Low Resource NLP Assignment

Yuan-Chia Cheng

April 2019

1 The Design of a WSD System

In the task, we design a pipeline, that is also a system, to perform **word sense disambiguation** task.

1.1 Deep into the Provided Annotated Training Data

The very first step is to get a clear whole picture of the content and structure of the provided annotated training data. Afterwards, here come 3 strategies to take to manipulate the data depending on the scale and distribution of data:

1. Pure online data search
2. Pure parsed data search
3. Hybrid search

If the size scale is practically numerous and the distribution is rather balanced, then we may as well take the first strategy **Pure online data search** where we go on our counting task online each time required by the task shown later.
If the size scale is practically affordable, then it's worth a try to take second strategy **Pure parsed data search** where we parse the online data into our desire format and save them to our database in turn. Otherwise, we're supposed to take the third strategy **Hybrid search**, which literally combines the first and the second ones. In the case of frequent word, we take the second strategy, and in the other, we take the first one.

1.2 Model: Mainly Apply Naive Bayes Model

We basically adapt Naive Bayes Model to determine the word sense in the context.
Here shows the core formula of Naive Bayes Model:

$$s^* = \arg\max_{s_k} \prod_{v_x \in C'} P(v_x|s_k)P(s_k) \tag{1}$$

where s_k = the candidates of the sense, $C^{'}$ is the context of one testing sample.
In order to evaluate the probability, we perform counting task:

$$P(v_x|s_k) = \frac{Count(v_x, s_k)}{Count(s_k)} \tag{2}$$

$$P(s_k) = \frac{Count(s_k)}{Count(w)} \tag{3}$$

Last but not least, some trick is applied to smooth the probabilities so as to avoid suffering from the zero probability problem especially. Here we adapt Add-one smoothing method.
So the final formula simply reduces to

$$s* = \arg\max_{s_k} \prod_{v_x \in C'} (Count(v_x, s_k) + 1) \tag{4}$$

As shown, the only actual procedure we need to go through turns out to be pure counting task.

1.3 Prediction

Let the testing sentence be $S = w_1, w_2, w_3, ..., w_k, , w_n$ with the actual word sense of w_k to be predicted.

We search all possible senses $s_1, s_2, ..., s_n$ of w_k on WordNet-style English Dictionary, and then we perform our counting task based on the provided annotated training data. In the end, we're able to have our result, that is the most likely word sense of w_k, by applying (4).

Note that, in real world, we may be confronted by the computation insufficiency caused by the numerous annotated training data. In this case, we can set an upper limit of the consumed time of searching data which contain w_k to collect a subset data, and we only make use of it to perform counting task.

2 The Design of a POS Tagger

In the task, we design a POS tagger to predict the parts of speech of words in one sentence.

2.1 Deep into the Provided Annotated Training Data

What we do in this step is the same as the one subscribed in **1.1 Deep into the Provided Annotated Training Data**. Likewise, we still have the same three strategies to tackle our data here.

2.2 Model: Mainly Apply Hidden Markov Model

Typical Hidden Markov Model with bi-gram LM is applied to help find the best sequence of POS.
Core formula is shown

$$\begin{aligned}[y*_i] &= \arg\max_{y_i} p(y_1)[p(y_2|y_1)p(y_3|y_2)...p(y_n|y_{n-1})] \prod_i^n p(x_i, y_i) \\ &= \arg\max_{y_i} LookupTheTable(y_i) \prod_i^n Count(x_i, y_i) \end{aligned} \tag{5}$$

For the part of bi-gram LM, considering the number of possible combinations is quite limited, we parse every two parts of speech of any adjacent words in all provided annotated training data and store the all possible $p(y_i|y_j)$ in a database(or any file). In the following steps, we can have a quicker access to the $p(y_i|y_j)$ when needed. The term $LookupTheTable(.)$ in (5) is exactly what we undergo here.

As to the probability $p(x_i, y_i)$, we apply *Good Turing Discounting* Method to avoid the zero probability problem.

2.3 Prediction

Base on the Hidden Markov Model, the prediction is simply to choose the one with highest value of all possible POS sequences.

What's more, instead of brute force method where we calculate every single POS sequence, *the Viterbi Algorithm* will be implemented to obtain better time complexity.
Here shows the core formula

$$\pi(i, u, v) = \max_{w \in S_{i-2}} \pi(i-1, w, u) p(v|w, u) p(x_i|v) \tag{5}$$

Our objective function goes to

$$\max_{u \in S_{n-1}, v \in S_n} \pi(n, u, v) p(STOP|u, v)$$

3 Analysis of the Two Tasks

The WSD system is easier to implement than the POS tagger, in that the main effort of WSD is covered by the one of POST tagger. The term $\prod_{v_x \in C'}(Count(v_x, s_k) + 1)$ in (4) requires the same effort as the term $\prod_i^n Count(x_i, y_i)$ in (5), while there is the other term $LookupTheTable(y_i)$ in (4) left to work on.

Moreover, to cope with the part $LookupTheTable(y_i)$ in POS tagger, we need storage to store the counting result which loads the implementation with extra burden.

References

[1] Kenneth W. Church and William A. Gale. A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech & Language*, 5(1):19 – 54, 1991.

[2] I. J. GOOD. THE POPULATION FREQUENCIES OF SPECIES AND THE ESTIMATION OF POPULATION PARAMETERS. *Biometrika*, 40(3-4):237–264, 12 1953.

[3] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69, February 2009.

[4] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.