文本表征学习

2. N-gram语言模型

EE1513

宋彦

大纲

- 什么是语言模型
- 概率语言模型
- 估计 N-gram 概率
- 语言模型的评价
- 平滑
  - 加一平滑
  - Good-Turing平滑

什么是语言模型

语言模型

- 什么是语言模型？
  - 一般的定义：建模语言使用的模型
- 语言是怎么使用的？
  - 词的任意顺序的组合？ 错

    我学习文本表征课程　　*我表征课程文本学习
  - 词特定顺序的组合？ 错

    *文本表征课程学习我

    乌龟对兔子说：“我早晚会赢过你的”
- 似乎有一些基本规则可以确定一个句子的格式是否正确

语言模型

- 我们可以使用基于规则的语言模型吗？

- 优点：
  - 可以解释为什么有些句子有效而有些无效
  - 不需要大量数据
  - ……

- 缺点：
  - 需要语言学知识
  - 知识系统难以维护
  - 难以解决歧义
  - ……

语言模型

- 我们可以使用数据驱动的语言模型吗？

- 优点：
  - 不需要语言学知识
  - 更容易维护
  - ……

- 缺点：
  - 难以解释
  - 模型性能在很大程度上取决于数据的数量和质量
  - 需要大量计算资源
  - ……

语言模型

- 语言模型可以应用于哪些应用？
  - 语音识别

    I bought two/too/to books.
  - 机器翻译

    我买了两本书 -> I bought two book/books
  - 输入法
  - 手写识别
  - …

语言模型

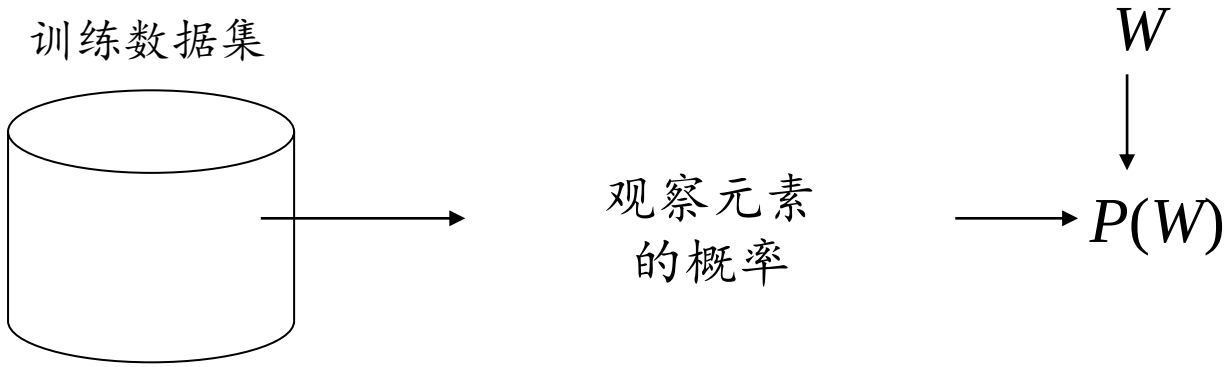
- 语言建模对文本表示学习有多重要?
  - Word2vec
  - ELMo
  - BERT/GPT

- 语言建模的特点是什么？
  - 应该从文本中学到什么
  - 影响语言建模的重要因素有哪些

概率语言模型

概率语言模型

- 目标：构建一个统计模型使得人们可以计算特定语言词序列 $W = w_1 w_2 \cdots w_n$ 的概率
- 一般的方法：



链式法则

- 链式法则：

$$P(X_1, X_2, \cdots, X_n) = P(X_1)P(X_2|X_1)\cdots P(X_n|X_1, \cdots, X_{n-1})$$

- 链式法则应用于语言模型N-gram 语言模型

$$P(w_1 \cdots w_n)$$
$$= p(w_1)p(w_2|w_1)\cdots p(w_n|w_1 \cdots w_{n-1})$$
$$= \prod_{i=1}^{n} P(w_i|w_1 \cdots w_{i-1})$$

