2018年7月20日

语言模型

1. 背景

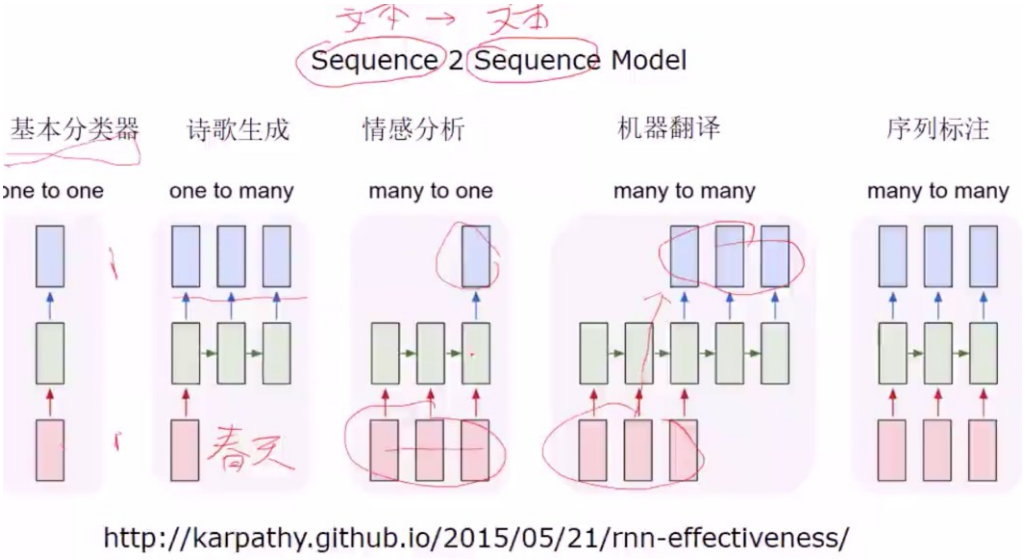
历史进程：

规则系统 概率系统（子任务误差累计，专家设计特征、流程） 深度学习（专家设计模型结构，end－to－end Training 子任务减少，特征自动提取）sequence to sequence

概率系统的一般工作方式

任务： “豆瓣酱用英语怎么说” => translate(豆瓣酱，Eng)

流程设计 “序列标注问题”	子任务1： 找出目标语言 “豆瓣酱用英语怎么说” 子任务2： 找出翻译目标 “豆瓣酱用英语怎么说”
收集训练数据	训练数据 （子任务1） “豆瓣酱用英语怎么说” “茄子用英语怎么说” “黄瓜怎么翻译成英语”
预处理	分词 “豆瓣酱 用 英语 怎么说”
抽取特征	前后各一个词 0 豆瓣酱： _ 用 0 用： 豆瓣酱 英语 1 英语： 用 怎么说 0 怎么说： 英语 _
分类器 “概率规则”	SVM/Neural Network ….
预测	0.1 茄子： _ 用 0.1 用： 豆瓣酱 0.9 英语： 用 怎么说 0.1 怎么说： 英语 _
评价	计算准确率



- Stanford Core NLP
  - 语义分析
- NLTK
  - 句子划分、读取语义树

2. 语言模型

描述词、语句乃至于整个文档这些不同的语法单元的概率分布的模型，能够用于衡量某句话或者词序列是否符合所处语言环境下人们日常的行文说话方式。（对所有文字进行概率上的打分）score（语言）

