600.668 Machine Translation
Homework 2: Word Alignment

Kun Qian, Szu-Jui Chen, Xutai Ma

October 4, 2017

1 IBM Model 1

After implementing the basic EM of IBM Model 1 for 15 iterations, we changed to use a threshold for decoding. Then, by tuning the thresholding value, we got lower AER. To get better result, we combined the English-French model and French-English model.
The result is below:

model	AER(%)
IBM-M1	33.0
IBM-M1 with default threshold	36.3
IBM-M1 threshold=0.4	30.6
IBM-M1 (E-F & F-E)	28.2

2 IBM Model 2

The IBM-Model2 is based on IBM-Model1 by adding an alignment probability which is irrelevant with actual words[1]:

$$p(f, a \mid e) = \prod_{i=1}^{m} q(a_i \mid i, l, m) \cdot p(f_i \mid e_{a_i})$$

The way we implement this model is to multiply the probability in IBM-Model1 with the alignment probability which is related to the length of both English and French sentence and the location of target words.
In other word, for each line, to calculate:

$$\delta(i, j) = \frac{q(j|i, l, m) \cdot p(f_i \mid e_j)}{\sum_{j=0}^{l} q((j \mid i, l, m) \cdot p(f_i \mid e_j)}$$

Use that term to add up counts for pair of (f_i, e_j) and (i, j, l, m) and so on.
And then, use the counts to update the probabilities.
Also, for IBM-Model 2, we also tried different ways to improve it.

1. using threshold for decoding
2. combining best-one and threshold for decoding
3. combining English-French model and French-English model
4. using both two combinations above
5. combining IBM-M1 and IBM-M2

Part of good results are listed below:

model	AER(%)
IBM-M2	25.9
IBM-M2 threshold=0.04	24.0
IBM-M2 combined decoding with t=0.02	21.3
IBM-M2 (E-F & F-E)	21.2
IBM-M1 & IBM-M2	31.9

3 Bayesian Word Alignment

3.1 mathematical expression:

The core equation is:

$$P(a_j = i \mid E, F, A^{\neg j}; \Theta) = \frac{N^{\neg j}_{e_i, f_i} + \theta_{e_i, f_i}}{\sum_{f=1}^{V_F} N^{\neg j}_{e_i, f} + \sum_{f=1}^{V_F} \theta_{e_i, f}}$$

This is the Gibbs sampling formula for individual alignments. Here, a_j means the alignment from English word e_j to French word f_i. E and F represent all the English and French words in that sentence. A is all the alignments except the $a_j = i$. The Θ is the set of hyperparameter of the distribution. $N^{\neg j}_{e_i, f}$ means the total number of alignments from e_i *to* f_i exclude the $a_j = i$.
Here is the pseudo code:

```
Input: E, F;  Output: K samples of A
Initialize A
for k = 1 to K do
    for each sentence-pair s in (E, F) do
        for j = 1 to J do
            for i = 0 to I do
                Calculate P(a_j = i| ··· )
                according to (7)
            Sample a new value for a_j
```

Figure 1: Pseudo Code [2]

To start, we need to initialize alignment randomly and build a count table for storing the counts for alignments e_i align to f_i. Then we will do Gibbs sampling for an English word in a sentence and update the alignment and table after it. Keep running this for at least 1000 iteration.

We implemented it based on IBM Model 1. It takes lots of time for running (estimated more than one week for 1000 iterations). If we use only few sentences or few iterations, the AER is pretty high.

4 Hidden Markov Model Alignment

In this section, we implemented a hidden markov model(HMM) alignment, proposed in [3]. Further, we use the method in [4] to train the HMM, forward and backward algorithm, which is different from original paper [3].

4.1 Mathematical expressions

First, define several terminology

$$\begin{aligned}
&P(i \mid i', I) = P(a_j = i \mid a_{j-1} = i', I) \\
&\pi_j = P(a_1 = j) \\
&\alpha_i(j) = P(f_1 \dots f_j, a_j = i) \\
&\beta_i(j) = P(f_{j+1} \dots f_J \mid a_j = i)
\end{aligned} \tag{1}$$

Where I, J is the lengths for one english french sentenece pair.

The set of parameters are $\theta = \{\pi_j, P(i \mid i'I), P(f_i|e_j)\}$

Then forward-backward algorithm is implemented. For forward procedure, we have

$$\begin{aligned}
&\alpha_i(1) = \pi_j p(f_1|e_1), \\
&\alpha_i(j) = p(f_j \mid e_i) \sum_{i'=1}^{I} \alpha_{i'}(j-1) p(i \mid i', I)
\end{aligned} \tag{2}$$

and for backward procedure,

$$\begin{aligned}
&\beta_{i'}(J) = 1, \\
&\beta_{i'}(j-1) = \sum_{i=1}^{I} \beta_{i'}(j) p(i \mid i', I) p(f_j \mid e_i)
\end{aligned} \tag{3}$$

In practice, because of precision problem, we use re-normalization in every step of forward and backward procedure. Define $Q_j = \sum_i^I \alpha_i(j)$, then make

$$\begin{aligned}
\alpha_i^*(j) &= \frac{\alpha_i(1)}{\Pi_{j'=0}^{j} Q_{j'}} \\
\beta_i^*(j) &= \frac{\alpha_i(1)}{\Pi_{j'=j}^{J} Q_{j'}}
\end{aligned} \tag{4}$$

