Chinahadoop NLP course note

Xing Shi

May 10, 2017

Chapter 1

Language Model

1.1 Interpolate Smoothing

We want to mix the trigram, bigram and unigram information:

$$\begin{aligned} P_{int}(w_i|w_{i-1}, w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1}, w_{i-2}) \\ + \lambda_2 P_{ML}(w_i|w_{i-1}) \\ + \lambda_1 P_{ML}(w_i) \end{aligned} \tag{1.1}$$

$$\lambda_3 + \lambda_2 + \lambda_1 = 1 \tag{1.2}$$

Usually, we have both *Training data* and development data. P_{ML} is estimated by counting the training data. The smoothing parameters λs are tuned on development data using EM algorithm.

We want to maximize the log-likelihood on dev set:

$$L(\lambda_3, \lambda_2, \lambda_1) = \sum_h count(h) \log(\sum_{i=1}^{3} \lambda_i P_{ML,i}(h)) \tag{1.3}$$

where

$$h = (w_{i-2}, w_{i-1}, w_i) \tag{1.4}$$

$$P_{ML,1}(h) = P_{ML}(w_i) \tag{1.5}$$

$$P_{ML,2}(h) = P_{ML}(w_i|w_{i-1}) \tag{1.6}$$

$$P_{ML,3}(h) = P_{ML}(w_i|w_{i-1}, w_{w-2}) \tag{1.7}$$

and $count(h)$ is the count of trigram h in dev set. To direct optimize 1.3 is hard as **there is addition inside log**, thus we use **EM algorithm**. We add an auxiliary variable $q_i(h)$ with the constrain $\sum_{i=1}^{3} q_i(h) = 1$.

$$L = \sum_h count(h) \log(\sum_{i=1}^{K} \lambda_i P_{ML,i}(h)) \tag{1.8}$$

$$= \sum_h count(h) \log(\sum_{i=1}^{K} q_i(h) \frac{\lambda_i P_{ML,i}(h)}{q_i(h)}) \tag{1.9}$$

$$= \sum_h count(h) \sum_{j=1}^{K} q_j(h) log(\frac{\lambda_j P_{ML,j}(h)}{q_j(h)}) \tag{1.10}$$

$$+ \sum_h count(h) \sum_{j=1}^{K} q_j(h) [log(\sum_{i=1}^{K} \lambda_i P_{ML,i}(h)) - log(\frac{\lambda_j P_{ML,j}(h)}{q_j(h)})] \tag{1.11}$$

$$= \sum_h count(h) \sum_{j=1}^{K} q_j(h) log(\frac{\lambda_j P_{ML,j}(h)}{q_j(h)}) \tag{1.12}$$

$$+ - \sum_h count(h) \sum_{j=1}^{K} q_j(h) log(\frac{\frac{\lambda_j P_{ML,j}(h)}{\sum_{i=1}^{K} \lambda_i P_{ML,i}(h))}}{q_j(h)}) \tag{1.13}$$

$$= \sum_h count(h) \{E_{q(h)}[log(\frac{\lambda_j P_{ML,j}(h)}{q_j(h)})] + KL(q_j(h) || \frac{\lambda_j P_{ML,j}(h)}{\sum_{i=1}^{K} \lambda_i P_{ML,i}(h))}) \} \tag{1.14}$$

Here,

$$Q = \sum_h count(h) E_{q(h)}[log(\frac{\lambda_j P_{ML,j}(h)}{q_j(h)})] \tag{1.15}$$

$$\tag{1.16}$$

Q is the lower bound for L as the KL divergence is positive.

So the EM algorithm works as follows:

1. Random initialize λs
2. E step: make the $KL(q_j(h) || \frac{\lambda_j P_{ML,j}(h)}{\sum_{i=1}^{K} \lambda_i P_{ML,i}(h))})$ equal to zero by choosing $q_j(h) = \frac{\lambda_j^{old} P_{ML,j}(h)}{\sum_{i=1}^{K} \lambda_i^{old} P_{ML,i}(h))}$.
3. M step: substitute $q_j(h)$ in Q and maximize Q using Lagrangian with the constrain $\sum_{i=1} N\lambda_i = 1$. The update function is :

$$\lambda_j^{new} = \frac{\sum_h count(h) q_j(h)}{\sum_{i=1}^{K} \sum_h count(h) q_i(h)} \tag{1.17}$$

4. Repeat E-M steps until λs converge.

1.2 Absolute Discounting

It generally has the following form:

$$P_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{abs}(w_i|w_{i-n+2}^{i-1}) \quad (1.18)$$

Generally, $0 < D < 1$. To make it sum to 1, we could find that:

$$\lambda_{w_{i-n+1}^{i}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet) \quad (1.19)$$

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i|c(w_{i-n+1}^{i}) > 0\}| \quad (1.20)$$

Now, the only hyper-parameter left to estimate is D, and we will estimate it using *leave-one-out* methods.