话题模型 score（话题｜语言）

A. 概述（目标）

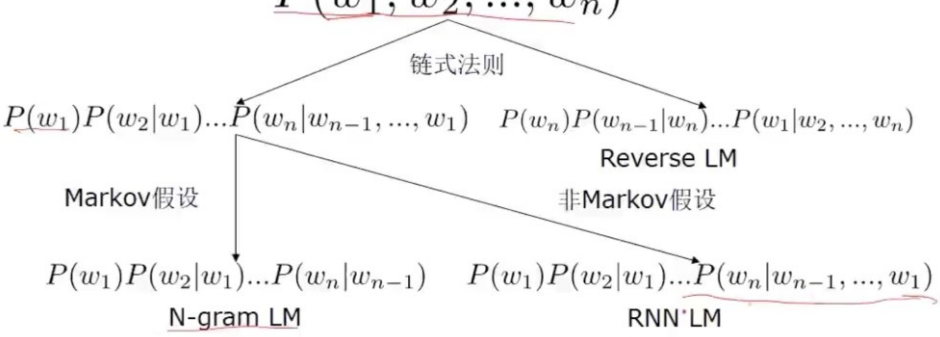
找到一个概率分布，它可以表示任意一个句子或序列出现的概率。

$$p(w_1, w_2, \dots, w_n)$$

目前在自然语言处理相关应用非常广泛，如语音识别(speech recognition)，机器翻译(machine translation)，词性标注(part-of-speech tagging)，句法分析(parsing)等。传统方法主要是基于统计学模型，最近几年基于神经网络的语言模型也越来越成熟。

$$P(sentence) = P(w_1, w_2, \dots, w_n)$$

$$\sum_{sentence \in L} P(sentence) = 1$$



正向语言模型（和后面词的关系）vs 反向语言模型（和前面词的关系）

B. 马尔科夫假设

（无记忆性）有限历史性假设： $p(x_n|x_1, x_2, \dots, x_{n-1}) = p(x_n|x_{n-1})$
下一个词的出现仅依赖于它前面的一个或几个词

C. 评价指标

i. 交叉熵(cross entropy)

（熵：描述一个随机变量的不确定性的数量，熵越大，不确定性越大，正确估计其值的可能性越小。越不确定的随机变量越需要大的信息量以确定其值。）

交叉熵：衡量估计模型和真实概率分布之间的差异：

$$\mathrm{H(L,q)} = -\lim_{n\to\infty}\frac{1}{n}\sum_{x_1^n} p(x_1^n) log_2 q(x_1^n)$$

L是一句话，q是估计模型。由于无法获取真实模型的概率，需要作出一个假设：假定 L 是稳态遍历的随机过程，即当 n 无穷大的时候，所有句子的概率和为 1。

$$\mathrm{H(L,q)} = \lim_{\mathrm{n}\to\infty}\frac{1}{\mathrm{n}}\log_2 \mathrm{q}(\mathrm{x}_1^{\mathrm{n}})$$

ii. 困惑度(ppl) perplexity

ppl 更容易计算，最小的 ppl 值意味着训练的模型最接近真实模型

$$\mathrm{PPL} = 2^{H(L,q)} \approx 2^{-\frac{1}{n}log_2 q(x_1^n)} = [q(x_1^n)]^{-\frac{1}{n}}$$

$$Perplexity(W_{test}) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 q(w_i)}$$

合同相同的词汇

$$= 2^{-\sum_{i=1}^{|V|}\frac{count(V_i)}{N}\log_2 q(v_i)}$$

$\hat{p}(v_i) = \frac{count(v_i)}{N}$

$$= 2^{-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)}$$

$-\log_2 p(v_i)$ 如果用概率分布q来编码v_i, 需要多少比特（bits）

$-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)$ v_i的分布服从p， 用q来编码v_i, 需要的比特数的期望。

$2^{-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)}$ W_{test}的等效状态的数目

$p(v_i) = \frac{count(v_i)}{N}$为$v_i$观察到的经验概率，|V|为词的集合，$q(v_i)$为 language model 求得的概率。如果$P(v_i)$为 0.5，需要 1bit 表示这个概率。

$2^{-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)}$ W_{test}的等效状态的数目

PPX(投掷硬币)= 2
PPX(投掷骰子)= 6

N-gram	Unigram	Bigram	Trigram
PPX	-bug-	219	174

ptb dataset by KenLM

{test 的等价状态数}
PPL 适合直接比较两个模型，交叉熵适合描述一个模型的提升。在描述模型的提升的时候，通常使用相比于基线降低的相对百分比

PPL	PPL after reduction	Relative PPL reduction	Entropy [bits]	Entropy after reduction	Relative entropy reduction
2	1.4	30%	1	0.49	51%
20	14	30%	4.32	3.81	11.8%
100	70	30%	6.64	6.13	7.7%
200	140	30%	7.64	7.13	6.7%
500	350	30%	8.97	8.45	5.8%
2000	1400	30%	10.97	10.45	4.7%

