n-gram Language Model

Ki Hyun Kim

nlp.with.deep.learning@gmail.com

What is good model?

- Generalization
  - Training(seen) data를 통해서 test(unseen) data에 대해 훌륭한 prediction을 할 수 있는가?

- 만약 모든 경우의 수에 대해 학습 데이터를 모을 수 있다면,
  table look-up으로 모든 문제를 풀 수 있을 것
  - 하지만 그것은 불가능하므로 generalization 능력이 중요

Count based Approximation

- Given sentence,

$$
\begin{aligned}
P(\text{<BOS>, I, love, to, play, <EOS>}) &= P(\text{<EOS>}|\text{<BOS>, I, love, to, play})P(\text{<BOS>, I, love, to, play}) \\
&= P(\text{<EOS>}|\text{<BOS>, I, love, to, play})P(\text{play}|\text{<BOS>, I, love, to})P(\text{<BOS>, I, love, to}) \\
&= P(\text{<EOS>}|\text{<BOS>, I, love, to, play})P(\text{play}|\text{<BOS>, I, love, to})P(\text{to}|\text{<BOS>, I, love})P(\text{<BOS>, I, love}) \\
&= P(\text{<EOS>}|\text{<BOS>, I, love, to, play})P(\text{play}|\text{<BOS>, I, love, to})P(\text{to}|\text{<BOS>, I, love})P(\text{love}|\text{<BOS>, I})P(\text{<BOS>, I}) \\
&= P(\text{<EOS>}|\text{<BOS>, I, love, to, play})P(\text{play}|\text{<BOS>, I, love, to})P(\text{to}|\text{<BOS>, I, love})P(\text{love}|\text{<BOS>, I})P(\text{I}|\text{<BOS>})P(\text{<BOS>})
\end{aligned}
$$

- We can approximate conditional probability by counting word sequence.

$$
P(\text{<EOS>}|\text{<BOS>, I, love, to, play}) \approx \frac{\text{COUNT}(\text{<BOS>, I, love, to, play, <EOS>})}{\text{COUNT}(\text{<BOS>, I, love, to, play})}
$$

- If we generalize this,

$$
P(x_n|x_{<n}) \approx \frac{\text{COUNT}(x_1, \cdots, x_n)}{\text{COUNT}(x_1, \cdots, x_{n-1})}
$$

Problem of Count based Approximation

- What if there is no such word sequence?

$$P(<\text{EOS}>|<\text{BOS}>, \text{I, love, to, play}) \approx \frac{\text{COUNT}(<\text{BOS}>, \text{I, love, to, play, } <\text{EOS}>)}{\text{COUNT}(<\text{BOS}>, \text{I, love, to, play})}$$

Apply Markov Assumption

- Approximate with counting only previous k tokens.

$$P(x_n|x_{<n}) \approx P(x_n|x_{n-1}, \cdots, x_{n-k}) \approx \frac{\text{COUNT}(x_{n-k}, \cdots, x_n)}{\text{COUNT}(x_{n-k}, \cdots, x_{n-1})}$$

- if $k = 2$,

$$P(x_n|x_{<n}) \approx P(x_n|x_{n-1}, x_{n-2}) \approx \frac{\text{COUNT}(x_{n-2}, x_{n-1}, x_n)}{\text{COUNT}(x_{n-2}, x_{n-1})}$$

If we expand this to sentence level,

- Now, we can cover more word sequences,
  even if they are unseen in training corpus.

$$\log P(x_{1:n}) = \sum_{i=1}^{n} \log P(x_i | x_{<i})$$

$$\approx \sum_{i=1}^{n} \log P(x_i | x_{i-1}, \cdots, x_{i-k})$$

n-gram

- $n = k + 1$

k	n-gram	명칭
0	1-gram	uni-gram
1	2-gram	bi-gram
2	3-gram	tri-gram

4-gram 부터는 그냥 four-gram...

n-gram

- n이 커질수록 오히려 확률이 정확하게 표현되는데 어려움
  - **적절한 n을 사용**하자

- 보통은 3-gram을 가장 많이 사용

- corpus(말뭉치)의 양이 많을 때는 4-gram을 사용하기도
  - 언어모델의 성능은 크게 오르지 않는데 반해,
  - 단어 조합의 경우의 수는 exponential하게 증가하므로 **효율성이 없음**

How to Train/Inference n-gram LM?

- SRILM
  - download: http://www.speech.sri.com/projects/srilm/download.html

- ngram-count: LM을 훈련
  - vocab: lexicon file name
  - text: training corpus file name
  - order: n-gram count
  - write: output countfile file name
  - unk: mark OOV as
  - kndiscount$n$: Use Kneser-Ney discounting for N-grams of order n
- ngram: LM을 활용
  - ppl: calculate perplexity for test file name
  - order: n-gram count
  - lm: language model file name

Summary

- 확률값을 근사하는 가장 간단한 방법은 코퍼스에서 빈도를 세는 것.
  - 하지만 복잡한 문장일수록 코퍼스에서 출현 빈도가 낮아, 부정확한 근사가 이루어질 것.

- 따라서 Markov assumption을 도입하여 확률값을 근사하자
  - 이제, 학습 코퍼스에서 보지 못한 문장에 대해서도 확률값을 구할 수 있다.
  - n의 크기가 중요함.
    - $n = 3 \sim 4$가 적당