马尔可夫假设

$$P(w_1 \cdots w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-k} \cdots w_{i-1})$$

- 换句话说，我们对乘积中的每项进行近似

$$P(w_i | w_1 \cdots w_{i-1}) \approx P(w_i | w_{i-k} \cdots w_{i-1})$$

两个最简单的案例:
Unigram/Bigram 模型

- Unigram 模型 $(k = 0)$:

$$P(w_1 \cdots w_n) \approx \prod_{i=1}^{n} P(w_i)$$

- 句子中的词不依赖于其他词

- Bigram 模型 $(k = 1)$:

$$P(w_1 \cdots w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-1})$$

- 单词依赖前一个单词

N-gram 语言模型

- 我们可以拓展到 tri-grams，4-grams，5-grams

- N-gram语言模型足够好吗？

- 通常地，N-gram语言模型不足以很好地建模语言使用
  - 因为它无法建模词之间的长距离依赖关系
  - 我学习文本表征课程 （“学习”与“课程”之间有依赖关系）

使用 N-gram 模型生成句子

- 如果我们有一个二元语言模型，给定 $w_1 w_2 \ldots w_n$，那么我们可以生成下一个词

- 如何选择下一个词w_{n+1}?
- 我们想最大化新词出现的概率

$$\hat{w}_{n+1} = \arg\max_{w \in V} P(w|w_n)$$

V 是所有词的词表

- 在实践中，我们在概率最高的 m 个候选词里面随机选择

N-gram 概率估计

以 bigram 语言模型为例

- Bigram 语言模型:

$$P(w_1 \cdots w_n) \approx \prod_{i=1}^{n} P(w_i|w_{i-1})$$

- 我们需要估计 $P(w_i|w_{i-1})$

估计 Bigram 概率

- 使用最大似然估计

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}w_i)}{count(w_{i-1})}$$

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

$c(\cdot)$ 代表n元组的数目

基于 Bigram 的句子概率估计

$$P(<s>\ I\ want\ english\ food\ </s>)$$

$$= P(I \mid <s>) \times P(want \mid I) \times P(english \mid want) \times P(food \mid english) \times P(</s> \mid food)$$

$$= 0.000031$$

- 如果句子很长，那么概率就会很小

- 实践中会有下溢问题

实际计算中的问题

- 计算很多小数的乘法时，我们可能会遇到下溢问题

- 我们使用 log 计算
  - 避免下溢
  - 加法计算速度比乘法快

$$\log\left(\prod_{i=1}^{n} p_i\right) = \sum_{i=1}^{n} \log p_i$$

语言模型工具

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

评价

模型评测

- 语言模型是否准确判断句子的好坏，即
  - 是否为“真实”或“经常观察到”的句子分配更高的概率
  - 是否为“不合语法”或“很少观察到”的句子更低的概率

- 在训练集上训练模型的参数，在未见过的数据上测试模型的性能
  - 测试集是一个看不见的数据集，与我们的训练集不同，完全未使用
  - 评估指标告诉我们模型在测试集上的表现如何

N-gram 模型的外部评估

- 比较模型 A 和 B
  - 将每个模型放入一个任务中
    - 拼写校正器、语音识别器、机器翻译系统
  - 运行任务，获得 A 和 B 的准确度
    - 正确纠正了多少拼写错误的单词
    - 正确翻译了多少字
  - 比较 A 和 B 的准确度

- 我们可以只使用外部评估吗？

N-gram 模型的外部评估的难度

- 外在评价
  - 耗时；可能需要几天或几周

- 内在评估：指标直接应用于模型
  - 困惑度： Perplexity

模型评测：困惑度

- 最好的语言模型是能够最好地预测未见过的测试集的模型
  - 给予最高 $P(W)$

$$PP(W) = P(w_1 w_2 \cdots w_n)^{-\frac{1}{n}}$$

- 困惑度是测试集的逆概率，由单词数归一化：
  - 链式法则：$PP(W) = (\prod_{i=1}^{n} P(w_i|w_1 \cdots w_{i-1}))^{-\frac{1}{n}}$
  - 对于 bigrams：$PP(W) = (\prod_{i=1}^{n} P(w_i|w_{i-1}))^{-\frac{1}{n}}$
- **最小化困惑度与最大化概率是等价的**
- 低困惑度意味着更好的模型

