不同语言模型smoothing技术

2020.04.19

Contents

- Recap
- Additive smoothing
- Good-Turing estimate
- Katz smoothing / Backoff
- Jelinek-Mercer smoothing / Interpolation
- Witten-Bell smoothing
- Absolute discounting
- Kneser-Ney smoothing
- Conclusion

Recap

Language Model（语言模型）

- A probability distribution $p(s)$ over strings s that describes how often the string s occurs as a sentence.

本质是概率分布或者统计语言模型

$$p(s) = p(w_1|\langle BOS\rangle) \times p(w_2|\langle BOS\rangle w_1) \times \ldots \times p(w_l|\langle BOS\rangle w_1 \ldots w_{l-1}) \times p(\langle EOS\rangle|\langle BOS\rangle w_1 \ldots w_l)$$

$$= \prod_{i=1}^{l+1} p(w_i|\langle BOS\rangle w_1 \ldots w_{i-1})$$

其中，$\langle BOS\rangle$是开始符，$\langle EOS\rangle$是结束符

- Example:

<BOS> John read a book <EOS>

基于2元文法的概率为:

p(John read a book) = p(John|<BOS>)×

p(read|John)×p(a|read)×

p(book|a)×p(<EOS>|book)

Recap

- N-gram模型与极大似然估计：

$$p(s) = \prod_{i=1}^{l+1} p(w_i | w_1 \cdots w_{i-1}) \approx \prod_{i=1}^{l+1} p(w_i | w_{i-n+1}^{i-1})$$

where w_i^j denotes the words $w_i \cdots w_j$

$$p_{\mathrm{ML}}(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})}$$

Recap

假设语料库S由三个句子组成：

- “Brown read holy Bible”
- “Mark read a text book”
- “He read a book by David”

用Bigram和最大似然法(MLE)求P(Brown read a book)：

- P(Brown | <BOS>) = 1/3
- P(Read | Brown) = 1
- P(a | read) = 2/3
- P(book | a) = 1/2
- P(<EOS> | book) = 1/2

Recap

- 为什么要进行smoothing操作?

在N-gram模型中，如果某些词的组合没有出现，它的概率即为0，是不合理的。根据Balh的研究，用150万词的训练语料来训练trigram，测试语料中23%的trigram没有在训练语料中出现过。

- 数据平滑的基本思想：

Smoothing就是用来解决如上问题的，调整最大似然估计的概率值，把0概率调高和把高概率调低，“劫富济贫”，消除0概率，改进模型的整体正确率。

基本目标：测试样本的语言模型困惑度越小越好。

基本约束：$\sum_{w_i} p(w_i|w_1, w_2, \ldots, w_{i-1}) = 1$

Additive smoothing

- Add-one smoothing: Laplace smoothing，假设 we saw each n-gram one more time than we did.

对于2-gram 有：

$$p(w_i \mid w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i}[1 + c(w_{i-1}w_i)]}$$

$$= \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}$$

其中，*V* 为被考虑语料的词汇量(全部可能的基元数)。

Additive smoothing

假设语料库S由三个句子组成:

- “Brown read holy Bible”
- “Mark read a text book”
- “He read a book by David”

加1平滑通常情况下是一种很糟糕的算法，与其他平滑方法相比显得非常差，然而我们可以把加1平滑用在其他任务中，如文本分类，或者非零计数没那么多的情况下。

用Bigram和最大似然法（MLE）求P(Brown read a book)

- P(Brown | <BOS>) = 1/3
- P(Read | Brown) = 1
- P(a | read) = 2/3
- P(book | a) = 1/2
- P(<EOS> | book) = 1/2

用Bigram 和加法平滑法求P(Brown read a book)

- P(Brown | <BOS>) = 2/14
- P(Read | Brown) = 2/12
- P(a | read) = 3/14
- P(book | a) = 2/13
- P(<EOS> | book) = 2/13

Additive smoothing

- 对加 1 平滑进行改进， pretend we’ve seen each n-gram δ times more than we have, 当然，Gale & Church (1994) 吐槽了这种方法，说表现很差。

$$p_{\text{add}}(w_i | w_{i-n+1}^{i-1}) = \frac{\delta + c(w_{i-n+1}^{i})}{\delta |V| + \sum_{w_i} c(w_{i-n+1}^{i})}$$

其中$0 < \delta \leqslant 1$

Good-Turing estimate

- I. J. Good 于1953 年引用 Turing 的方法来估计概率分布。
- 基本思想：reallocate the probability mass of n-grams that occur $r+1$ times in the training data to the n-grams that occur r times. In particular, reallocate the probability mass of n-grams that were seen once to the n-grams that were never seen.
- Example1:

你在钓鱼，然后抓到了18条鱼，种类如下：10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel.

