08 N-Gram Language Model

Language Model

Language model assign a probability to a sequence of words: $P(w_1, w_2, \ldots, w_m)$
通过语言模型, 我们可以给一句话计算一个概率.
也可以做到给定前文, 计算下一个单词的概率分布.
除此之外, 因为我们有了下一个词的 conditional distribution, 我们还可以从一个 initial word 出发, 生成文本.
可用于:

- Speech recognition
- Spelling correction
- Machine translation
- Query completion
- Optical character recognition

Chain Rule

使用 chain rule 可以把$P(w_1, w_2, \ldots, w_m)$变成一串相乘的 conditional probability:

$$P(w_1, w_2, \ldots, w_m) = P(w_1)\ P(w_2 \mid w_1)\ P(w_3 \mid w_1, w_2) \cdots P(w_m \mid w_1, \ldots, w_{m-1})$$

N-Gram Model (the Markov Assumption)

但是把概率公式写成上面这个形式, 依旧很难搞. 这时候如果我们使用The markov assumption, 假设本 word 的概率只和自己和前 n-1 个 word 有关, 那么概率公式就可以更加简化. 这个就叫做 n-gram model.

- 5-gram model 是目前业界最常用的, but higher order sometimes can be used if large amounts of data are available.

Unigram Model

When n=1, 一个 word 的概率只和自己有关(一共涉及1 word):

$$p(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} p(w_i)$$

$$p(w_i) = \frac{C(w_i)}{M}$$ where $C(w_i)$是语料库中w_i出现的次数, M是整个语料库的单词数 (总 freq).

Bigram Model

When n = 2, 一个 word 的概率只和自己和前1个 word有关(一共涉及2 word):

$$p(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} p(w_i \mid w_{i-1})$$

$$p(w_i \mid w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})}$$ where $C(w_{i-1} w_1)$ 是语料库中 Wi-1和 Wi 连续出现的次数

Trigram Model

When n = 2, 一个 word 的概率只和自己和前1个 word有关(一共涉及3 word):

$$p(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} p(w_i \mid w_{i-2}, w_{i-1})$$

$$p(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2} w_{i-1} w_i)}{C(w_{i-2} w_{i-1})}$$

句子开头<S>结尾</S>

在 n-gram model 中, 在每个 sequence of words 开头补上 n-1 个 <s> 结尾补上一个 </s>. 这样开头n-1 个 <s>加上第一个word 能形成第一个 n-gram, 结尾 n-1 个 word 加上</s>是我们的最后一个 n-gram.

Smoothing (To Solve Unseen Problem)

如果一个词或者一串词组合(n-gram)从没在语料库出现过, 那么它的 count 就会是 0, 一旦有这么一个 0 出现, 最后整个概率公式都会是 0. 所以我们需要 smoothing for unseen n-gram.
Smoothing 的基本思想: 从别的出现过的 n-gram 哪里抢一点概率来给 unseen n-gram, 让它的概率起码不要是0. 注意概率有一个原则就是必须 sum up to 1, 我们把概率抢来抢去的时候也必须注意保证这一原则.

Laplacian (Add-One) Smoothing

基本思想: 在 n-gram model 中, 对于每个可能的 n-gram ,假装我们在实际见到它之外已经见过1次. 这样 0 freq 的 n-gram 就变成了 1 freq.

For unigram model: $p_{add1}(w_i) = \frac{C(w_i)+1}{M+|\mathbf{V}|}$ where M是整个语料库的单词数 (总 freq), $|\mathbf{V}|$是词汇表 size, 也就是 unique unigram 的个数

For bigram model: $p_{add1}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)+1}{C(w_{i-1})+|\mathbf{V}|}$

For trigram model: $p_{add1}(w_i|w_{i-2},w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i)+1}{C(w_{i-2}w_{i-1})+|\mathbf{V}|}$

Add-K Smoothing

Adding one is often too much. Instead, add a fraction k. Have to pre-choose a k value.

