NATURAL LANGUAGE PROCESSING FALL 2018
ASSIGNMENT 2 REPORT

1. Description of Viterbi Decoding

We are given a set of words x_1 to x_N and a set of tags t_1 to t_L. For the implementation of Viterbi encoding, the most important data structure to maintain is the 'score' matrix which is the record of the best score along a certain path (of tags t_j) up till a certain word x_i. We also maintain a matrix 'backpointers' which keeps track of the previous score that was used to compute a particular score by storing the index j of the tag that was used.

To compute scores for the first word x_1, we add the start transition scores of a tag with the emission scores of the tag corresponding to the first word. For all further words x_i from the second word to the last word, we compute the score of a particular tag being assigned to a particular word x_i as the sum of the score of the path up till the t_j^{th} tag of the previous word, the transition score from the previous tag t_j to the current tag and the emission score of the current tag with the current word x_i. We compute these scores for all L tags. We then identify which score out of all L scores is maximum and choose this score to be the score corresponding to the current tag of the current word. We store this score in the 'scores' matrix and the index j of tag t_j which yielded maximum score in the 'backpointers' matrix.

For the final row, we add end transition scores corresponding to each tag. We then identify the maximum of all the values in this row. This maximum value is the score of the best sequence. We now use the 'backpointers' matrix to retrieve the path taken by this best sequence.

There was no real issue or challenge that cropped up during this phase of the assignment. With a clear understanding of Viterbi encoding as explained in class and interpreting the equation given in the assignment PDF correctly, the implementation of Viterbi encoding becomes fairly simple. Using correct indices to retrieve entries from the emission_scores and trans_scores matrices is something that needs to be done with care.

2. Description of Feature Generation

The features provided were basic features. Therefore, I tried to generate additional features that captured certain properties of language as well as ones that captured certain properties of the dataset provided. The features were generated with the aim of not making the operations computationally expensive. The features generated for the task are:

a) IS_HASHTAG: Since the data used for our task is Twitter data, we try to include certain Twitter-exclusive features. A hashtag is a word preceded by a '#' symbol. The hashtag is popularly used on Twitter to make a reference to something the tweet is referring to or talking about and so it is sure to occur many times in the train data. Because of the special character, all the hashtagged words would most likely be classified as a specific tag. Thus, including this as a feature would help the tagger.

b) IS_MENTION: This is once again a feature of Twitter data. A word preceded by a '@' symbol is referred to as a mention. It is a popular Twitter feature used to tag (reference) another Twitter user in a tweet. Because of the special character, all the mentions would most likely be classified as a specific tag. Thus, including this feature could prove highly useful for the tagger.

c) IS_PUNC: This feature is used to denote whether the particular word is a punctuation or not. Punctuations are given a specific tag and so this feature would prove to be useful to the tagger.

d) IS_LINK: Tweets often include URLs that link to pictures, news articles and other outside sources. These links will always start with the string 'http'. It is likely that the tagger will classify all these links to the same POS tag, so encoding the occurrence of a link as a feature could prove to be helpful.

e) IS_PROPER: This feature is used to denote whether the first letter of the word is capitalized. When the first letter of the word is capitalized, it is highly likely that the word is a proper noun. The first letter of a word could also be capitalized in a title.

f) IS_EMOTICON: Since our textual data is from Twitter, a social media site, there will be many instances of emoticons throughout the data. Our intuition is that the tagger will tag these emoticons in a similar manner, and so identifying these emoticons could greatly help the tagger.

g) IS_STOPWORD: Stopwords are words that are present in textual data that add no true information to the understanding of the sentence. While performing textual analysis of data, these words are often removed. These stopwords are often prepositions or determiners so including a feature that indicates which words are stopwords could possibly help the tagger identify tags correctly.

h) IS_ABBR: This feature is used to identify whether a particular word is a commonly used abbreviation, especially in the world of texting and social media. These abbreviations tend to be nouns so the tagger might benefit by possessing such information.

i) IS_SLANG: Since our data is Twitter textual data, there will be several slang words used in the tweets. Slang words are informal words used in texting and social media. Intuitively, these words would be verbs or nouns and so having this information might be useful to the tagger.

j) ENDS_WITH_ING: In general, words that end with 'ing' tend to be verbs. Therefore, by identifying such words and making the knowledge available to the tagger, it might possibly make better tag predictions.

k) ENDS_WITH_LY: In general, words that end with 'ly' tend to be adverbs. Therefore, by identifying such words and making the knowledge available to the tagger, it might possibly make better tag predictions.

I conducted several tests to try to identify useful features by adding single features to additional features and training the model as well as by removing features one by one from the full set of features generated. I discarded certain features that did not contribute to improving accuracy our classifiers. More information is available in the tables below.

3. Comparison of New Features against Basic Features

First, I observed the accuracy of the classifiers on the basic features.

