自然语言处理课程笔记

lesson1

sub-language（子语言的定义）

- 子语言表示的是某一个领域的语言。比如一个新闻评论的数据集就可以构成一个子语言，在这个子语言中可以统计其一些性质。
- **Zipf's Law**：词出现的频率和它的rank是成反比的。（一个简单的经验规律）
- 表层意义与隐藏意义：表层意义是指一句话的书写形式，隐藏意义是指指代分辨，词义去模糊等隐藏含义。

  一些NLP任务是给定一个表层义，求取他的隐藏义，这也是一个贝叶斯分类器的过程。

  $$H^* = \arg\max_H P(H|S) = \arg\max_H \frac{P(H,S)}{P(S)} = \arg\max_H P(H,S)$$

  其中H是隐藏义任务，S是显式文本。
- Source Channel Paradigm，就是一个推导过程。

  给定一个Y，经过一个channel，变成X，我们的目标是找到这样的Y，使得在给定X的条件下P(Y|X)最大。**就是给定X的情况下最有可能出现的Y.**

  $$Y^* = \arg\max_Y P(Y|X) = \arg\max_Y \frac{P(Y)P(X|Y)}{P(X)} = \arg\max_Y P(Y)P(X|Y)$$

  这种方法叫做贝叶斯分类器（Bayes classifier）

lesson2

词向量概述

1. 用向量表示一个词，有几种方法，最直接的是localist方法，就是使用one-hot向量表示一个词
2. 词向量的几种不同表示：
   - distributional representation：意思是**分布式表示**（基于分布式假设，如果词语的上下文相似，那么这个词的意思也是相似的），直观理解就是考虑词语的上下文的表示方法就是分布式表示。

     比如说共现矩阵的表示方法就是分布式表示
   - distributed representation：意思是**分散式表示**，表示的是将一个文本或者一个单词的意思分散到低维空间的不同维度上。这样的表示法最后可以获得一个稠密、低维、连续的向量。

     比如说word2vec，就是分散式的表示方法，其实也是分布式，训练方法基于上下文。

     对于分散式表示，可以理解为两种数据类型的多对多表示，在具体表示过程中，又有着如下的表示方法。
     - 二元编码表示，比如一个个性质都可以用二进制的形式表示
     - 特征提取，使用连续向量提取到其中观点特征。

    - tradeoff：高维向量占用空间，并且会形成稀疏；低维向量的表示能力有限。所以词向量的维度大小其实也是一个经验值。

word2vec

- 该表示方法实际上是给one-hot向量x乘上了一个矩阵E，比如$x \in R^V$,矩阵$E \in R^{V \cdot D}$，最后乘出来的是一个D维向量。对于这个E有几种理解，首先他相当于是一个查找表，look-up-table，查找第i行就是第i个词的词向量。其次E实际上等价于给网络加了一个线性层。
- 具体的训练方法。
    - skip-gram：知道中间词，预测周围固定大小窗口的词;CBOW:知道两边的词，去预测中心词。**实际上skip-gram是，我们希望调整中间词的词向量，使得其周围的词出现的条件概率尽可能大**，我们的目标公式化如下

$$L(\Theta) = \Pi_{t=1}^{T} \Pi_{-m \leq j \leq m} P(w_{t+j}|w_t, \Theta)$$

    - 我们构建损失函数。$J(\Theta) = -\frac{1}{T}\log\left(L(\Theta)\right)$
    - 为了完善上述的定义，我们使用softmax函数去得到这样的条件概率

$$P(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w \in V} \exp\left(u_w^T v_c\right)}$$

      取指数的目的只是为了让上下都是大于0的，并且softmax的最好的地方在于方便求导。

      在这里面u，v是两个相同的向量，只不过u表示的是该词作为语境词的词向量，v表示作为中心词的词向量。构建两个向量主要是方便查询。

    - 经过训练之后获得这两个词向量去平均得到最后的词向量。
    - 流程如下

$$x \quad -> xE \quad -> \text{上下文预测任务} \quad -> \text{修改} E$$

- **word2vec存在的问题**，对于上述的损失函数，每次对于某一个向量的改变，都会需要遍历词汇表中的所有向量。这一缺陷是因为分母中是对于所有词向量进行计算导致的。
- **解决方法：负采样技术**
    - 我们对于每一个$P(o|c)$ 我们考虑使用sigmoid函数，只考虑$u_o^T v_c$ 和一些通过一些分布筛选出来的负样本，得到改进后的损失函数如下

$$J(\Theta) = \frac{1}{T}\sum_{t=1}^{T} J_t(\Theta)$$

    - $$J_t(\Theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^{k} \mathbf{E}_{j \sim P(w)}[\log \sigma(-u_j^T v_c)]$$