3. 统计语言模型

A. N –gram 模型（KenLM）

$$P(w_1, w_2, .., w_n) = P(w_1)P(w_2|w_1)...P(w_n|w_1, ..., w_{n-1})$$

p(S)=p(w1)p(w2|w1)p(w3|w1,w2)...p(wn|w1,w2,...,wn-1)

=p(w1)p(w2|w1)p(w3|w2)...p(wn|wn-1) // **bigram**

p(S)=p(w1)p(w2|w1)p(w3|w1,w2)...p(wn|w1,w2,...,wn-1)

=p(w1)p(w2|w1)p(w3|w1,w2)...p(wn|wn-1,wn-2) // **trigram**

$$P(w_1, w_2, ..., w_n) = P(w_1|START)P(w_2|w_1)...P(w_n|w_{n-1})P(EOS|w_n)$$

如何选择依赖词的个数 n:

- 更大的 n：对下一个词出现的约束信息更多，具有更大的**辨别力**；
- 更小的 n：在训练语料库中出现的次数更多，具有更可靠的统计信息，具有更高的**可靠性**。

B. OOV&平滑方法

Trigram Model

$$P(w_i|w_{i-1},w_{i-2}) = \frac{count(w_{i-2},w_{i-1},w_i)}{count(w_{i-2},w_{i-1})}$$

Training Set	Testing Set
我 喜欢 开车 我 喜欢 上网 我 喜欢 篮球 你 喜欢 编程	我 喜欢 上网 我 喜欢 编程 我 喜欢 王者荣耀

P(王者荣耀|我 喜欢) = 0 (Training中来没有出现的词)
→ OOV (Out of Vocabulary)

P(编程|我 喜欢) = 0　　(Training中没有出现的trigram)
→ Smoothing

OOV：

补 UNK 解决未出现在 train 中的 test 词

将出现概率比较小的词替换成 UNK

平滑：

降低已出现 n-gram 条件概率分布，以使未出现的 n-gram 条件概率分布非零，且经数据平滑后一定保证概率和为 1（平滑的意思是以防语言模型的概率相成等于 0 的情况出现）解决 train data 中未出现的组合在 test 中出现的问题。

A. 政府给大家每人都发一笔钱
B. 找父母要
C. 劫富济贫

- ☐ +1 平滑 (A)
- ☐ Back-off 回退法 (B)
- ☐ Interpolate 插值法 (B)
- ☐ Absolute Discount (C)
- ☐ Kneser-Ney Smoothing (C)
- ☐ Modified Kneser-Ney Smoothing (最优的方法) (C)

i. add-one（Laplace）Smoothing 加一平滑法，又称拉普拉斯定律（1920）表现一般，通常不用

$$P_{add}(w_i|w_{i-1},w_{i-2}) = \frac{\delta + count(w_{i-2},w_{i-1},w_i)}{\delta|V| + \sum_{w_i} count(w_{i-2},w_{i-1},w_i)}$$

ii. back - off 回退法（1987）

思路：
使用 Trigram 如果 count(trigram) 满足一定的条件
否则使用Bigram;
否则使用Unigram;　　具体参考 “Katz smoothing”

iii. interpolate 插值法

将Trigram, Bigram, Unigram线性组合起来:

$$P_{int}(w_i|w_{i-1},w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1},w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

$$\lambda_3 + \lambda_2 + \lambda_1 = 1$$

$$P_{int}(w_i|w_{i-1},w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1},w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

