ヒューマンインタフェース 61－1
（1995. 7. 20）

Prediction of Part of Speech Categories for Words in English Text Using Second Order Markov Model

Aksanul FARDHI [†] Kenji ARAKI [††] Yoshikazu MIYANAGA [†] Koji TOCHINAI [†]

[†] Faculty of Engineering, Hokkaido University
[††] Faculty of Engineering, Hokkai-Gakuen University

Abstract

In natural language processing, one of the problems for sentence parsing is to assign the correct part of speech category to each word. This kind of task is also known as tagging. Nowadays, tagging system mainly uses first order markov model as its mathematical based. In this paper we present a tagging system which is based on second order markov model. First, we trained the system with a training data which was derived from a manually tagged LOB corpus. After that, we tested the system on our 116,785 words test data which was also derived from the LOB corpus. We carried several tests and the results were very satisfactory.

英文を対象とした２次マルコフモデルを用いた単語品詞予測

ファルディ・アクサヌル [†] 荒木健治 [††] 宮永喜一 [†] 栃内香次 [†]

[†] Faculty of Engineering, Hokkaido University
[††] Faculty of Engineering, Hokkai-Gakuen University

あらまし

自然言語処理では、構文解析の一つの問題は各単語に正しい品詞をあてはめることである．このような処理をタギングという．従来のほとんどのタギングシステムは１次マルコフモデルを用いている。これに対して、本稿では２次マルコフモデルを用いたタギング手法を提案する．さらに、本手法を品詞付きコーパスを用いて学習させた上で、他のコーパスを用いてその有効性を評価し、良好な結果が得られた。この結果についても報告する。

1 Introduction

The task of predicting part of speech category is also known as tagging. English text tagging is widely researched due to the need of text with disambiguated words, while English vocabulary contains many ambiguous words. An ambiguous word is a word which has more than one part of speech categories. This kind of word appears in the sentence with different part of speech category, depends on the context of the sentence. De Rose[2] has investigated this ambiguity of English vocabulary that occured in the Brown corpus, the one million words corpus widely used on English text research. He found out that 11.5% of words on the corpus' vocabulary and 40% of running words are categorically ambiguous. Meanwhile, the part of speech category of a word reflects its syntactic and semantic role in a sentence. A correct interpretation of an English word is impossible without disambiguating the word – that is to assign the correct part of speech category to the word.

There are two main approach commonly used in a tagging system: rule-based approach and probabilistic approach. The system constructed by Klein et al.[5] is one of the systems which are considered to be rule-based. They used 30 categories on their Klein and Simmons algorithm and claimed an accuracy of 90% in tagging the words. Greene et al.[4] were the first to attempt tagging a large sample of data. Using a palette of 86 part of speech categories, they succeeded in tagging correctly 77% of the million words in the Brown corpus.

In probabilistic approach, a large corpus is required to construct the dictionary containing word, part of speech category combinations and their probabilities. Charniak et al.[1] divided the Brown corpus into two: training corpus and test corpus. The training corpus was used to construct the system's dictionary. The system was then tested on the test corpus. With 181 distinct part of speech categories they claimed an accuracy of 96.45% on their system's best performance. Some other example of achievement through probabilistic approach are achieved by Kupiec[6] and Merialdo[7]. Kupiec introduced word equivalence classes as substitutes for words. By doing this, he succeeded in reducing the size of the dictionary. Merialdo implemented a Maximum Likelihood Training to train the system with untagged corpus.

In this paper, we present a tagging system which is based on second order markov model. We created a similar circumstance to that of Charniak's system. We tested the system to it and the results were satisfactory. However, it took a lot of time. That was because the number of possible combinations on second order markov model was larger than that of first order one. To solve the problem, we introduced the keyword concept. By implementing this concept, we succeeded in shortening the processing time without decreasing the accuracy. On its best performance, our system yielded an accuracy of 97.09% on a 116,785 words test data with 149 part of speech categories. The data was derived from the Lancaster-Oslo Bergen (LOB) corpus which was already manually tagged.

