01.112 Machine Learning Design Project

Li Linwei 1002199, Li Xingxuan 1002189, Wu Xinke 1003935

SUTD

1 Part 2 - HMM Emission

We have emission parameters defined as follows:

$$emission : b(x|y) = \begin{cases} \frac{Count(y->x)}{Count(y)+1} & \text{x in training set.} \\ \frac{1}{Count(y)+1} & \text{x=\#!UNK!\#.} \end{cases}$$

Using only the emission parameters in prediction, we get the following result:

Table 1. F1 scores using emission.

Dataset	F1(Correct Entity)/F1(Correct Label)
FR	0.2751/0.1169
CN	0.1929/0.1134
EN	0.6279/0.4595
SG	0.2956/0.1753

2 Part 3 - HMM Transition and Viterbi Inference

2.1 Transition Parameters

We have transition parameters defined as follows:

$$\begin{aligned} transition : \quad & a(v|u) = \frac{Count(u,v)}{Count(u)} \\ & a(u|START) = \frac{Count(u \text{ at the start of a sentence})}{n} \\ & a(END|u) = \frac{Count(u \text{ at the end of a sentence})}{Count(u)} \\ & n : \text{number of sentences} \\ & u, v : labels \end{aligned}$$

Using only the transition parameters in prediction, we get the following result: The reason for zero F1 score is that the algorithm predicts the same label for all words.

Table 2. F1 scores using transition.

Dataset	F1(Correct Entity)/F1(Correct Label)
FR	0/0
CN	0/0
EN	0/0
SG	0/0

2.2 Viterbi and Prediction

We used logarithm to calculate the Viterbi table, the formula is as follows:

$$\pi[i][j] = max_{k=1,2,..,t}\{\pi[k][j-1] + log[a(y_i|y_k)] + log[b(x_j|y_i])\}$$

And to avoid doing backtrace, we directly remember the best previous label while doing forward calculation:

$$backtrace[i][j] = argmax_{k=1,2,..,t}\{\pi[k][j-1] + log[a(y_i|y_k)] + log[b(x_j|y_i])\}$$

To find the best route, we just need to refer to the backtrace table.

The performance of HMM is significantly better than previous algorithms used:

Table 3. F1 scores using first order HMM.

Dataset	F1(Correct Entity)/F1(Correct Label)
FR	0.3982/0.2225
CN	0.3991/0.2716
EN	0.6617/0.5792
SG	0.4555/0.2862

3 Part 4 - Second Order HMM

3.1 Transition and Emission Parameters

The emission parameters remain the same, but the transition parameters are modified as follows:

$$
\begin{aligned}
transition: \quad & a(w|u,v) = \frac{Count(u,v,w)}{Count(u,v)} \\
& a(u|START, START) = \frac{Count(u \text{ at the start of a sentence})}{n} \\
& a(v|START, u) = \frac{Count(u,v \text{ at the start of a sentence})}{Count(u \text{ at the start of a sentence})} \\
& a(END|u,v) = \frac{Count(u,v \text{ at the end of a sentence})}{Count(u,v)} \\
& n : \text{number of sentences} \\
& u, v : labels
\end{aligned}
$$

3.2 Viterbi and Prediction

We define the Viterbi table as follows:

$$\pi[u][v][j] = max\, P(y_{j-1} = u, y_j = v, x_1, x_2, ..., x_j)$$

We can use dynamic programming to derive the Viterbi table:

$$
\begin{aligned}
& \pi[u][v][j] = max\{\pi[k][u][j-1] * a(v|k,u) * b(x_j|v)\} \\
& x_j : \text{word at index j} \\
& u, v, k : \text{labels}
\end{aligned}
$$

The time complexity of this Viterbi algorithm is

$$O(mt^3) \quad m : \text{length of the sentence} \quad t : \text{number of labels}$$

The performance of second-order HMM dropped compared to the first-order HMM.(Table 4.) There is a performance increase in the French Dataset. However, this increase is not perceived in other languages.

Table 4. F1 scores using HMM.

Dataset	First Order	Second Order
FR	0.3982/0.2225	0.5019/0.2868
CN	0.3991/0.2716	0.3942/0.2680
EN	0.6617/0.5792	0.6573/0.5693
SG	0.4555/0.2862	0.4710/0.3059

4 Part 5 - Design Challenge

4.1 Methodology

In order to achieve a better performance, we decided to use a linear classification algorithm which is structured perceptron. Furthermore, to reduce the bias caused by the specific order of the training examples, we average the weights of different trained models.

4.2 Implementation

To train the structure perceptron model, we process the training set sentence by sentence due to the limitation of the training set. To reduce the bias, we implemented two methods: (1) Average Perceptron, (2) Shuffle and Average.

Structured Perceptron Since we are not allowed to use external library, we decided to train the model as follows:

(1) Pre-process: Convert all the words in the input data into lowercase both during training and testing phase. Specify a threshold k and replace words which appear less than k times with 'UNK'.