下一条鱼是 trout 的概率是多少？

1/18

下一条鱼是新品种的概率是多少？

不考虑其他，那么概率是 0，然而根据出现一次的概率来估计新事物，概率是 3/18

在此基础上，下一条鱼是 trout 的概率是多少？

肯定就小于 1/18，那么怎么估计呢？

Good-Turing estimate

- For each count r, we compute an adjusted count r^*:

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

where, n_r is the number of n-grams seen exactly r times.

- Then, we have:

$$p_{GT}(x : c(x) = r) = \frac{r^*}{N}$$

where, $N = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty} n_{r+1}(r+1) = \sum_{r=1}^{\infty} n_r r$

- 这样，原训练样本中所有事件的概率之和为：

$$\sum_{r>0} n_r \times p_r = 1 - \frac{n_1}{N} < 1$$

因此，有$\frac{n_1}{N}$的剩余的概率量就可以均分给所有的未见事件(r =0)

- Example1:

$r = 1$时，$r^*_{trout} = 2 \times \frac{1}{3} = \frac{2}{3}$, $p_{GT}(trout) = \frac{2/3}{18} = \frac{1}{27}$

Good-Turing estimate

- Example2: 假设有如下英语文本，估计2-gram概率：

<BOS>John read Moby Dick<EOS>
<BOS>Mary read a different book<EOS>
<BOS>She read a book by Cher<EOS>
……

从文本中统计出不同 2-gram 出现的次数：

<BOS> John	**15**
<BOS> Mary	**10**
……	
read Moby	**5**
……	

Good-Turing estimate

- Example2: 假设要估计以read开始的2-gram概率，列出以read开始的所有2-gram,并转化为频率信息：

r	n_r	r^*
1	2053	0.446
2	458	1.25
3	191	2.24
4	107	3.22
5	69	4.17
6	48	5.25
7	36	保持原来的计数

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

$n_{r+1} = 0$

得到r^*后，就可以应用公式计算概率：

$$p_r = \frac{r^*}{N}$$

其中，N为以read开始的bigram的总数（样本空间），即read出现的次数。

Good-Turing estimate

那么，以read开始，没有出现过的bigram的概率总和为：

$$p_0 = \frac{n_1}{N}$$

以read作为开始，没有出现过的2-gram的个数等于：

$$n_0 = |V_T| - \sum_{r>0} n_r$$

其中，$|V_T|$为语料的词汇量。那么，没有出现过的那些以read为开始的2-gram的概率平均为：$\frac{p_0}{n_0}$

r	n_r	r^*
1	2053	0.446
2	458	1.25
3	191	2.24
4	107	3.22
5	69	4.17
6	48	5.25
7	36	—

注意：$\sum_{r=0}^{7} p_r \neq 1$

因此，需要归一化处理：

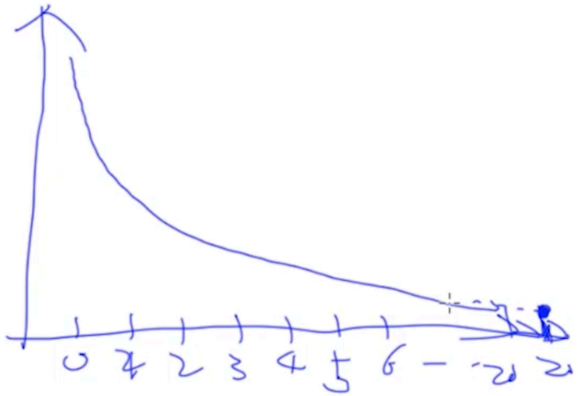
$$\hat{p}_r = \frac{p_r}{\sum_r p_r}$$

Good-Turing estimate

- Problem:
  - what if $n_{r+1} = 0$? This is common for high r : there are “holes” in the counts of counts.
  - Even if we’re not just below a hole, for high r, the n_r is quite noisy.
  - Really, we should think of r^* as:

$$r^* = (r+1)\frac{E[n_{r+1}]}{E[n_r]}$$

  - But how do we estimate that expectation?



  - Good-Turing thus requires elaboration to be useful. It forms a foundation on which other smoothing methods build.

Interpolation (插值) vs. Backoff (回退)

平滑的两种思想，一个是插值，简单来讲，就是把不同阶的模型结合起来，另一种是回退，直观的理解，就是说如果没有 trigram，就用 bigram，如果没有 bigram，就用 unigram。

- Both interpolation (Jelinek-Mercer) and Backoff (Katz) involve combining information from higher- and lower-order models.
- Key difference: in determining the probability of n-grams with nonzero counts, interpolated models use information from lower-order models while backoff models do not.
- Same point: In both backoff and interpolated models, lower-order models are used in determining the probability of n-grams with zero counts.
- It turns out that it’s not hard to create a backoff version of an interpolated algorithm, and vice-versa. (Kneser-Ney was originally backoff; Chen & Goodman made interpolated version.)
- Backoff requires fewer parameters, and can be directly determined without repeated estimations, so it is more convenient to implement.