2 Mathematical Model

First, let us assume that the English language that we know has some fixed vocabulary. We decide to make the mathematical expression for this vocabulary as $W = \{w^1, w^2, \ldots, w^x\}$ where $w^1, w^2, \ldots, w^x$ are the member words of the vocabulary, and x is the number of the words. For example, W can be like $W = \{a, aback, \ldots, zymotic\}$. Secondly, we assume that the English language also has some fixed part of speech categories. Like the vocabulary above, we make a mathematical expression for these categories, that is $T = \{t^1, t^2, \ldots, t^y\}$ e.g. $\{adjective, adverb, \ldots, verb\}$ where $t^1, t^2, \ldots, t^y$ are the part of speech categories and y is the number of the categories.

After that, we make assumption on sentences occured on English language. We assume that sentence is a words' sequence which can be expressed as $w_1, w_2, \ldots, w_n$ where n is the number of words on the sentence. Any word can become one of $w_1, w_2, \ldots, w_n$ as long as it is a member of vocabulary W. In the same way, we assume that part of speech categories of words in a sentence can also be expressed as $t_1, t_2, \ldots, t_n$, and any part of speech categories can become one of $t_1, t_2, \ldots, t_n$ as long as it is a member of part of speech category set T. With all these assumptions, we define the tagging problem as below.

$$\mathcal{S}(w_{1,n}) = \underset{t_{1,n}}{\text{arg max}}\, P(t_{1,n} \mid w_{1,n}) \tag{1}$$

It means that the most suitable part of speech categories sequence $t_{1,n}$ for sentence $w_{1,n}$ is that with the maximum probability value. We expand equation (1) into equation (2).

$$\begin{aligned} \mathcal{S}(w_{1,n}) &= \underset{t_{1,n}}{\text{arg max}}\, P(t_{1,n}|w_{1,n}) \\ &= \underset{t_{1,n}}{\text{arg max}}\, \frac{P(t_{1,n},w_{1,n})}{P(w_{1,n})} \end{aligned} \tag{2}$$

Because the denominator on equation (2) does not have anything to do with any combination of $t_{1,n}$, the maximum value of equation (2) simply depends on its numerator. That make the equation become as simple as equation (3).

$$\mathcal{S}(w_{1,n}) = \underset{t_{1,n}}{\text{arg max}}\, P(t_{1,n}, w_{1,n}) \tag{3}$$

At this point, we could simply estimate $P(t_{1,n}, w_{1,n})$ like equation (4) below, where N is the number of sentences on the training data.

$$P(t_{1,n}, w_{1,n}) \approx \frac{C(t_{1,n}, w_{1,n})}{N} \tag{4}$$

However that will not be practical. First, since the training process will be carried on the sentence level, it will not create a large number of statistic data. Second, there will not be any universality since each sentence on the training data will vary in number of words. Because of that, we break $P(t_{1,n}, w_{1,n})$ on equation (3).

$$P(t_{1,n}, w_{1,n}) = P(t_1 \mid w_1) \prod_{i=2}^{n} P(t_i \mid w_{i-1}, w_i) \tag{5}$$

Next, we substitute $P(t_{1,n}, w_{1,n})$ on equation (3) with what we have got here on equation (5).

$$\mathcal{S}(w_{1,n}) = \underset{t_{1,n}}{\text{arg max}} \left\{ P(t_1 \mid w_1) \prod_{i=2}^{n} P(t_i \mid w_{i-1}, w_i) \right\} \tag{6}$$

Finally, in order to make the process run on word level we modify function $\mathcal{S}(w_{1,n})$ into $\mathcal{S}'(w_i)$.

$$\mathcal{S}'(w_i) = \begin{cases} \underset{t_i \in T}{\text{arg max}}\, P(t_i \mid w_i), & \text{for } i = 1 \\ \underset{t_i \in T}{\text{arg max}}\, P(t_i \mid w_{i-1}, w_i), & \text{for } 2 \le i \le n \end{cases} \tag{7}$$

Equation (7) means that:

- Only the first word of the sentence is solved with first order markov model.
- The second word and every word to the right of the second word is solved with second order markov model.

