 84.389782403 Sentence-wise accuracy 12.5	Verb tag accuracy improved a bit
4	**Basic features +adjective +adverb:** HAS_ADJCONJ HAS_AVCONJ	Token-wise accuracy 84.4370860927 Token-wise F1 (macro) 83.6677104231 Token-wise F1 (micro) 84.4370860927 Sentence-wise accuracy 10.7142857143	Token-wise accuracy 84.9101229896 Token-wise F1 (macro) 83.8945109413 Token-wise F1 (micro) 84.9101229896 Sentence-wise accuracy 10.7142857143	Adjective tag and adverb accuracy improved a bit
5	**Basic features +noun:** CAP_START IS_PRURAL HAS_’	Token-wise accuracy 84.4843897824 Token-wise F1 (macro) 83.5089233591 Token-wise F1 (micro) 84.4843897824	Token-wise accuracy 84.1532639546 Token-wise F1 (macro) 83.5658128691 Token-wise F1 (micro) 84.1532639546	Noun and pronoun tag accuracy improved a bit

		Sentence-wise accuracy 8.92857142857	Sentence-wise accuracy 11.6071428571	
6	**Basic features+ Punctuation:** IS_PUNCT HAS_- HAS_' HAS_@#	Token-wise accuracy 85.004730369 Token-wise F1 (macro) 83.8376818468 Token-wise F1 (micro) 85.004730369 Sentence-wise accuracy 9.82142857143	Token-wise accuracy 84.7209082308 Token-wise F1 (macro) 83.2437786437 Token-wise F1 (micro) 84.7209082308 Sentence-wise accuracy 11.6071428571	X tag and . accuracy improved
7	**Basic features+ Word validity:** IS_VALID	Token-wise accuracy 84.4370860927 Token-wise F1 (macro) 83.3281774819 Token-wise F1 (micro) 84.4370860927 Sentence-wise accuracy 8.03571428571	Token-wise accuracy 84.5789971618 Token-wise F1 (macro) 83.8235480166 Token-wise F1 (micro) 84.5789971618 Sentence-wise accuracy 9.82142857143	Overall accuracy improved
8	**Basic features+ More neigh under consideration**	Token-wise accuracy 83.0652790918 Token-wise F1 (macro) 81.8797155615 Token-wise F1 (micro) 83.0652790918 Sentence-wise accuracy 8.92857142857	Token-wise accuracy 81.8826868496 Token-wise F1 (macro) 81.2051092685 Token-wise F1 (micro) 81.8826868496 Sentence-wise accuracy 8.03571428571	Accuracy decreased due to overfitting
914.	**Best features set: Basic features +** IS_DIGIT, IS_PLURAL IS_PUNCT IS_VALID	Token-wise accuracy **85.2412488174** Token-wise F1 (macro) 84.3351365943	Token-wise accuracy **85.6196783349** Token-wise F1 (macro) 85.0400365421	Best model so far. Highest accuracy

	HAS_VCONJ HAS_ADJCONJ HAS_AVCONJ HAS_@# HAS_' CAP_START	Token-wise F1 (micro) 85.2412488174 Sentence-wise accuracy **15.1785714286**	Token-wise F1 (micro) 85.6196783349 Sentence-wise accuracy **15.1785714286**	

OUTPUT FOR BEST MODEL:

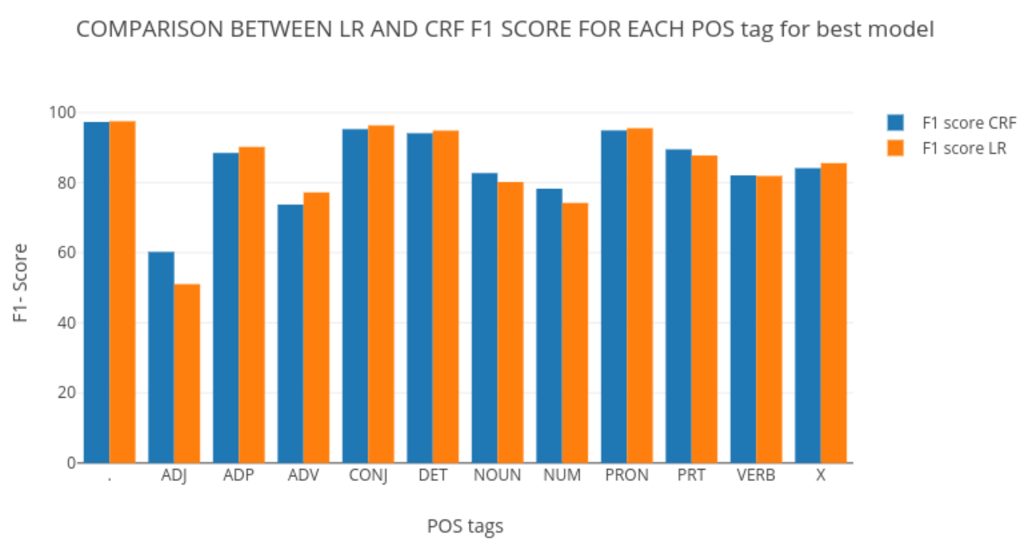
LOGISTIC REGRESSION:

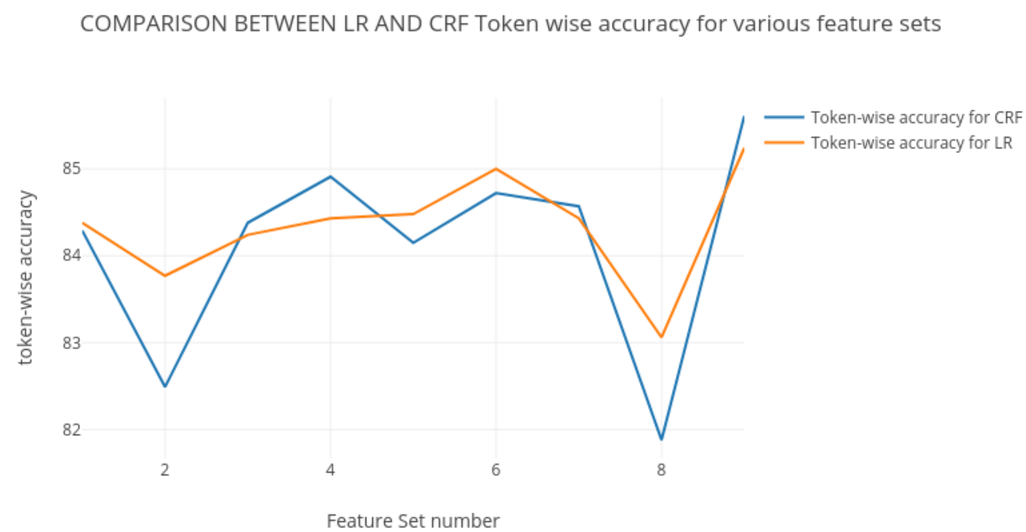
```
### Dev evaluation
Token-wise accuracy 85.2412488174
Token-wise F1 (macro) 84.3351365943
Token-wise F1 (micro) 85.2412488174
Sentence-wise accuracy 15.1785714286
             precision    recall  f1-score   support

          .       0.96      0.99      0.97       254
        ADJ       0.80      0.37      0.51        99
        ADP       0.92      0.88      0.90       151
        ADV       0.89      0.68      0.77       129
       CONJ       1.00      0.93      0.96        42
        DET       0.98      0.92      0.95       130
       NOUN       0.73      0.89      0.80       479
        NUM       0.82      0.68      0.74        34
       PRON       0.98      0.93      0.96       194
        PRT       0.88      0.88      0.88        57
       VERB       0.80      0.84      0.82       362
          X       0.89      0.83      0.86       183

avg / total       0.86      0.85      0.85      2114

(/Users/aishwarya/anaconda2) Aishwaryas-MacBook-Pro:Assignment2 aishwarya$ ./conlleval.pl -r -d \\t < ./predictions/twitter_dev.lr.pred
processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1802.
accuracy:  85.24%; precision:  85.24%; recall:  85.24%; FB1:  85.24
                .: precision:  95.82%; recall:  99.21%; FB1:  97.49  263
              ADJ: precision:  80.43%; recall:  37.37%; FB1:  51.03  46
              ADP: precision:  92.36%; recall:  88.08%; FB1:  90.17  144
              ADV: precision:  88.89%; recall:  68.22%; FB1:  77.19  99
             CONJ: precision: 100.00%; recall:  92.86%; FB1:  96.30  39
              DET: precision:  98.35%; recall:  91.54%; FB1:  94.82  121
             NOUN: precision:  72.95%; recall:  88.94%; FB1:  80.15  584
              NUM: precision:  82.14%; recall:  67.65%; FB1:  74.19  28
             PRON: precision:  97.84%; recall:  93.30%; FB1:  95.51  185
              PRT: precision:  87.72%; recall:  87.72%; FB1:  87.72  57
             VERB: precision:  80.16%; recall:  83.70%; FB1:  81.89  378
                X: precision:  88.82%; recall:  82.51%; FB1:  85.55  170
```