Feature	MEMM accuracy (%)	CRF accuracy (%)
Basic features	84.39	84.29

Next, I trained models using the basic features and only one of the generated features at a time. The results of the experiments were as follows:

Feature	MEMM accuracy (%)	CRF accuracy (%)
IS_HASHTAG	84.53	84.48
IS_MENTION	84.77	84.67
IS_PUNC	84.58	84.48
IS_LINK	84.39	84.48
IS_PROPER	84.67	84.53
IS_EMOTICON	84.44	84.11
IS_STOPWORD	84.30	84.53
IS_ABBR	84.39	84.15
IS_SLANG	84.43	84.48
ENDS_WITH_ING	84.34	85.19
ENDS_WITH_LY	84.91	84.43

I then trained the classifier on all the generated features.

Feature	MEMM accuracy (%)	CRF accuracy (%)
All basic + generated features	85.19	85.43

With a lot of trial and error and choosing to remove certain features based loosely on the information above, I conducted several experiments. Below are a few results -

Removed features	MEMM accuracy (%)	CRF accuracy (%)
Abbreviations	85.09	85.66
Abbreviations, Emoticons	84.95	85.33
Stopwords	85.38	85.43

The combination of features that gave the highest accuracy is:

Features	MEMM accuracy (%)	CRF accuracy (%)
Basic + IS_HASHTAG + IS_PUNC + IS_MENTION + IS_LINK + IS_PROPER + IS_EMOTICON + IS_SLANG + ENDS_WITH_ING + ENDS_WITH_LY	85.29	85.67

I attempted to tune parameters for CRF. I changed the number of iterations for which the model was trained to 50 from 25. This increased the accuracy of the CRF classifier.

Change	MEMM accuracy (%)	CRF accuracy (%)
Changed max_iter = 25 to 50	85.29	85.86

4. Comparison of MEMM vs CRF

For our basic features, we can observe from the table above that MEMM yields a higher accuracy than CRF by 0.1%. However, by introducing more relevant and informative features in the feature generation step, the accuracy of both the models increased, but CRF ended up yielding a higher accuracy. After tuning hyperparameters, CRF had a 85.86% accuracy which was 0.57% better than MEMM, which yielded an accuracy of 85.28%.

```
Really  ADV    ADV
hope   VERB   VERB
I  PRON   PRON
can    VERB   VERB
get    VERB   VERB
```

```
to PRT  PRT
@glasgowfilm  X  X
for  ADP  ADP
Winter NOUN  NOUN
's PRT  PRT
Bone  NOUN  NOUN
-  .  .
need  VERB  VERB
to PRT  PRT
get  VERB  VERB
on ADP  ADP
with  ADP  ADP
job  NOUN  NOUN
applications  NOUN  NOUN
tonight  NOUN  NOUN
then  ADV  ADP
!  .  .
Trailer  NOUN  NOUN
:  .  .
http://bit.ly/bhUlum  X  X
```

Above we can see an example of the tag predictions for different words as done by the CRF classifier. Below we'll see the same prediction for the MEMM classifier.

```
Really  ADV  ADV
hope  VERB  VERB
I  PRON  PRON
can  VERB  VERB
get  VERB  VERB
to PRT  PRT
@glasgowfilm  X  X
for  ADP  ADP
Winter NOUN  NOUN
's PRT  VERB
Bone  NOUN  NOUN
-  .  .
need  VERB  VERB
```

```
to PRT    PRT
get    VERB   VERB
on ADP    ADP
with   ADP    ADP
job    NOUN   NOUN
applications   NOUN   NOUN
tonight    NOUN   NOUN
then   ADV    ADV
!  .  .
Trailer    NOUN   NOUN
:  .  .
http://bit.ly/bhUlum   X  X
```

Thus, we see the MEMM classifier makes one incorrect tag prediction while the CRF tag makes all correct predictions. Error may creep into the MEMM classifier possibly because of label bias issues which is eliminated in CRF by global normalization.

We can also see that including our generated features improved our accuracy by 1-1.5% in our classifiers. Thus, with the help of our generated features, the classifiers were able to assign the tags better. Below is an example of the same-

With basic features:-

```
Going   VERB   VERB
to PRT    PRT
start  VERB   VERB
#HelloMornings X  .
on ADP    ADP
Oct    NOUN   NOUN
1  NUM    NUM
-  .  .
Check  VERB   VERB
it PRON   PRON
out    PRT    PRT
on ADP    ADP
@michellebygrace   X  X
```

With basic + generated features:-

```
Going   VERB   VERB
to PRT    PRT
start  VERB   VERB
#HelloMornings X  X
on ADP    ADP
Oct    NOUN   NOUN
1  NUM    NUM
-  .  .
Check  VERB   VERB
it PRON   PRON
out    PRT    PRT
on ADP    ADP
@michellebygrace   X  X
```

We see that by including the IS_HASHTAG feature, the tagger assigns the correct tag to #HelloMornings in the above example.