平滑

过拟合的危险

- 只有当测试语料库与训练语料库十分相似时，基于 n-gram 语言模型的单词预测才会准确
- 现实生活中，测试语料与训练语料往往不同
- 因此，我们需要训练具有泛化能力的鲁棒模型

- 当测试集中出现了训练集中没有的n-gram时，会有概率为0的问题

概率为0的问题

- 训练集:
  - … denied the allegations
  - … denied the reports
  - … denied the claims
  - … denied the request
- P(offer | denied the) = 0

- 测试集
  - … denied the offer
  - … denied the loan

Bigram 概率为0

- 概率为零的 bigram
  - 意味着我们将 0 概率分配给测试集！

- 因此我们无法计算困惑度（不能除以 0）！

- 因此，我们需要平滑！

加一平滑

加一平滑

- 假装我们比实际多看到每个单词一次
- 只需在所有计数上加一！

- 最大似然估计：

$$P_{MLE}(w_i \mid w_{i\text{-}1}) = \frac{c(w_{i\text{-}1}, w_i)}{c(w_{i\text{-}1})}$$

- 加一估计：

$$P_{Add\text{-}1}(w_i \mid w_{i\text{-}1}) = \frac{c(w_{i\text{-}1}, w_i)+1}{c(w_{i\text{-}1})+V}$$

V 是词表大小

加 δ 平滑

- 人们通常不对 n-gram 使用加一平滑
  - 加一给的 n-grams 频数太多了

- 解决方案：加 δ 平滑
  - 加一个小的频数 δ 到没见过的 n-gram 中

$$P(w_i|w_{i-1}) = \frac{\delta + c(w_i, w_{i-1})}{\delta * V + c(w_{i-1})}$$

  - 需要选择 δ 的值
  - 这个方法比加一平滑要好，但仍旧不理想

Good-Turing 平滑

基本想法

- 加 δ 平滑基于n-grams，这给未见过的n-grams增加了太多的概率
  - 通常，未见过的 n-gram 类型的数量相对较多，并且大多数不是语言中的有效 n-gram

- Good-Turing（GT）平滑
  - 假设未见过的 n-gram 的行为与出现一次的 n-gram 相似
  - 类似的情况同样适用 unigrams 到 bigrams，bigrams 到 trigrams，……

基本想法

- GT通过以下方式估计条件概率

$$P_{GT}(w_i|w_1,\ldots,w_{i-1}) = \frac{c^*(w_1,\ldots,w_{i-1},w_i)}{c^*(w_1,\ldots,w_{i-1})}$$

其中c^*是从 GT 估计中获得的新的 n-gram 计数。

- 如何获得c^*？

估计 c^*

常用的数目少，不常用的数目多

- 令 N_c 是出现了 c 次的不同 n-gram 的数量

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

- 对出现了 c 次的n-gram，我们假设它出现了 c^* 次：

$$P_{GT}(w_1, \ldots, w_n) = \frac{c^*(w_1, \ldots, w_n)}{N}$$

N 是数据集大小

- 如何计算未见过的 n-gram 的概率？

估计未见过的 N-gram 的概率

- 所有未见过的 n-gram 的概率可以计算为

$$
\begin{aligned}
p_0 &= 1 - \sum_{c>0} N_c \cdot p_c \\
&= 1 - \frac{1}{N} \sum_{c>0} N_c \cdot c^* \\
&= \frac{N - \sum_{c>0} N_c \cdot c^*}{N} \\
&= \frac{N - \sum_{c>0} N_{c+1} \cdot (c+1)}{N} \\
&= \frac{N_1}{N}
\end{aligned}
$$

$c^* = (c+1)\frac{N_{c+1}}{N_c}$

$N = \sum_{c>0} c \cdot N_c$

N_c 是出现了 c 次的不同 n-gram 的数量