$$L = \sum_h count(h) \log(\sum_{i=1}^{K} \lambda_i P_{ML,i}(h))$$

iv. Absolute Discounting 绝对折扣法

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

递归停止： Unigram 或者 zero-gram

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

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

v. Kneser-Ney Smoothing

词的适配性

LQD NC

WSC ZB WSC LY WSC LPD

(_____, 网红 SLD) 需要选一个交际广泛的人（词）

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

递归停止： Unigram $P(w_i)$

$$p_{KN}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)}$$

w_i的 bigram

总共的 bigram

i. Modified KN Smoothing

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

设置三个D:
D_1 如果 c = 1
D_2 如果 c = 2
D_{3+} 如果 c >= 3

模型	简单理解	只需要记住
+1 平滑	政府印钱	没用
Backoff	用爸爸的钱	
Interpolate	自己和爸爸都出点	Development Set; EM
Absolute Discouting	有钱人缴固定税， 按爸爸的资产分配	Leave-one-out;
Kneser-Ney	有钱人缴固定税， 按爸爸人脉分配	词的适配度
Modified KN	有钱人缴阶梯税， 按爸爸人脉分配	阶梯税率 最好的方法！

对“词”的理解有限
N-gram 上下文的长度有限

只进行统计

4. 概率语言模型

为了解决自然处理中两个常见的分类任务：

将文本按主题归类（比如将所有介绍亚运会的新闻归到体育类）

将词汇表中的字词按意思归类（比如将各种体育运动的名称各归成一类）

分类的关键：计算相关性

A. PLSA

i. LSA(隐性语意分析)

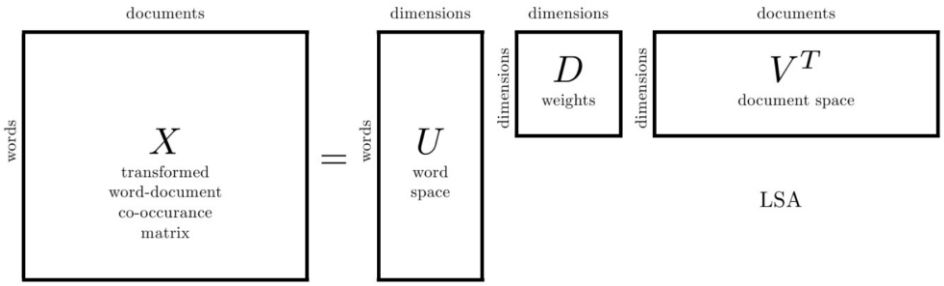
目的：从文本中发现隐含的语义维度-即“Topic”或者“Concept”

解决文档的空间向量模型（VSM）的问题：

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
d_1	1	2		5		7		9		
d_2		3		4		6	8			
d_3	10		11		12			13	14	15

向量空间模型没有能力处理一词多义和一义多词问题,例如**同义词**也分别被表示成独立的一维,计算向量的余弦相似度时会低估用户期望的相似度;而某个词项有**多个词义**时, 始终对应同一维度,因此计算的结果会高估用户期望的相似度。

LSA：



U ：每一行表示意思相关的一类词，其中的每个非零元素表示这类词中每个词的重要性（或者说相关性），数值越大越相关。（矩阵 X 表示词和“语义词”的关系）

V ：每一列表示同一主题一类文章，其中每个元素表示这类文章中每篇文章的相关性。

D ： 表示类词和文章类之间的相关性。

基于 SVD 分解可以构造一个原始向量矩阵的一个低秩逼近矩阵,具体的做法是将词项文档矩阵做 SVD 分解。

$$C = U\Sigma V^T \quad C_k = U\Sigma_k V^T$$

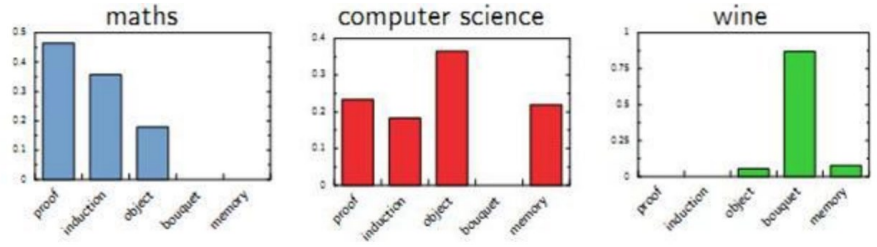
一共有 t 行 d 列, 矩阵的元素为词项的 tf-idf 值。保留Σ的 r 个对角元素的前 k 个保留(最大的 k 个保留),后面最小的 r-k 个奇异值置 0，得到Σ_k;最后计算一个近似的分解矩阵。每个奇异值对应的是每个“语义”维度的权重,将不太重要的权重置为 0,只保留最重要的维度信息,去掉一些信息“nosie”,因而可以得到文档的一种更优表示形式。