However, practically that is not true. The second order markov model data that we build is based on training data. Because of that, only words combination which appeared at least once somewhere on the training data has a second order markov model probability. Words combination which never appeared on the training data will be treated as an unknown combination and will have to be solved with first order markov model. So we decided to modify equation (7) into equation (8). Here, α was the weight given to the first term of the equation (8).

$$\begin{aligned} \mathcal{S}'(w_i) = \underset{t_i \in T}{\text{arg max}}\; &\{\alpha P(t_i|w_{i-1}, w_i) + \\ &(1-\alpha) P(t_i|w_i)\} \end{aligned} \tag{8}$$

3 Training the System

It was one of our goals from the very beginning to make it an open system. As the consequence, two different data were required. To fulfill the requirement we derive two different data from the LOB corpus. The LOB corpus itself contains 1,157,260 words in 54,490 sentences. The question is, how many words are required for the training data and how many are for the test data. Fortunately, there is a good example for this. Charniak et al.[1] used 114,203 words for their test data. The size of the training data is never been revealed but it is said to be nine times of the size of the test data. Furthermore, Charniak et al. used first order markov model which was very similar to second order markov model we use. This similarity will make the evaluation and the comparison between the two system easier. For the reason above, we decide to make our test data's size in the neighbor of one hundred thousand words and to leave the rest as the training data.

Another important requirement for an open system is the heterogeneousness of both the training data and the test data. To clear this requirement the test data has to be sampled at a fix interval. In this paper, we sample every ten sentences starting from the second sentence of the LOB corpus. The sampled sentences become the test data and the rest become the training data. The outcomes are a training data with 1,040,475 running words in 49,041 sentences and a test data with 116,785 running words in 5,449 sentences.

After getting a training data in a sufficient size, we train the system with the training data. The purpose of the training is to obtain frequency data so that we could approximate the probability function like the equations below.

$$P(t_i = t^j \mid w_i = w^k) \approx \frac{C(t_i = t^j, w_i = w^k)}{C(w_i = w^k)} \quad (9)$$

$$P(t_i = t^j \mid w_{i-1} = w^k, w_i = w^l) \approx \frac{C(t_i=t^j, w_{i-1}=w^k, w_i=w^l)}{C(w_{i-1}=w^k, w_i=w^l)} \quad (10)$$

Where,

- i is an arbitrary natural number on equation (9).
- i is an arbitrary natural number larger than 1 on equation (10).
- $t^j \in T = \{t^1, t^2, \ldots, t^y\}$
- $w^k, w^l \in W = \{w^1, w^2, \ldots, w^x\}$.

Equation (9) means that the probability of part of speech category t^j become the tag of the word w^k is approximated by the appearence frequency of word w^k with t^j as its tag, divided by the total appearence frequency of the word w^k.

Almost the same meaning goes to equation (10). The probability of part of speech category t^j become the tag of word w^l when its previous word is w^k, is approximated by the appearence frequency of word w^l with t^j as its tag and w^k as its previous word, divided by the total appearence frequency of the word w^l with w^k as its previous word.

Tag(s)	*Word(s)*	*Frequency*
1	41,811	430,299
2	4,114	252,450
3	822	86,201
4	87	125,216
5	27	64,066
6	9	15,884
7	2	21,418
8	3	45,862
10	1	6,645

Table 1. The Distribution of $P(t_i \mid w_i)$

From the training process using equation (9) we obtain 53,090 word-tag combinations. Table 1 shows the distribution. From 53,090 combinations, 41,811 of them (around 78.76 %) are disambiguous. If we see it from the appearence frequency's point of view, of 1,040,475 total word appearences only 430,299 appearences are disambiguous. That is only about 41.36%. The rest are ambiguous.

Tag(s)	*Combination(s)*	*Frequency*
1	371,575	877,846
2	5,762	81,951
3	340	24,501
4	44	5,451
5	9	3,493
6	1	69

Table 2. The Distribution of $P(t_i \mid w_{i-1}, w_i)$

From the training process using equation (10), we obtain 384,346 combinations. Table 2 shows the distribution. From 384,346 total combinations, 371,575 of them (around 96.68%) are disambiguous. It is better compare to the disambiguity of first order markov model data which is only 78.76%. Table 2 also shows that the appearence frequencies of the disambiguous combinations are high. From 991,434 combinations detected on the training data, 877,846 of them (88.54%) are disambiguous. On first order markov model, it was only 41.36%. Last but not least, on second order markov model you only have to deal only with six probable tags even on the worst case of ambiguity. With first order markov model, on the worst case of ambiguity you have to deal with up to ten probable tags. This is another good things about second order markov model. From the above data, we can naturely expect a higher accuracy of tagging result compare to that of first order markov model.

