HOMEWORK FOUR

Chang Zhang

University of California San Diego, La Jolla, CA 92093, USA

ABSTRACT

In this paper, we evaluate the performance of the trigram hidden Markov model (HMM) on a sequence tagging task by comparing its performance with the performance of the baseline model. In addition to having a standard trigram HMM, we use informative word classes to help deal with rare or unseen words. We discovered that, by using a more informative word class, we can improve the model accuracy. Also, we found that trigram HMM greatly improves the model accuracy.

1. DATA

In this paper, we use a training dataset and a development dataset. In the training dataset, we have 137954 sentences with each word or punctuation tagged with either O or I-gene. We use the development dataset, which contains 50928 sentences, to predict the tag for each given word and examine the accuracy of tag prediction.

2. BASELINE

The baseline model predicts tags by selecting the tag that maximizing the emission probability.

$$y*_i = argmax_{yi} e(xi|yi) = argmax_{yi} \frac{Count(yi \rightsquigarrow xi)}{Count(yi)}$$

One problem of such model is that the emission probability would be zero when the model encounters a word that is not seen in the training set. One method of solving this issue is to group a set of words that are rare (the count of the word appearance is less than 5) and replace them with a common symbol _RARE_. When encountering a word that is unseen from the training set, we can calculate the emission probability by treating the word as _RARE_. To further improve its performance, we added additional word classes. For words that have less then 5 occurrences, we categorize them into either number, non-alphanumerical, or neither, which we assign them tags: _NUMB_, _SYN_, and _RARE_, respectively. To be categorized as a number, every character in the word has to be a number. To be categorized as non-alphanumerical, every character in the word has to be neither a number or alphabetical.

2.1. Design Intuition

: We picked number and non-alphanumerical as two addition word classes as it is relatively more computationally efficient to categorize words/punctuation into these classes than other methods that can extract more detailed information of a given word.This means the model is still able to make predictions efficiently. Another reason is that numbers and punctuation are more frequently tagged as O. Knowing it is either it is a punctuation or number can potentially help the model make better predictions.

2.2. Performance

The performance of the baseline model with different word classes is shown in the table below. We see that adding number as an additional word class improves the performance by a small amount. However, adding non-alphanumerical decreases the model performance. Overall, the baseline model is prone to over classify (high false positive rate), which leads to a low F1 score.

Model	Word classes	Number of I-GENE found	Expected	Correct	Precision	Recall	F1-Socre
Baseline	rare	2669	642	424	0.1589	0.6604	0.2561
Baseline	rare & number	2644	642	424	0.1603	0.6604	0.258
baseline	rare & non-alphanumeric	2675	642	424	0.1585	0.6604	0.2557
Baseline	rare & number & non-alphanumeric	2649	642	424	0.1601	0.6604	0.2577

Fig. 1: The performance of the baseline model on development dataset.

3. TRIGRAM HMM

The goal of the Trigram HMM is to find a sequence of tags that maximizes the joint probability of a sentence.

$$y *_1 ...y*_n =$$

$$argmax_{y_1..y_n} q(y_1|*,*) \times q(y_2|*,y_1) \times q(STOP|y_{n-1},y_n)$$
$$\times \prod_{i=3}^{n} q(y_i|y_{i-2},y_{i-1}) \times \prod_{i=1}^{n} e(x_i|y_i)$$

$$q(y_i|y_{i-2}, y_{i-1}) = \frac{Count(y_{i-2}, y_{i-1}, y_i)}{Count(y_{i-2}, y_{i-1})}$$

$q(y_i|y_{i-2}, y_{i-1})$ is the trigram transition probability.

3.1. Viterbi algorithm

We used Viterbi algorithm to find the max probability tag sequence, which is a dynamic programming algorithm. This method is significantly better than the brute force method or the greedy algorithm. The greedy algorithm is computationally efficient but it will not find the globally optimal solution. Brute force method will find the most optimal value, however, it has a time complexity of 3 to the power of the number of words in a given sentence. It is extremely inefficient which makes it impossible to compute for a large given dataset. Viterbi algorithm is able to reduce time complexity down to the number of word times the tag set the size to the power of three.

The Viterbi algorithm has three parts: the base case, a recursion, and a mechanism to retrieve the tag sequence. The base case is the probability of the start-of-sentence symbol are the tags prior to the start of a sentence, which is 1. We denote it as: $\pi(0, *, *) = 1$. In the recursion, we go thru each tag at k-2 and k-1 to determine the most optimal tag at k-2 position by using the emission probability, the transition probability and π_{k-1} of the particular combination of previous two tags. The most optimal k-2 tag is then stored in a set of back pointers. Then we use the set of back pointers to retrieve the most optimal tags. We first have to determine the last two optimal tags by finding the combination of the two tags that produces the highest probability. With the last two tags initialized, we can start the backtracking using the back pointer set.

Essentially, we are computing the joint probability tag by tag, each probability of a tag depends on the probability of its previous tag and the word it produces. The argmax of this joint probability is the desired sequence of tags.

3.2. Trigram HMM Performance

The performance of the baseline model with different word classes is shown in the table below.The accuracy scores have been improved hugely from the baseline model. We also found that this time, accuracy only slightly improved after adding non-alphanumerical word class. Unlike for the baseline model, the addition of the number class reduces the accuracy.

It is also worth noting that the trigram HMM underestimates the I-GENE tags. With 642 correct values, trigram HMM only found about 370, in which only about 200 of them were correct. This behavoir is exactly the opposite of the baseline model.

Model	Word classes	Number of I-GENE found	Expected	Correct	Precision	Recall	F1-Socre
Trigram HMM	rare	376	642	202	0.5372	0.3146	0.3969
Trigram HMM	rare & number	377	642	201	0.53331	0.3131	0.3945
Trigram HMM	rare & non-alphanumeric	371	642	202	0.5441	0.3146	0.3988
Trigram HMM	rare & number & non-alphanumeric	373	642	202	0.5415	0.3146	0.398

Fig. 2: The performance of the trigram HMM model on development dataset.

4. CONCLUSION/DISCUSSION

In our experiment, we have shown that Trigram HMM is a superior method over the baseline model when tackling the task of sequence tagging, as well as the baseline model overestimates the number of I-GENE tags while Trigram HMM would under-estimate. We also attempted to discover the impact of adding to informative word classes instead of only having rare/unseen and regular words. We notice that, in general, adding more informative word classes can improve accuracy. However, since the number of I-GENE tags in the development dataset is small, the impact of one correctly or incorrectly classified tag is large. And because changing the informative word classes only changes the number of missclassified or correctly classified by less than a handful in the case of Trigram HMM, we cannot confidently say that we will be able to see the trend we discovered on other datasets.