只要对关联矩阵 X 进行一次奇异值分解（SVD），我们就可以同时完成了近义词分类和文章的分类。（同时得到每类文章和每类词的相关性）。

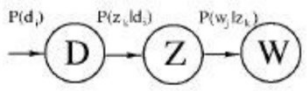
ii. PLSA（基于概率统计的隐性语意分析）

原因：尽管基于 SVD 的 LSA 取得了一定的成功,但是其缺乏严谨的数理统计基础,而且 SVD 分解非常耗时。

Hofmann认为一篇文章（Doc）可以由多个主题（Topic）混合而成，而每个Topic都是词汇上的概率分布，文章中的每个词都是由一个固定的Topic生成的。



PLSA 的概率图模型如下



(1) 以$P(d_i)$的概率选中文档d_i;

(2) 以$P(z_k|d_i)$的概率选中主题z_k;

(3) 以$P(w_j|z_k)$的概率产生一个单词。

其中 D 代表文档,Z 代表隐含类别或者主题,W 为观察到的单词

$$P(d_i, w_j) = P(d_i)P(w_j \mid d_i), \quad P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k)P(z_k \mid d_i).$$

(d_i, w_j)可以直接观察到，z_k为隐含变量。P($w_i|z_k$)， P($z_k|d_i$)分布对应了两组 Multinomial 分布,目标：需要估计这两组分布的参数。

EM 算法估计 PLSA 参数：

Incomplete data 和 complete data：当原始数据的似然函数很复杂时,通过增加一些隐含变量来增强数据,得到 “complete data”,其似然函数更加简单,方便求极大值。通过最大化“complete data”似然函数的期望来最大 化 ″incomplete data″的似然函数,以便得到求似然函数最大值更为简单的计算途径。

EM 算法的步骤是：

(1)E 步骤：求隐含变量 Given 当前估计的参数条件下的后验概率。

(2)M 步骤：最大化 Complete data 对数似然函数的期望，此时我们使用 E 步骤里计算的隐含变量的后验概率，得到新的参数值。

两步迭代进行直到收敛。

PLSA:

目标函数：针对可直接观察到的变量建立似然函数：

$$\begin{aligned} L &= \Pi_m^M \Pi_n^N p(d_m, w_n) \\ &= \Pi_m^{M'} \Pi_n^{N'} p(d_m, w_n)^{n(d_m, w_n)} \end{aligned}$$

其中，M', N' 为文档与词汇数量

$$\begin{aligned} l &= \sum_m^{M'} \sum_n^{N'} n(d_m, w_n) \log p(d_m, w_n) \\ &= \sum_m^{M'} \sum_n^{N'} n(d_m, w_n) \log p(d_m) p(w_n|d_m) \\ &= \sum_m^{M'} \sum_n^{N'} n(d_m, w_n) \log p(d_m) \sum_k^{K} p(z_k|d_m) p(w_n|z_k) \\ &= \sum_m^{M'} \sum_n^{N'} n(d_m, w_n) \log \sum_k^{K} p(d_m) p(z_k|d_m) p(w_n|z_k) \end{aligned}$$

E 步骤：直接使用贝叶斯公式计算隐含变量在当前参数取值 条件下的后验概率

$$P(z_k \mid d_i, w_j) = \frac{P(w_j \mid z_k)P(z_k \mid d_i)}{\sum_{l=1}^{K} P(w_j \mid z_l)P(z_l \mid d_i).}$$