The forward and backward procedure become,

$$
\begin{aligned}
&\alpha_i^*(0) = \pi_i p(f_j|e_i), \\
&\alpha_i^*(j) = \frac{1}{Q_j} p(f_j \mid e_i) \sum_{i'=1}^{I} \alpha_i^*(j-1) p(i \mid i', I) \\
&\beta_N^*(j) = \frac{1}{Q_T}, \\
&\beta_i^*(j-1) = \frac{1}{Q_{j-1}} \sum_{i=1}^{I} \beta_{i'}(j) p(i \mid i', I) p(f_j \mid e_i)
\end{aligned}
\tag{5}
$$

Next, according Bayes rule, we have

$$
\begin{aligned}
\gamma_i(j) &= P(a_i = j \mid \mathbf{f}, \theta) \\
&= \frac{\alpha_i(j)\beta_i(j)}{\sum_{i'=1}^{N} \alpha_{i'}(j)\beta_{i'}(j)} \\
&= \frac{\alpha_i^*(j)\beta_i^*(j) Q_j}{\sum_{i'=1}^{N} \alpha_{i'}^*(j)\beta_{i'}^*(j)}
\end{aligned}
\tag{6}
$$

and,

$$
\begin{aligned}
\zeta_{ti'i}(j-1) &= P(a_{j-1} = i', a_j = i \mid \mathbf{f}, \theta) \\
&= \frac{\alpha_i(j) p(i \mid i', I) \beta_i(j) p(f_j \mid e_i)}{\sum_{l=1}^{I} \alpha_i(J)} \\
&= \frac{\alpha_i^*(j) p(i \mid i', I) \beta_i^*(j) p(f_j \mid e_i)}{\sum_{l=1}^{I} \alpha_i^*(J)}
\end{aligned}
\tag{7}
$$

We count follow statistics

$$
\begin{aligned}
c(f, e) &= \sum_{(\mathbf{f},\mathbf{e}) \in \mathbf{D}} \sum_{i,j} \gamma_i(j) \delta(f_j, f) \delta(e_i, e) \\
c_d(d) &= \sum_{i,i'} \delta(i - d, i') \\
c(i, i', I) &= \sum_{(\mathbf{f},\mathbf{e}) \in \mathbf{D}} \sum_{i,i'} c_d(i - i') \\
c_{init}(i) &= \sum_{(\mathbf{f},\mathbf{e}) \in \mathbf{D}} \gamma_i(0) \delta(|\mathbf{e}|, I)
\end{aligned}
\tag{8}
$$

and update the parameters,

$$
\begin{aligned}
&p(f \mid e) = \frac{c(f, e)}{\sum_f c(f, e)} \\
&\pi_i = p(a_0 = i|\mathbf{f}, \theta) = \frac{c_{init}(i)}{\sum_{i'} c_{init}(i')} \\
&p(i \mid i', I) = \frac{c(i, i', I)}{\sum_{i', I} c(i, i', I)}
\end{aligned}
\tag{9}
$$

After we have the parameters, we can decode the alignment based on Viterbi search

$$\begin{aligned} V[i,1] &= p(i)p(f_q|e_i) \\ V[i,j] &= max_{i'}\left\{V[i',j-1]p(i \mid i',I)p(f_j \mid e_i)\right\} \end{aligned} \tag{10}$$

Then find the maximum V[i, J], and trace back the path, we can get the alignment.

4.2 Experiment

The experimental set up and performance are shown as following table. The first one is a toy experiment. Since this experiment is very time consuming and we only run it on personal computer, we only train the original HMM word alignment without any further variants that has the potential to improve the performance.

Experiment set up	dev	time(s)
Number of words : 1000 Epochs: 30	0.357	506
Number of words : 1000000 Epochs : 1	0.368	4676
Number of words : 1000000 Epochs : 2	0.301	9373
Number of words : 1000000 Epochs : 3	0.270	14086
Number of words : 1000000 Epochs : 4	0.258	18889
Number of words : 1000000 Epochs : 5	0.258	23606
Number of words : 1000000 Epochs : 6	0.259	28390
Number of words : 1000000 Epochs : 7	0.238	33093
Number of words : 1000000 Epochs : 8	0.245	37821
Number of words : 1000000 Epochs : 9	0.248	42646

4.3 Code Description

- Train:
  python hmm.py -n NUM_TRAIN_SENTENCE -s PREFIX_PARAMETER
- Search:
  python viterbi.py -n NUM_SEARCH_SENTENCE -m PARAMETWER > hmm.a
- Evaluate:
  python score-alighment < hmm.a

While training, we save parameter after every epoch.

References

[1] Collins, M., 2011. Statistical machine translation: IBM models 1 and 2. Columbia Columbia Univ.

[2] MERMER, Coşkun; SARAÇLAR, Murat. Bayesian word alignment for statistical machine translation. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2. Association for Computational Linguistics, 2011. p. 182-187.

[3] Vogel, Stephan, Hermann Ney, and Christoph Tillmann. "HMM-based word alignment in statistical translation." Proceedings of the 16th conference on Computational linguistics-Volume 2. Association for Computational Linguistics, 1996.

[4] Bigvand, Anahita Mansouri. "Word Alignment for Statistical Machine Translation Using Hidden Markov Models." (2015).