**Machine Learning Project**

Group members: Koh Kai Wei, Chan Wei Ren, Tenzin Chan

All code can be run just by typing the command “python part<x>.py”, where <x> is an integer from 2 to 5 (representing code for the different parts of the project), while in the root directory of the submission.

In this report, “observation” refers to a word and “label” or “tag” refers to its respective label.

Part 2

The implementation for part 2 is straightforward. To calculate emissions, we just count the total number of each observation-label pair and divide that by the number of occurrences of that label.

To replace rare words with #UNK#, we simply counted the number of times each observation occurred, and if it was less than k, all the counts of observation-label pairs containing that observation would be added to the #UNK#-label pair count of that label. To get the emission, the observation-label pair counts are divided by the number of occurrences of the label.

To predict, if the observation is not in the list of all words in the emissions table, the observation will be replaced with #UNK#. Then, the observation-label pair with the highest emission value where the observation occurred will be picked.

Output:

CN

Entities in gold data : 362

Entities in prediction: 3318

entities

- Correct : 183

- Precision : 0.0552

- Recall : 0.5055

- F score : 0.0995

sentiment

- Correct : 57

- Precision : 0.0172

- Recall : 0.1575

- F score : 0.0310

EN

Entities in gold data : 226

Entities in prediction: 1201

entities

- Correct : 165

- Precision : 0.1374

- Recall : 0.7301

- F score : 0.2313

sentiment

- Correct : 71

- Precision : 0.0591

- Recall : 0.3142

- F score : 0.0995

FR

Entities in gold data : 223

Entities in prediction: 1149

entities

- Correct : 182

- Precision : 0.1584

- Recall : 0.8161

- F score : 0.2653

sentiment

- Correct : 68

- Precision : 0.0592

- Recall : 0.3049

- F score : 0.0991

SG

Entities in gold data : 1382

Entities in prediction: 6599

entities

- Correct : 794

- Precision : 0.1203

- Recall : 0.5745

- F score : 0.1990

sentiment

- Correct : 315

- Precision : 0.0477

- Recall : 0.2279

- F score : 0.0789

Part 3

To get the transition parameters, all the combinations of label1 to label2 were counted. This was done by looping through the training set’s tweets and then through the observation-label pairs in each tweet. The START and STOP symbols were added at the start and end of each tweet. Then, the counts for each pair was divided by the number of times label1 occurred.

We implemented a recursive form of the Viterbi algorithm, with the π function taking in k and v, the index of the observation and the label of the observation respectively. When k is 1, the π function returns the product of the emission value of the first observation from the label and the transition from START to the label. Otherwise, it would return the maximum of the products of a recursive call to itself on k-1 and u, the transition value from u to v and the emission value of the observation at index k from v, for every u in the set of possible labels.

To counter numerical underflow, the π function is set to multiply the output by 100 at every call to it.

We also implemented a backtrack function which would return the most probable label for the observation at the index k. To predict, this function is called starting with k as the length of the tweet and v as the STOP symbol. Each label is appended to a list of labels and the backtrack function is called on each previous observation in the tweet. The reversed list of labels is then returned.

Output:

CN

Entities in gold data : 362

Entities in prediction: 226

entities

- Correct : 65

- Precision : 0.2876

- Recall : 0.1796

- F score : 0.2211

sentiment

- Correct : 47

- Precision : 0.2080

- Recall : 0.1298

- F score : 0.1599

EN

Entities in gold data : 226

Entities in prediction: 162

entities

- Correct : 104

- Precision : 0.6420

- Recall : 0.4602

- F score : 0.5361

sentiment

- Correct : 64

- Precision : 0.3951

- Recall : 0.2832

- F score : 0.3299

FR

Entities in gold data : 223

Entities in prediction: 166

entities

- Correct : 112

- Precision : 0.6747

- Recall : 0.5022

- F score : 0.5758

sentiment

- Correct : 72

- Precision : 0.4337

- Recall : 0.3229

- F score : 0.3702

SG

Entities in gold data : 1382

Entities in prediction: 731

entities

- Correct : 386

- Precision : 0.5280

- Recall : 0.2793

- F score : 0.3654

sentiment

- Correct : 244

- Precision : 0.3338

- Recall : 0.1766

- F score : 0.2310

Part 4

We also implemented a recursive form of the forward-backward algorithm to perform max-marginal decoding.