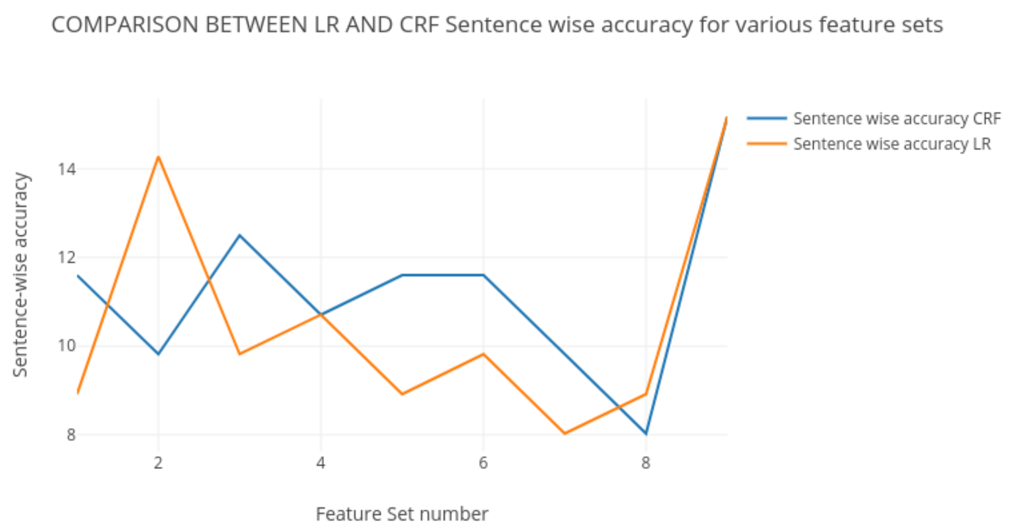
CRF:

```
### Dev evaluation
Token-wise accuracy 85.6196783349
Token-wise F1 (macro) 85.0400365421
Token-wise F1 (micro) 85.6196783349
Sentence-wise accuracy 15.1785714286
             precision    recall  f1-score   support

          .       0.96      0.98      0.97       254
        ADJ       0.69      0.54      0.60        99
        ADP       0.88      0.89      0.88       151
        ADV       0.81      0.67      0.74       129
       CONJ       0.95      0.95      0.95        42
        DET       0.97      0.92      0.94       130
       NOUN       0.80      0.85      0.83       479
        NUM       0.77      0.79      0.78        34
       PRON       0.94      0.95      0.95       194
        PRT       0.89      0.89      0.89        57
       VERB       0.80      0.84      0.82       362
          X       0.86      0.83      0.84       183

avg / total       0.86      0.86      0.85      2114

(/Users/aishwarya/anaconda2) Aishwaryas-MacBook-Pro:Assignment2 aishwarya$ ./conlleval.pl -r -d \\t < ./predictions/twitter_dev.crf.pred
processed 2114 tokens with 2114 phrases; found: 2114 phrases; correct: 1810.
accuracy:  85.62%; precision:  85.62%; recall:  85.62%; FB1:  85.62
                .: precision:  96.15%; recall:  98.43%; FB1:  97.28  260
              ADJ: precision:  68.83%; recall:  53.54%; FB1:  60.23  77
              ADP: precision:  88.16%; recall:  88.74%; FB1:  88.45  152
              ADV: precision:  81.31%; recall:  67.44%; FB1:  73.73  107
             CONJ: precision:  95.24%; recall:  95.24%; FB1:  95.24  42
              DET: precision:  96.75%; recall:  91.54%; FB1:  94.07  123
             NOUN: precision:  80.20%; recall:  85.39%; FB1:  82.71  510
              NUM: precision:  77.14%; recall:  79.41%; FB1:  78.26  35
             PRON: precision:  94.39%; recall:  95.36%; FB1:  94.87  196
              PRT: precision:  89.47%; recall:  89.47%; FB1:  89.47  57
             VERB: precision:  80.21%; recall:  83.98%; FB1:  82.05  379
                X: precision:  85.80%; recall:  82.51%; FB1:  84.12  176
(/Users/aishwarya/anaconda2) Aishwaryas-MacBook-Pro:Assignment2 aishwarya$
```

4) Comparison of Logistic Regression and CRFs







Thus we can conclude that CRF gives better overall accuracy both sentence and token-wise than logistic regression as it uses more information to find the POS tag for a given sentence.

References:

1. https://web.stanford.edu/~jurafsky/slp3/10.pdf - for Viterbi image
2. https://en.wikipedia.org/wiki/Brown_clustering
3.https://stackoverflow.com/questions/1787110/what-is-the-true-difference-between-lemmatization-vs-stemming
4.Plotly to draw graphs