**NATURAL LANGUAGE PROCESSING FALL 2018**

**ASSIGNMENT 2 REPORT**

1. **Description of Viterbi Decoding**

We are given a set of words x1 to xN and a set of tags t1 to tL. 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 tj) up till a certain word xi. 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 x1, 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 xi from the second word to the last word, we compute the score of a particular tag being assigned to a particular word xi as the sum of the score of the path up till the tjth tag of the previous word, the transition score from the previous tag tj to the current tag and the emission score of the current tag with the current word xi. 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 tj 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.

1. **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:

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.

1. **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 |

1. **Comparison of MEMM vs CRF**