4 Testing the System

4.1 Without Keyword

Like we have already mentioned above, the test data, which we are going to test the system to, is different from the training data. In this way, the system is kept as an open system.

To test the system, the weight α on equation (8) needs to be fixed. Because we did not have any clues how much the optimum value for the weight α was, for the first test we fixed it on 0.8. The result was, 113,669 words of 116,785 words on the test data were correctly tagged. That is a 97.33% accuracy.

However, we did not take a high accuracy of tagging result as a sign that the weight α was already optimum. Like we have already explained on section 3 above the data itself, which was obtained using second order markov model, was very disambiguous. This factor could be the one that gave us such a high tagging accuracy. We also thought that an optimized value of weight α could give us a high accuracy. Because of that on the second test we tried in our own way to optimize the weight α.

The optimization was done at the intermission after one sentence was finished being tagged and before the tagging process for the next sentence was started. The equation for the optimization was as follows. Here f is the total number of words until the intermission, which has a non-zero value of $P(t_i \mid w_{i-1}, w_i)$. s is the total number of running words until the intermission, which was processed. We fixed the initial value of weight α on 0.8.

$$\alpha = \frac{f}{s} \tag{11}$$

The example below will help you to understand our optimization method. Let us assume that the sentences below are the first three sentences on the test data.

1. Bill Clinton is the president of the United States.
2. He is from Arkansas.
3. He is also from the Democrat Party.

From three sentences above we get 16 combinations, those are **Bill_Clinton, Clinton_is, is_the, ..., Democrat_Party**. Let us assume that from 16 combinations, 10 of them:

- **Bill_Clinton**
- **is_the**
- **the_president**
- **of_the**
- **the_United**
- **United_States**
- **He_is**
- **from_the**
- **the_Democrat**
- **Democrat_Party**

appeared at least once on the training data. Thus, these combinations' $P(t_i \mid w_{i-1}, w_i)$ not equal to zero. We call these combinations known combinations. The rest combinations have zero $P(t_i \mid w_{i-1}, w_i)$ because they never appeared on the training data. We call these combinations unknown combinations.

After understanding the unknown combination consept above, please note the movement of weight α from one intermission to another intermission.

1. Because we fixed the initial value of weight α on 0.8, during the first sentence tagging process weight α is 0.8
2. At the intermission between the first and the second sentence, because on the first sentence there are six known combinations ($f = 6$) and the total running words of the first sentence are nine ($s = 9$) the weight α is optimized to:
$$\alpha = \frac{f}{s} = \frac{6}{9} = 0.667$$
3. The second sentence is tagged with $\alpha = 0.667$
4. At the intermission between the second and the third sentence, because on the first sentence and on the second sentence there are seven known combinations ($f = 7$) and the total running words of the first and the second sentence are thirteen ($s = 13$) the weight α is then optimized to:
$$\alpha = \frac{f}{s} = \frac{7}{13} = 0.539$$
5. The third sentence is tagged with $\alpha = 0.539$
6. At the intermission between the third and the fourth sentence, because from the first sentence to the second sentence total there are eleven known combinations ($f = 11$) and the total running words from the first to the third sentence are twenty ($s = 20$) the weight α is then optimized to:
$$\alpha = \frac{f}{s} = \frac{11}{20} = 0.55$$
7. And so on.

