CSE 5525 HW2

HMM and CRF for Named Entity Recognition Report

Kun Peng

Discussed with Shibo Zhang

Introduction

This project aims to build up a Named Entity Recognition (NER) system with Hidden Markov Model (HMM) and Conditional random fields (CRFs). The NER system aims to predict the NER chunks, which are certain types of named entities of unseen sentences. The project uses data presented in the CoNLL 2003 Shared Task (Tjong Kim Sang and De Meulder, 2003) for training and testing. For the HMM model, the project uses Viterbi algorithm for decoding, and the CRF model also uses forward-backward algorithm based on Viterbi algorithm.

Part 1. Viterbi Decoding

This algorithm recursively computes the probability of the most likely subsequence of states that accounts for the first observation to the t observation, ending in state s_j. It also records the back-pointers for back tracing the most probable state sequence. Time complexity of the Viterbi algorithm is $O(TN^2)$. Start state and Stop state are omitted in this project, and all variables are stored in logarithm. I get a **76.89 F1** on the development set.

Part 2. CRF Training

This model introduces forward-backward algorithm based on the Viterbi algorithm from part 1. Since transition features are slow to learn in this model, I set up constraints on the models to prohibit a transition to tag I-X from anything except I-X and B-X, i.e. tag O should not follow I-X, and I-A/B-A should not follow I-B. To realize this constraint, I set the score of the invalid state to minus infinity so it can never be the max value among all probable states. I use unregularized Adagrad as the optimizer and set the batch size to 1. To speed up the training process, I use the feature cache and use Counter class to compute gradients. Additionally, I shuffle the training set to increase accuracy. Although shuffling the training set may influence the final weights, the differences on accuracy are not so large. In this model, all variables are also stored in logarithm.

During the testing, I find that the learning speed of this model is slow – only one epoch takes

500+ seconds and gets **84.63 F1**. According to the profile, the most time-consuming step is optimizer.score(). After the second epoch, my model has already reached **86.51 F1** which is higher than 85. For comparison, I train the model for up to 10 epochs and save the updated weights after each epoch with numpy.save() method. According to the outputs, the final weights with 10 epochs gets the highest F1 which is **87.70**, so I chose this set of weights to do the blind test. All F1s of different epochs are listed in the following figure.

model	epoch	F1	epoch	F1
HMM	N/A	76.89		
CRF	1	84.63	6	87.43
	2	86.51	7	87.52
	3	86.80	8	87.43
	4	87.15	9	87.38
	5	87.16	10	**87.70**

For testing purpose, I comment the code of the training part and use numpy.load() to load my pre-trained weights.

Part 3. Unsolved Problems

There are several problems which can be improved in future:

- Although it is not required, there is no constraint on the HMM model, so in the output there are still warnings that I tag comes after O tag.
- Although the CRF model reaches the requirement of 85 F1 with 2 epochs, it still takes too long. The code may be improved to speed up the learning.
- In the CRF training, there is no constraint on the initial states to prohibit I tags as the first state, though it does not matter for these data.

References

[1] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. arXiv preprint cs/0306050, 2003.