(2) Initialization: Initiate two feature matrices for both transition and emission. Set the matrices to 0.

$$transition_{y_{i-1},y_i} : \text{parameter for transition from } y_{i-1} \text{ to } y_i \tag{1}$$

$$emission_{y_i}(x_i) : \text{parameter for emission from } y_i \text{ to } x_i \tag{2}$$

(3) Viterbi algorithm: To find the optimal output of the model on the ith training sentence, we use viterbi algorithm.

$$y_{1,\ldots,n} = \text{argmax} \sum_{i=1}^{n+1} transition_{y_{i-1},y_i} + \sum_{i=1}^{n} emission_{y_i}(x_i) \tag{3}$$

(4) Update feature matrices: To update the feature matrices, according to training label, if the label is incorrect, we add a weight into desired transition or emission. At the meantime, we extract a weight from the transition or emission we predict.

$$transition_{y_{i-1},y_i} \begin{cases} += 1 & \text{if prediction is wrong and } y_{i-1} \text{ to } y_i \text{ is desired transition.} \\ -= 1 & \text{if prediction is wrong and } y_{i-1} \text{ to } y_i \text{ is prediction.} \end{cases}$$

$$emission_{y_i}(x_i) \begin{cases} += 1 & \text{if prediction is wrong and } x_i \text{ is desired emission.} \\ -= 1 & \text{if prediction is wrong and } x_i \text{ is prediction.} \end{cases}$$

(5) Iterate between (3) and (4) for T times.

Bias Reduction The trick we use here is to shuffle the training data at sentence level and train a different set of parameters for each shuffle. And then we average the parameters to reduce the bias resulting from the order of the sentences.

(1) For each shuffle, we realize that the parameter sets are the accumulation of T sets from T iterations. So the final parameter sets are as followed:

$$transition_{y_{i-1},y_i} = \frac{1}{T}\sum_{t=1}^{T} transition^{(t)}_{y_{i-1},y_i} \tag{4}$$

$$emission_{y_i}(x_i) = \frac{1}{T}\sum_{t=1}^{T} emission_{y_i}(x_i)^{(t)} \tag{5}$$

(2) As we mentioned above, the above parameters are strongly dependent on the order of the sentences, randomly shuffling the sentences results in different parameter sets. Based on this observation, we implement the following equation:

$$transition_{y_{i-1},y_i} = \frac{\sum_{j=1}^{n} transition^{(j)}_{y_{i-1},y_i}}{n} \tag{6}$$

$$emission_{y_i}(x_i) = \frac{\sum_{j=1}^{n} emission_{y_i}(x_i)^{(j)}}{n} \tag{7}$$

(3) Furthermore, we realize that due to the limitation of the training set, when the feature parameter equals 0, it does not necessarily imply this feature does not have chances to be tested in the testing set. So we further modify the equation and only average the non-zero parameters:

$$transition_{y_{i-1},y_i} = \frac{\sum_{j=1}^{n} transition^{(j)}_{y_{i-1},y_i}}{|(j|transition^{(j)}_{y_{i-1},y_i} \neq 0, j = 1, ..., n)|} \tag{8}$$

$$emission_{y_i}(x_i) = \frac{\sum_{j=1}^{n} emission_{y_i}(x_i)^{(j)}}{|(j|emission_{y_i}(x_i)^{(j)} \neq 0, j = 1, ..., n)|} \tag{9}$$

4.3 Experiments

Since the training data are shuffled randomly, the results vary from time to time. Here we give the best result for both English and French.

Table 5. Best Test Results for English and French.

Dataset	F1(Correct Entity)	F1(Correct Label)	Iterations	Shuffles	Threshold k
EN	0.7074	0.6379	10	10	1
FR	0.6707	0.4268	10	5	2

For learning purpose, we also compute the result for CN and SG. From the result, we realize that structured perceptron is not suitable for all languages. For example, it does not increase the performance of CN greatly compared with second order HMM. This may result from the different semantic features between Chinese and English.

Table 6. Best Test Results for CN and SG.

Dataset	F1(Correct Entity)	F1(Correct Label)	Iterations	Shuffles	Threshold k
CN	0.4042	0.3066	10	5	1
SG	0.5271	0.2952	5	1	1

4.4 Deployment

The script must be put under the same directory as the training data, as well as evalResultHelper.py. Please see figure 1 as an example.

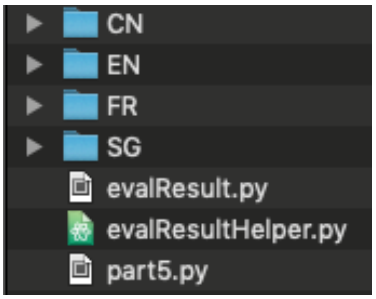

Fig. 1. Directory structure.

Please execute the following command to run the script for English and French respectively.

```
python part5.py -d EN -i 10 -k 1 -s 10

python part5.py -d FR -i 10 -k 2 -s 5
```

```
-d: string, file path to the dataset
-i: int, default=10, number of iterations for each shuffle
-k: int, default=1, threshold for word pre-processing
-s: int, default=5, number of shuffles
```

References

1. Zhang, K., Su, J., Zhou, C. (2014). Regularized Structured Perceptron: A Case Study on Chinese Word Segmentation, POS Tagging and Parsing. Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics. https://doi.org/10.3115/v1/e14-1018