50.007 Machine Learning Project Report

Team Member	Student ID
Qiao Yingjie	1004514
Zhang Peiyuan	1004539
Huang He	1004561

Introduction

In this project, we present a pure python based implementation of the hidden markov model (HMM) and the second-order hidden markov model for the task of simple sentiment analysis.

Dependencies:

- **Python 3.6+ Only** (no external libraries required)

Please refer to the README.md for instructions on how to run the project. All outputs are saved in the `outputs` folder. Implementation details and results are reported below.

Part 1

Relevant code section:

- Function that computes the emission probability: hmm.py Line 36-67
- Introducing $k = 1$ for calculating the emission probability for <UNK>: hmm.py Line 60-65
- Simple sentiment analysis based on emission probability: hmm.py Line 298-321

Part 2 & 3

Relevant code section:

- Function that computes the transition probability: hmm.py Line 69-108
- We use log-likelihood to compute the probability of emission and transition to avoid underflow: hmm.py Line 100
- We also introduce a smoothing factor which is a small constant that will be used to estimate close-to-zero probability to avoid undefined log-likelihood probabilities. hmm.py Line 100

```
prob_func = lambda x: math.log(x) if x > 0 else math.log(smoothing_factor)
```

- Top-k Viterbi algorithm: hmm.py Line 186-260

Part 4: RNN & 2nd Order HMM

RNN

We first attempted using a pure python implementation of Recurrent Neural Network (RNN) to perform this sequence labeling task. However, the result was poor. Relevant code can be found in RNN folder, but we will not use that for our final submission.

Second Order Hidden Markov Model

We then opt-for the second order HMM model. Relevant code: second_order_hmm.py

Transition matrix

$$P(y_i \mid y_{i-2}, y_{i-1}) = \frac{\texttt{count}(y_i, y_{i-2}, y_{i-1})}{\texttt{count}(y_{i-2}, y_{i-1})}$$

Viterbi Algorithm

We also need to rewrite Viterbi algorithm.

We use $S_t(j,k)$ to denote the set of possible decoding sequences ending with $y_{t-1} = j$ and $y_t = k$.

We use $M_t(j,k)$ to denote $\max\limits_{i \in S_t(j,k)} \{P(i|X)\}$

Then M can be calculated using Dynamic Programming:

$$M_t(j,k) = \max_i \{M_{t-1}(i,j) \times P(y_k \mid y_i, y_j) \times P(x = X_t \mid y = k)\}$$

We then perform backtracking to find out the decoding with maximum likelihood.

Label Smoothing

Additionally, we add Laplacian smoothing as label smoothing in the model when calculating model parameters, we set $\lambda = 1$ for calculating transition probability and $\lambda = 0.3$ for calculating emission probability.

$$P_\lambda(X^{(j)} = a_{jl} \mid Y = c_k) = \frac{\sum_{i=1}^{N} I(x_i^{(j)} = a_{jl}, y_i = c_k) + \lambda}{\sum_{i=1}^{N} I(y_i = c_k) + S_j \lambda}$$

Evaluation Results on ES & RU Dev Set

Please refer to our README for instructions on how to run the code.

Please refer to the outputs folder for the output files. We summarize the results as follows:

		Entity				Sentiment			
		#Correct	Precision	Recall	F Score	#Correct	Precision	Recall	F Score
ES	dev.p1.out	205	0.1183	0.8039	0.2062	113	0.0652	0.4431	0.1137
	dev.p2.out	130	0.6468	0.5098	0.5702	105	0.5224	0.4118	0.4605
	dev.p3.out	116	0.2437	0.4549	0.3174	93	0.1954	0.3647	0.2544
	dev.p4.out	110	0.6077	0.4314	0.5046	89	0.4917	0.3490	0.4083
RU	dev.p1.out	335	0.1604	0.7267	0.2627	136	0.0651	0.2950	0.1067
	dev.p2.out	226	0.6706	0.4902	0.5664	156	0.4629	0.3384	0.3910
	dev.p3.out	198	0.2532	0.4295	0.3186	132	0.1688	0.2863	0.2124
	dev.p4.out	175	0.4605	0.3796	0.4162	130	0.3421	0.2820	0.3092

Acknowledgement

We refer to the following resources when implementing this project:

- Sung-Hyun, Yang, et al. “Log-Viterbi Algorithm Applied on Second-Order Hidden Markov Model for Human Activity Recognition.” International Journal of Distributed Sensor Networks, Apr. 2018, doi:10.1177/1550147718772541.
- RNN: pangolulu / rnn-from-scratch