Following [3], Let's first rewrite 1.18 to a simple format. We will represent the different n-gram occurrence w_{i-n+1}^{i} as k, and $k = 1, 2, 3, ..., K$, and $N(k) = count(k)$, $N = \sum_{k=1}^{K} N(k)$.

$$p(N(k)) = \frac{\max(N(k) - D, 0)}{N} + D\frac{K - n_0}{N} q(k) \quad (1.21)$$

Equation 1.21 is hard to evaluate, thus we go back to back-off model:

$$p(N(k)) = \begin{cases} \frac{N(k)-b}{N} & \text{if } N(k) > 0 \\ b\hat{q}(k) & \text{if } N(k) = 0 \end{cases} \quad (1.22)$$

Then we can easily calculate the log likelihood of the leave-one-out validations:

$$L = \sum_{k=1}^{K} N(K) \log p(N(k) - 1) \quad (1.23)$$

$$= \sum_{k:N(k)=1} 1 * \log b\hat{q}(k) + \sum_{k:N(k)>1} N(k) * \log \frac{N(k) - 1 - b}{N} \quad (1.24)$$

$$= constant(b) + n_1 \log b + \sum_{r>1} r n_r \log(r - 1 - b) \quad (1.25)$$

Calculate the partial directives w.r.t b,

$$\frac{\partial L}{\partial b} = n_1/b - \sum_{r>1} r n_r/(r - 1 - b) \quad (1.26)$$

Make the derivative equals zero:

$$n_1/b - 2n_2/(1-b) = \sum_{r>2} rn_r/(r-1-b) \geq 0 \tag{1.27}$$

Thus,

$$b \leq \frac{n_1}{n_1 + 2n_2} \tag{1.28}$$

1.3 Kneser-Ney Smoothing

KN smoothing is an extension of the absolute discounting. The difference is that KN smoothing models the unigram distribution $p(w_i)$ in a more clever way:

$$P_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{KN}(w_i|w_{i-n+2}^{i-1}) \tag{1.29}$$

$$p_{KN}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)} \tag{1.30}$$

The intuition is that we want to fill the blanks "I lost my reading __". Consider these two words "Francisco" and "glasses". It is quite possible that $p(\text{Francisco}) > p((glasses))$ due to "San Francisco" always appears in Daily News. However, "Francisco" always follow "San". Thus, when it backoff to unigram model, the unigram probability should be proportional to how many different words it follows in the training data.

1.4 Modified Kneser-Ney Smoothing

The "modified" part [1] is that D depends on $c(w_{i-n+1}^{i})$. And for count 1, 2 and 3+, we have three discounts D_1, D_2 and D_{3+}. For the implementation details, the only reliable material is [2].

1.5 Other Materials

Some other good materials about Language Model

1. Stanford NLP Courses: `https://web.stanford.edu/class/cs124/lec/languagemodeling.pdf`
2. A good slides: `https://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf`

3. Micheal Collins note: http://www.cs.columbia.edu/~mcollins/lm-spring2013.pdf

4. David Chiang's course: https://www3.nd.edu/~dchiang/teaching/nlp/2015/notes/chapter5v1.pdf

5. A good phd study report: https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2012/tr-2012-03.pdf

6. Koehn's slide about NNLM and RNN-LM: http://mt-class.org/jhu/slides/lecture-nn-lm.pdf

Bibliography

[1] Chen, S. F., and Goodman, J. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (1996), Association for Computational Linguistics, pp. 310–318.

[2] Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. Scalable modified kneser-ney language model estimation. In *ACL (2)* (2013), pp. 690–696.

[3] Ney, H., Essen, U., and Kneser, R. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language 8*, 1 (1994), 1–38.