Assignment 1 Summaries for three chapters/sessions

Wenyue Liu 730028157

Sec 4.5 Kneser-Ney Smoothing for N-gram Language Models

N-grams language model is a traditional language model which make Markov assumption. It provided the maximum likelihood estimate of next word based on left context. However, because of corpus sparsity and some spike distribution of words, smoothing (discounting) techniques are commonly used in N-grams LM.

Kneser-Ney smoothing which roots in **absolute discounting** is one of the most commonly used and best performing N-gram smoothing techniques. Based on Church and Gale's observation, held-out the bigram count in held-out corpus could be estimated by bigram count in training material subtracting constants for count is not too small (count = 0, 1). Therefore, by discounting a constant d, we can easily estimate count of bigram in the unknown corpus, and further use N-gram to find MLE. Not limited on absolute discounting which count unigrams/ bigrams/ trigrams only, Kneser-Ney method wants to include probability of continuation into it. For example, Kong has high unigram count however, it occurs only after Hong as Hong Kong. However, even 'glasses' has lower unigram count, but it can complete several bigram types, which means the $P_{\text{CONTINUATION}}$ of 'glasses' is much higher than $P_{\text{CONTINUATION}}$ of 'Kong'. Finally, KN method smoothing for bigrams shows as:

$$P_{\text{KN}}(w_i|w_{i-1}) = \frac{\max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{\text{CONTINUATION}}(w_i)$$

Where λ is the normalizing constant $\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)}|\{w : C(w_{i-1}w) > 0\}|$.

Compared to absolute discounting, KN smoothing introduced the concept 'continuation', which take the number of contexts this word completed in to consideration. Improved the traditional N-grams LM. In my opinion, this method is still a 'discounting' method, consider fixed number of grams in each combination, if mixed combination of unigram/bigram /trigram is considered and more specific constant applied, it will provide higher accuracy.

Sec 8.4 Training Neural Nets

Neural networks are an essential computational tool for natural language processing. Even Neural networks is an old concept, it attracts people's attention recently because of the limitation of computation power in neural network training. To train a Neural network, we need to define a metric for whether the system gets better or not. Loss function is the tool to measure distance between our system output and the golden output. Typical loss function can be mean-squared error (MSE) for regression problem, cross entropy loss (negative log likelihood) for classification problem.

To train the neural nets to find the minimum of loss function, gradient descent is a fantastic way for training algorithm. It is because it finds out by which direction, the function's slope is rising the most steeply, by moving in the opposite direction, training process will be finished faster. The gradient is calculated by partial derivative of the loss function with respect each parameter. In neural network, initial value x propagates through hidden layer and finally produces y hat. Another useful algorithm backprop allows information flow from cost backward to compute the gradient.

In addition, there are many other algorithms which use the gradient computed through backprop to achieve deep learning, these optimized methods (optimizers) have better performance compared to SGD, for example Adagrad, Adam etc.

Sec 8.5 Neural Language Models

Compared with traditional N-gram LM, Language Models based on Neural Network are much more powerful, however, these language models also come with slower training speed. In Neural Language Models, each word is embedded in to a vector space with length 50-500 (300 might be the best choice). In embedding vectors, similar words have similar embedding vectors. Therefore, the problem in N-gram that if system did not see a n-gram combination such as 'feed the dog', but only 'feed the cat', the system can't predict the last word is cat rather than dog. In Neural LM, because they have similar embedding vectors, the probability of 'cat' and 'dog' after 'feed the' should be similar.

By training the FFNNLM or RNN on some pretraining big corpus, people got some pretrained embedding model such as word2vec and glove. Even these embedding model is trained with other corpus, it is demonstrated these embedding is good enough for our purposes.

To train the model, for each previous word, we multiply it with its embedding vector and used as x in normal NN training. X multiply by weight; plus bias; after relu; multiply another matrix; apply softmax; will provide the final outcome y hat. Through loss function, backprop and optimizer, we will finally give an accurate prediction.

In this article, it introduced the FFNNLM, it based on precious words, and does not provide any weight on each previous words. However, in real world, the closer one word to our prediction, the higher importance it shows to the prediction. If we can 'forget' some of the precious words, it would enhance the prediction accuracy a lot.