M 步骤：

$$
\begin{aligned}
l &= \sum_{m}^{M'}\sum_{n}^{N'} n(d_m, w_n)\log p(w_n|d_m) + \sum_{m}^{M'}\sum_{n}^{N'} n(d_m, w_n)\log p(d_m) \\
l' &= \sum_{m}^{M'}\sum_{n}^{N'} n(d_m, w_n)\log p(w_n|d_m) \\
E(l') &= \sum_{m}^{M'}\sum_{n}^{N'} n(d_m, w_n)\sum_{k}^{K} p(z_k|d_m, w_n)\log p(w_n, z_k|d_m) \\
&= \sum_{m}^{M'}\sum_{n}^{N'} n(d_m, w_n)\sum_{k}^{K} p(z_k|d_m, w_n)\log p(w_n|z_k)p(z_k|d_m) \\
E(l') &= \sum_{m}^{M'}\sum_{n}^{N'} n(d_m, w_n)\sum_{k}^{K} p(z_k|d_m, w_n)\log p(w_n|z_k)p(z_k|d_m) \\
s.t. &\quad \sum_{k}^{K} p(z_k|d_m) = 1 \\
&\quad \sum_{n}^{N} p(w_n|z_k) = 1
\end{aligned}
$$

拉格朗日乘子法：

$$
f = \sum_{m}^{M'}\sum_{n}^{N'} n(d_m, w_n)\sum_{k}^{K} p(z_k|d_m, w_n)\log p(w_n|z_k)p(z_k|d_m) + \sum_{m}^{M}\lambda_m(1 - \sum_{k}^{K} p(z_k|d_m)) + \sum_{k}^{K}\lambda_k(1 - \sum_{n}^{N} p(w_n|z_k))
$$

$$
\begin{aligned}
\frac{\partial f}{\partial p(z_k|d_m)} &= \frac{\sum_{n}^{N'} n(d_m, w_n)p(z_k|d_m, w_n)}{p(z_k|d_m)} - \lambda_m \\
\frac{\partial f}{\partial p(w_n|z_k)} &= \frac{\sum_{m}^{M'} n(d_m, w_n)p(z_k|d_m, w_n)}{p(w_n|z_k)} - \lambda_k
\end{aligned}
$$

另偏导为 0:

$$
\begin{aligned}
p(z_k|d_m) &= \frac{\sum_{n}^{N'} n(d_m, w_n)p(z_k|d_m, w_n)}{\sum_{k}^{K}\sum_{n}^{N'} n(d_m, w_n)p(z_k|d_m, w_n)} \\
p(w_n|z_k) &= \frac{\sum_{m}^{M'} n(d_m, w_n)p(z_k|d_m, w_n)}{\sum_{n}^{N}\sum_{m}^{M'} n(d_m, w_n)p(z_k|d_m, w_n)}
\end{aligned}
$$

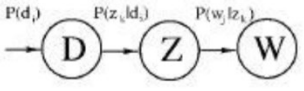
回代到 E 步骤。

iii. 通过训练得到了什么

给定了一个测试样本 d，要判断它是属于那些主题的，我们就需要计算 P(z|d)

PLSA 生成文档的过程：

PLSA 的概率图模型如下



- **1. 按照概率 $p(d_i)$ 选择一篇文档 d_i**
- **2. 根据选择的文档 d_i，从从主题分布中按照概率 $p(\zeta_k \mid d_i)$ 选择一个隐含的主题类别 ζ_k**
- **3. 根据选择的主题 ζ_k，从词分布中按照概率 $p(\omega_j \mid \zeta_k)$ 选择一个词 ω_j**

B. LDA（Latent Dirichlet Allocation）

LDA 在 PLSA 的基础上，为主题分布和词分布分别加了两个 Dirichlet 先验

共轭：

在贝叶斯概率理论中，如果后验概率 P(θ|x)和先验概率 p(θ)满足同样的分布律，那么，先验分布和后验分布被叫做共轭分布，同时，先验分布叫做似然函数的共轭先验分布。

狄利克雷(Dirichlet)分布是多项式分布的共轭分布。

多项分布，是二项分布扩展到多维的情况. 多项分布是指单次试验中的随机变量的取值不再是0-1的，而是有多种离散值可能（1,2,3...,k）.概率密度函数为：

$$P(x_1,x_2,\ldots,x_k;n,p_1,p_2,\ldots,p_k)=\frac{n!}{x_1!\ldots x_k!}{p_1}^{x_1}\ldots{p_k}^{x_k}$$