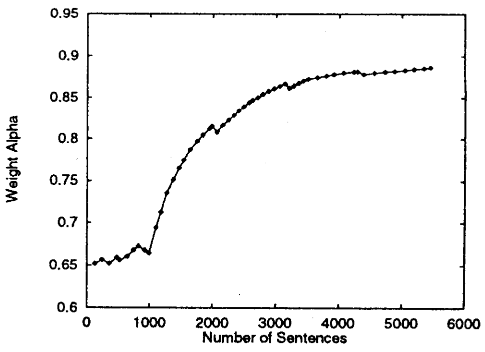

Fig.1 Weight α's Behaviour During Experiment

With weight α optimized every sentence, we carried the second experiment. The result was, 113,657 words of 116,785 words on the test data were correctly tagged. During the experiment, we sampled weight α's value about every hundred sentence. Fig.1 shows its behaviour. At the end of experiment, it converged at 0.885841 $(103,453/116,785)$.

This second experiment did not give us result as good as we expected it would be. But from this experiment we knew that of 116,785 total running words on the test data, 103,453 words among them were solvable with second order markov model. Hence, $103,453/116,785 = 0.885841$ is the optimum value for weight α. On the third experiment we fixed weight α on 0.885841 and we succeeded in tagging correctly 113,668 words of 116,785 total words.

4.2 With Keyword

So far, we have a satisfactory achievement in the field of tagging accuracy. Three of our previous experiments gave us result higher than 96.45% which was achieved by Charniak et al.[1].

However we considered that there was something we could do with the processing time. So far our fastest model was the first model which gave a result with 97.33% accuracy. The model took 40 hours 34 minutes and two seconds of time.

α	*WITHOUT KEYWORD*		*WITH KEYWORD*	
	RESULT	*TIME*	*RESULT*	*TIME*
0.8	97.33%	$40 : 34'02''$	97.33%	$39 : 48'19''$
Opt.	97.32%	$42 : 21'35''$	97.33%	$34 : 34'24''$

Table 3. Summary of Results

In a bid to cut the processing time we used the disambiguity which exists in the training data. As we have already explained, 41,811 words (78.76%) of distinct words which were detected on the training data are disambiguous (see Table 1.) We used these ambiguous words as keyword. As the result, we succeeded in cutting the processing time. With keyword, our slowest model took 39 hours 48 minutes and 19 seconds to do the tagging process. Table 3 shows the details.

5 Conclusions

In this paper we presented our method to tag English text using second order markov model. We also proved that second order markov model gave better result than first order markov model. Our experiments also show that the system took relatively long time. That was because the combinations detected were also relatively many. To counter the problem, we implemented the keyword and we succeeded in cutting the processing time without decreasing the tagging accuracy.

However, only a high accuracy of tagging result does not guarantee a good result of sentence parsing. A lot of hard work is still required to make tagging useful in the sentence parsing. The system we presented in this paper only refers to two in a row words combinations. We think it is worth to try other kind of words combinations such as one word skip combination or two words skip combination.

We also think there is a need to reconsider the number of part of speech categories used. In this paper we used 149 distinct part of speech categories. There is still a possibility to reduce it.

References

[1] *Charniak, E., Hendrickson, C., Jacobson, N. and Perkowitz, M.* Equations for Part of Speech Tagging, Proc. of the 11th National Conference on Artificial Intelligence, pp.784-789, 1993

[2] *De Rose, Steven J.* Grammatical Category Disambiguation by Statistical Optimization, Computational Linguistics, Vol.14, No.1, pp.31-39, 1988

[3] *Fardhi, A., Araki, K., Miyanaga, Y. and Tochinai, K.* Next Word's Probability Based

English Text Part of Speech Tagging, Proc. of the 1995 IEICE General Conference, Information System Part I, p.284, 1995

[4] *Greene, Barbara B. and Rubin, Gerald M.* Automated Grammatical Tagging of English, Department of Linguistics, Brown University, Providence, Rhode Island, 1971

[5] *Klein, S. and Simmons, R.F.* A Grammatical Approach to Grammatical Coding of English Words, JACM, 10, pp.334-347, 1963

[6] *Kupiec, Julian* Robust Part of Speech Tagging Using a Hidden Markov Model, Computer Speech and Language, No.6, pp.225-242, 1992

[7] *Merialdo, Bernard* Tagging English Text With a Probabilistic Model, Computational Linguistics, Vol.20, No.2, pp.155-171, 1994