The forward function takes in a label u and an index j. When j is 1, the function returns the transition value from the START symbol to u. Otherwise, it returns the sum of the products of a recursive call to itself with inputs v and j-1, the transition value from v to u and the emission value of the observation at the (j-1)-th index from v, for every v in the set of possible labels.

The backward function also takes in a label u and an index j. However, instead, when j is n, the function returns the product of the transition value from u to the STOP symbol and the emission value of the observation at the j-th index from u. Otherwise, it returns the sum of the products of a recursive call to itself with inputs v and j+1, the transition value from u to v and the emission value of the observation at the j-th index from u, for every v in the set of possible labels.

To predict, the value of Z is calculated, which is the sum of the products of the alpha and beta functions with v and an integer, for every v in the set of possible labels. To verify that the algorithm is working properly, the integer passed in for each tweet is randomly generated from 1 to n (the length of the tweet).

Then, the product of the alpha and beta functions divided by Z is calculated with inputs as a label u and every integer from 1 to n, for every u in the set of possible labels. For each index from 1 to n, the label that produced the highest product of alpha and beta with inputs as that label and that index is chosen and appended to a list. That list is then returned.

Output:

EN

Entities in gold data : 226

Entities in prediction: 175

entities

- Correct : 108

- Precision : 0.6171

- Recall : 0.4779

- F score : 0.5387

sentiment

- Correct : 69

- Precision : 0.3943

- Recall : 0.3053

- F score : 0.3441

FR

Entities in gold data : 223

Entities in prediction: 173

entities

- Correct : 113

- Precision : 0.6532

- Recall : 0.5067

- F score : 0.5707

sentiment

- Correct : 73

- Precision : 0.4220

- Recall : 0.3274

- F score : 0.3687

Part 5

For this part, we optimized the hyperparameter k for replacing unknown words, and then applied the Viterbi algorithm. We found that using unknown symbols for unseen words will depend heavily on the number of observations used to calculate the emission parameters for the unknown symbol. We thus found that k = 2 works well for both EN and FR datasets. We also replaced any observations that looked like websites, emails, friend tags, numbers, years and punctuation with their respective symbols. This would make the algorithm more robust to noise in unseen data.

We also tried implementing a neural network for POS tagging. The neural network was implemented without any library, using lists for weights and deltas for each layer. ReLU units were used for the hidden layers and the Softmax function (normalized sigmoid function over all nodes in the layer) was used for the output layer. The prediction made would be the highest value output by the output layer.

The neural network appeared to work with a single layer of 50 nodes, but eventually converged to 0.9 for the ‘O’ label as the label distribution was extremely skewed. As such, we tried to reduce class imbalance by trying to train the network on examples as equally as possible.

In the end, we chose to use the Viterbi with the optimized hyperparameter k as it performed better than both the max marginal and Viterbi with k set to 3 for the replacement of rare words.

EN

Viterbi best: 0.545455 0.348485

Entities in gold data : 226

Entities in prediction: 170

entities

- Correct : 108

- Precision : 0.6353

- Recall : 0.4779

- F score : 0.5455

sentiment

- Correct : 69

- Precision : 0.4059

- Recall : 0.3053

- F score : 0.3485

FR

Viterbi best: 0.597015 0.407960

Entities in gold data : 223

Entities in prediction: 179

entities

- Correct : 120

- Precision : 0.6704

- Recall : 0.5381

- F score : 0.5970

sentiment

- Correct : 82

- Precision : 0.4581

- Recall : 0.3677

- F score : 0.4080