Dirichlet的概率密度函数为：

$$f(x_1,x_2,\ldots,x_k;\alpha_1,\alpha_2,\ldots,\alpha_k)=\frac{1}{B(\alpha)}\prod_{i=1}^{k}{x_i}^{\alpha^i-1} \quad (4)$$

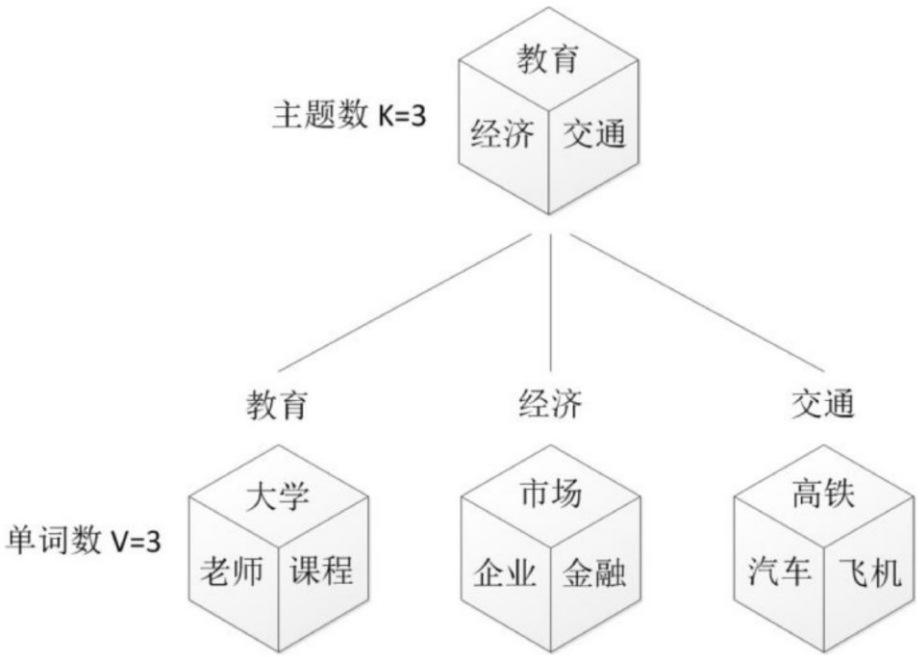
其中，

$$B(\alpha)=\frac{\prod_{i=1}^{k}\Gamma(\alpha^i)}{\Gamma(\sum_{i=1}^{k}\alpha^i)},\sum_{i=1}^{k}x^i=1 \quad (5)$$

共轭的意思是，数据符合多项分布的时候，参数的先验分布和后验分布都能保持狄利克雷分布的形式，这种形式不变的好处是，我们能够在先验分布中赋予参数很明确的物理意义，这个物理意义可以延续到后续分布中进行解释，同时从先验变换到后验过程中从数据中补充的知识也容易有物理解释。

LDA 生成文档的过程：

- 1. 按照先验概率 $p(d_i)$ 选择一篇文档 d_i
- 2. 从Dirichlet分布 α 中取样生成文档 d_i 的主题分布 θ_i ，主题分布 θ_i 由超参数为 α 的Dirichlet分布生成
- 3. 从主题的多项式分布 θ_i 中取样生成文档 d_i 第 j 个词的主题 $z_{i,j}$
- 4. 从Dirichlet分布 β 中取样生成主题 $z_{i,j}$ 对应的词语分布 $\phi_{z_{i,j}}$ ，词语分布 $\phi_{z_{i,j}}$ 由参数为 β 的Dirichlet分布生成
- 5. 从词语的多项式分布 $\phi_{z_{i,j}}$ 中采样最终生成词语 $\omega_{i,j}$



上图中有三个主题，在 PLSA 中，我们会以固定的概率来抽取一个主题词，比如 0.5 的概率抽取教育这个主题词，然后根据抽取出来的主题词，找其对应的词分布，再根据词分布，抽取一个词汇。由此，可以看出 PLSA 中，主题分布和词分布都是唯一确定的。但是，在 LDA 中，主题分布和词分布是不确定的，LDA 的作者们采用的是贝叶斯派的思想，认为它们应该服从一个分布，主题分布和词分布都是多项式分布，因为多项式分布和狄利克雷分布是共轭结构，在 LDA 中主题分布和词分布使用了 Dirichlet 分布作为它们的共轭先验分布。所以，也就有了一句广为流传的话 -- LDA 就是 PLSA 的贝叶斯化版本。