Katz smoothing / Backoff

- 基本思想：当某一事件(n-gram)在样本中出现的频率大于阈值K（通常取0或1)时，运用最大似然估计的减值法来估计其概率，否则，使用低阶的，即(n-1)-gram的概率替代n-gram的概率，而这种替代需要受归一化因子α的作用。
- Consult the most detailed model first and, if that doesn't work, back off to a lower-order model:
  - If the trigram is reliable (has a high count), then use the trigram model.
  - Otherwise, back off and use a bigram model.
  - Continue backing off until you reach a model that has some counts.

Katz smoothing / Backoff

- Example:

If c(BURNISH THE)=0 and c(BURNISH THOU)=0, then under both additive smoothing and Good-Turing:

$$p(THE|BURNISH) = p(THOU|BURNISH)$$

This seems wrong: we should have

$$p(THE|BURNISH) > p(THOU|BURNISH)$$

Because THE is much more common than THOU, $p(THE) > p(THOU)$..

Solution: Use unigram instead, sometimes it's helpful to use less context

Katz smoothing / Backoff

以bigram为例：

- As in Good-Turing, we compute adjusted counts.
- Bigrams with nonzero count r are discounted according to discount ratio d_r , which is approximately $\frac{r^*}{r}$, the discount predicted by Good-Turing.
- Count mass subtracted from nonzero counts is redistributed among the zero-count bigrams according to next lower-order distribution (i.e. the unigram model).

Katz adjusted counts:

$$c_{katz}(w_{i-1}^i) = \begin{cases} d_r r & \text{if } r > 0 \\ \alpha(w_{i-1}) p_{ML}(w_i) & \text{if } r = 0 \end{cases}$$

$\alpha(w_{i-1})$ is chosen so that $\sum_{w_i} c_{katz}(w_{i-1}^i) = \sum_{w_i} c(w_{i-1}^i)$:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i:c(w_{i-1}^i)>0} p_{katz}(w_i|w_{i-1})}{1 - \sum_{w_i:c(w_{i-1}^i)>0} p_{ML}(w_i)}$$

Compute $p_{katz}(w_i|w_{i-1})$ from corrected count by normalizing:

$$p_{katz}(w_i|w_{i-1}) = \frac{c_{katz}(w_{i-1}^i)}{\sum_{w_i} c_{katz}(w_{i-1}^i)}$$

Jelinek-Mercer smoothing / Interpolation

- Instead of backing off, we could combine all the models, i.e. using evidence from unigram, bigram, trigram, etc.

$$
\begin{aligned}
p(w_n|w_{n-1}w_{n-2}) = {} & \lambda_1 p(w_n|w_{n-1}w_{n-2}) \\
& + \lambda_2 p(w_n|w_{n-1}) \\
& + \lambda_3 p(w_n)
\end{aligned}
$$

$$\sum_i \lambda_i = 1$$

- How to set lambdas:

Train different n-gram language models on the training data.
Use these language models, optimize lambdas to perform best on the development data.
Evaluate the final system on the test data.

Jelinek-Mercer smoothing / Interpolation

- Example:

If c(BURNISH THE)=0 and c(BURNISH THOU)=0, then under both additive smoothing and Good-Turing:

$$p(THE|BURNISH) = p(THOU|BURNISH)$$

This seems wrong: we should have

$$p(THE|BURNISH) > p(THOU|BURNISH)$$

Because THE is much more common than THOU, $p(THE) > p(THOU)$.
Solution: interpolate between bigram and unigram models:

$$p_{interp}(w_i|w_{i-1}) = \lambda p_{ML}(w_i|w_{i-1}) + (1-\lambda)p_{ML}(w_i)$$

Jelinek-Mercer smoothing / Interpolation

- Recursive formulation: nth-order smoothed model is defined recursively as a linear interpolation between the nth-order ML model and the (n − 1)th-order smoothed model.

$$p_{interp}(w_i|w_{i-n+1}^{i-1}) =$$
$$\lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{interp}(w_i|w_{i-n+2}^{i-1})$$

- Can ground recursion with:

– 1st-order model: ML (or otherwise smoothed) unigram model (MLE)

– 0th-order model: uniform model

$$p_{unif}(w_i) = \frac{1}{|V|}$$

- $\lambda_{w_{i-n+1}^{i-1}}$ can be estimated using EM on held-out data (held-out interpolation) or in cross-validation fashion (deleted interpolation).
- The optimal $\lambda_{w_{i-n+1}^{i-1}}$ depends on context: high-frequency contexts should get high λs.