For trigram model: $p_{add\_k}(w_i|w_{i-2},w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i)+k}{C(w_{i-2}w_{i-1})+k|\mathbf{V}|}$

Kneser-Ney Smoothing

State-of-the-art smoothing method for n-gram language models

包含了四种思想:

Backoff

基本思想: Fall back to n-1 gram only when n-gram counts are zero

$$P_{BO}(w_i|w_{i-2},w_{i-1}) = \begin{cases} P^*(w_i|w_{i-2},w_{i-1}) & if\ C(w_{i-2},w_{i-1},w_i)>0 \\ \alpha(w_{i-2},w_{i-1}) * P_{BO}(w_i|w_{i-1}) & otherwise \end{cases}$$

variety of the original probability calculation; 如果词串Wi-2Wi-1Wi曾经出现过; a function to ensure "sum to 1"; 如果词串Wi-2Wi-1Wi从未出现过

P^* is a variety version of the original count probability calculation. $\alpha(\ )$ is a function. They must preserve "sum-to-1" property.

Interpolation

基本思想: Taking a linear combination of all relevant probabilities

$$P_{interp}(w_i|w_{i-2},w_{i-1}) = \lambda(w_{i-2},w_{i-1})\,P(w_i|w_{i-2},w_{i-1}) + (1-\lambda(w_{i-2},w_{i-1}))P_{interp}(w_i|w_{i-1})$$

original probability calculation using raw count

注意这个公式是recursive的, 最后一项$P_{interp}(w_i|w_{i-1})$带到整个公式, 会发现里面还包含 unigram probability. 所以说Interpolation的思想是综合所有相关的probabilities. $\lambda(\ )$是一个关于 context words 的 function, 我们可以把它无脑设置为一个常数, 亦可以设为一个根据 context words 变化的 function. Interpolation of probabilities 仍旧需要保证 "sum to 1". $\lambda(\ )$need to be trained on held out data.

Absolute Discounting

基本思想: 从别的出现过的 n-gram 哪里抢一点概率来给 unseen n-gram

$$P_{abs}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)-d}{C(w_{i-1})} + \lambda(w_{i-1},d)P(w_i)$$

d is a discount value which is a constant, $\lambda(\ )$ is a function of context words and d. They need to preserve "sum-to-1".

Continuation Counts

Back-off 和 interpolation 这两种方法会去掉或者削弱前文的影响力, 有些情况下这是非常不合理的. 比如说 P(Zealand|New) 很高, 但是 P(Zealand|Old) 就会很低, 如果去掉了前文, 但看 Zealand 的 count 是不合理的. 当 Zealand 前面没有 new 的时候, 我们并不希望给它一个很高的 probability.

continuation_count(w_i)= $|\{v: count(v\ w_i)>0\}|$,v代表曾在w_i 前面出现过的词

综合以上思想, Kneser-Ney smoothing 公式:

$$P_{KN}(w_i|w_{i-2},w_{i-1}) = \frac{\max(0, C_{KN}(w_{i-2},w_{i-1},w_i)-d)}{C_{KN}(w_{i-2},w_{i-1})} + \lambda(w_{i-2},w_{i-1})P_{KN}(w_i|w_{i-1})$$

special type of count; d: discount value we need to set d

where $\lambda(\ )$ is a function of context words:

$$\lambda(w_{i-2}, w_{i-1}) = \frac{d}{C_{KN}(w_{i-2}, w_{i-1})} |\{w: C_{KN}(w_{i-2}, w_{i-1}, w) > 0\}|$$

C_{KN} is a continuation count, except for the highest n-gram order: we use a regular count instead. n 大的时候不用, 比如说 5-gram 用普通 count,4-gram 用 continuation count.

- Best Kneser-Ney version uses different discount values d for each n-gram order. 2-gram, 3-gram, 4-gram 用不同的 d 值

Evaluation a Language Model

Extrinsic 外在的

How well it work on an actual task? Like spelling correction task / machine translation task?

Intrinsic 内在的

Calculate the metric called Perplexity on held-out test set.

Perplexity 困惑度

Perplexity is a metric to evaluate language models. The lower the better.

Perplexity

$$PP(w_1, w_2, .. w_m) = \sqrt[m]{\frac{1}{P(w_1, w_2, .. w_m)}}$$

$w_1, \ldots, w_m$ 是 held-out test set 中的句子, 这些句子都是 valid 句子, 如果我们的 language model 给这些句子 assign 了高概率, 那么这个 model 就是好 model.