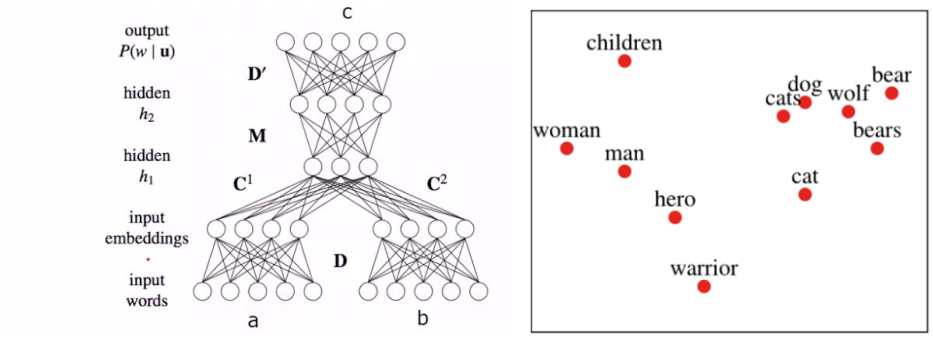
$\vec{\alpha} \to \vec{\theta}_m \to \zeta_{m,n}$ ，这个过程表示在生成第m篇文档的时候，先从抽取了一个doc-topic骰子 $\vec{\theta}_m$ ，然后投掷这个骰子生成了文档中第n个词的topic编号 $\zeta_{m,n}$ ；

$\vec{\beta} \to \vec{\phi}_k \to \omega_{m,n} \mid = \zeta_{m,n}$ ，这个过程表示，从K个topic-word骰子 $\vec{\phi}_k$ 中，挑选编号为 $k = \zeta_{m,n}$ 的骰子进行投掷，然后生成词汇 $\omega_{m,n}$ ；

5. 神经语言模型

A. 神经概率语言模型（NNLM）

“a b c”



（trigram model）

通过把 word 的 one－hot 编码转换为词嵌入（embedding），增加对词的理解（包括词法，语法和语义）

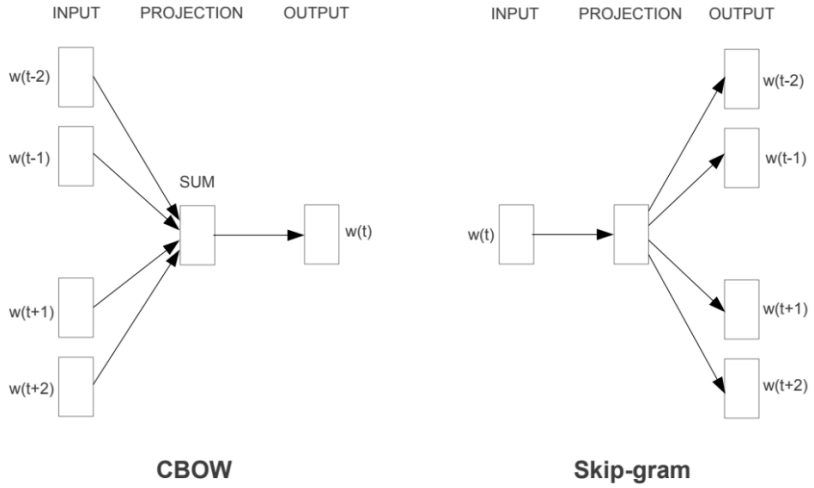
特征学习 representation learning

神经网络中的线性变换和激活函数相当于提取了 word 不同潜在特征

https://blog.csdn.net/u012328159/article/details/72847297?locationNum=8&fps=1

B. CBOW 和 Skip-gram 模型

skip-gram 简化了 NNLM，只包含线性模型，快＋大数据



寻找近义词，作为其他 NLP 任务的特征

https://blog.csdn.net/qq_31456593/article/details/77542071