Witten-Bell smoothing

- An instance of Jelinek-Mercer smoothing.

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) p_{WB}(w_i|w_{i-n+2}^{i-1})$$

- $1-\lambda_{w_{i-n+1}^{i-1}}$ should be the probability that a word not seen after w_{i-n+1}^{i-1} in training data occurs after that history in test data.
- Estimate this by the number of unique words that follow the history w_{i-n+1}^{i-1} in the training data.
- To compute the λs, we'll need the number of unique words that follow the history w_{i-n+1}^{i-1}:

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

- Set λs such that:

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

Witten-Bell smoothing

- Example:

考虑spite和constant的bigram，在Europarl corpus中，两个bigram都出现了993次，以spite开始的bigram只有9种，大多数情况下spite后面跟着of (979次)，因为in spite of是常见的表达，而跟在constant后的单词有415种，所以，我们见到一个跟在constant后面的bigram的可行性更低，Witten-Bell平滑就考虑了这种单词预测的多样性，所以：

$$1 - \lambda_{spite} = \frac{9}{9+993} = 0.00898$$

$$1 - \lambda_{constant} = \frac{415}{415+993} = 0.29474$$

Absolute discounting

- Like Jelinek-Mercer, involves interpolation of higher- and lower-order models.
- But instead of multiplying the higher-order p_{ML} by a λ, we subtract a fixed discount $\delta \in [0, 1]$ from each nonzero count.
- 以bigram为例:

discounted bigram

Interpolation weight

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigram

Absolute discounting

- Complete equation:

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

- To make it sum to 1:

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

- Choose δ using held-out estimation. 一般来说， δ就直接取 0.75 好啦~

Kneser-Ney smoothing

- An extension of absolute discounting with a clever way of constructing the lower-order (backoff) model.
- Idea: the lower-order model is significant only when count is small or zero in the higher-order model, and so should be optimized for that purpose.
- Example:

```
I can't see without my reading ________
```

Choice: glasses/Francisco

- If we've never seen the bigram "reading glasses", we'll back off to just $p(glasses)$.
- "San Francisco" is quite common, therefore "Francisco" will get a high unigram probability:

$$p(Francisco) > p(glasses)$$

- Then absolute discounting will give a high probability to "Francisco" appearing after novel bigram histories.
- Better to give "Francisco" a low unigram probability, because the only time it occurs is after "San", in which case the bigram model fits well.

Kneser-Ney smoothing

- Solution: Instead of looking at how likely is w, $p(w)$, we want to use $p_{continuation}(w)$, how likely is w to appear as a novel continuation.

$$N_{1+}(\bullet\ w_i) = |\{w_{i-1} : c(w_{i-1}w_i) > 0\}|$$

- The total number of word bigram types:

$$N_{1+}(\bullet\ \bullet) = \sum_{w_i} N_{1+}(\bullet\ w_i)$$

- $p_{continuation}(w_i) = p_{KN}(w_i) = \dfrac{N_{1+}(\bullet\ w_i)}{N_{1+}(\bullet\ \bullet)}$

- Put them all together:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - \delta, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{\delta}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\ \bullet) p_{KN}(w_i|w_{i-n+2}^{i-1})$$

Conclusion

Consider n-gram only:

Additive smoothing   Good-Turing smoothing

Consider with lower-order models:

Katz smoothing (Backoff)   Jelinek–Mercer smoothing (Interpolation)

- Absolute discounting
  - Kneser–Ney
- Witten–Bell

Conclusion

- 大部分平滑方法都可以写成：

$$p_{\text{smooth}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \tau(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1})p_{\text{smooth}}(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0. \end{cases}$$

- 如果n阶语言模型有非零的计数，就用分布$\tau\left(w_i \middle| w_{i-n+1}^{i-1}\right)$。
- 否则，就回退到低阶分布，选择$\gamma(w_{i-n+1}^{i-1})$使得概率和为1。
- Backoff 和 Interpolation 模型的根本区别在于，确定非零计数的n-gram 的概率时，Interpolation使用低阶分布的信息，Backoff模型没有。
- 不管时Backoff还是Interpolation，都使用了低阶分布来确定计数为0的n-gram的概率。

Conclusion

- The factor with the largest influence is the use of a modified backoff distribution as in Kneser-Ney smoothing.
- Jelinek-Mercer performs better on small training sets; Katz performs better on large training sets.
- Katz smoothing performs well on n-grams with large counts; Kneser-Ney is best for small counts.
- Absolute discounting is superior to linear discounting.
- Interpolated models are superior to backoff models for low (nonzero) counts.
- Adding free parameters (those black dots) to an algorithm and optimizing these parameters on held-out data can improve performance.

Adapted from Chen & Goodman (1998)

Thanks ❤

2020.04.19