    - 对于上述公式，除以T是将损失函数平均化，与句子长度无关，对于每个单词的损失函数，点积表示相似度，**第一项表示中心词的词向量给定的情况下，指定位置的单词时o的概率，我们希望这个概率尽可能大，同时希望指定位置的单词不为o的概率尽可能小，也就是第二项。**（其实$\log \sigma(-u_j^T v_c)$ 相当于是1-P，1-P尽可能大，则负样本出现的概率尽可能小）。
    - 负采样技术中，选用的分布是，取$\frac{3}{4}$是希望让出现频率较少的词能被更多的选取，相当于拉平了这个分布。

$$P(w) \propto U(w)^{\frac{3}{4}}$$

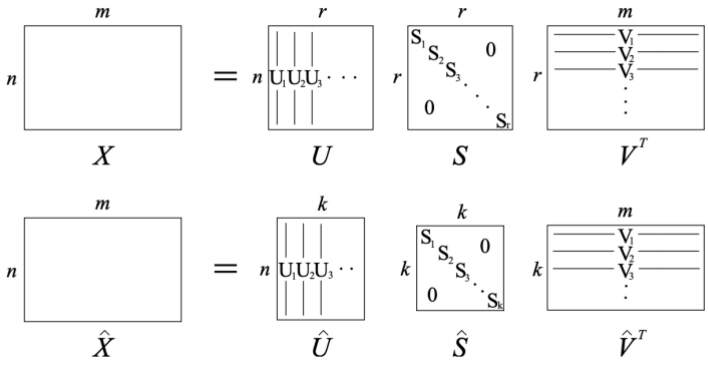
共现矩阵表示法（count-base基于计数的表示法）

- 类似于主成分分析，首先根据词库和特定的window大小构建一个共现矩阵，然后做奇异值分解，最后取得其中前k大的奇异值将原始矩阵近似，取U矩阵中的表示法即可

$$
\begin{gathered}
\text{改变前} \\
M \in R^{n \times m} \\
U \in R^{n \times r} \\
\Sigma \in R^{r \times r} \\
V \in R^{m \times r} \\
M = U \Sigma V^T \\
\text{改变后} \\
\widetilde{M} \in R^{n \times m} \\
\widetilde{U} \in R^{n \times k} \\
\widetilde{\Sigma} \in R^{k \times k} \\
\widetilde{V} \in R^{m \times k} \\
\widetilde{M} = \widetilde{U} \widetilde{\Sigma} \widetilde{V}^T
\end{gathered}
$$

- 共现矩阵的例子
  - Window length 1 (more common: 5-10)
  - Symmetric (irrelevant whether left or right context)
  - Ecample corpus:
    - I like deep learning
    - I like NLP
    - I enjoy flying

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0



选取前k大的奇异值进行矩阵的近似。

- 使用基于计数的方法的缺点
  - 有一些功能词出现了太多次，也被统计了太多次。这些功能词会对整个词库造成影响
- 一些解决措施
  - 我们对于整个的频率分布对数化(相当于也是对整体概率分布进行拉平)，并且忽视功能词(the/he/has等)
  - 使用一些相关系数代替词的计数。

word2vec vs count-base

- count-base：
  - 优点：迅速的计算，时间开销不大；很好地利用了语料库的统计信息。
  - 缺点：事实上并没有学习到词的特征，会存在信息的损失。
- word2vec：（低维向量的连续表示）
  - 优点：可以提取更深层次的信息。
  - 缺点：计算复杂，并且复杂度随着文本的长度增加，随着词汇量的大小增加

Glove embeddings

- 不同于共现矩阵方法的计算word-window中词语出现的次数。glove计算的是条件概率和条件概率的商。比如

$$P(solid|ice) = co-occur(solid, ice)/occur(ice)$$

- 最后得到的条件概率表格如下

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

  在这个表格中，差距越大的词k，越能够区分ice和steam的性质。

- 我们希望定义一个对数回归模型，这个模型的目标是找到这样的词向量，使得

$$\log P(i|j) = w_i^T w_j$$

  其中$P(i|j) = \frac{X_{i,j}}{X_j}$,则有

$$w_i^T w_j = \log(X_{i,j}) - \log(X_i)$$

  对于等式右边的第二项，不涉及到两个词之间的关系，可以直接使用偏置量代替bias，最后得到的损失函数如下。

$$J = \sum_{i,j=1}^{V} f(X_{i,j})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{i,j}))^2$$

- glove方法摆脱了只是基于计数的手段，可以利用统计信息去得到一个低维的向量表示，事实上综合了前面两种方法的特点。
- glove方法在某一些情况下表现要优于word2vec，也是现在用到比较多的词向量表示。

词向量的验证方法

- 内在验证：使用一些和词向量直接相关的任务，比如词义消解，聚类分析（将相似的词聚类）
- 外部验证：使用下游任务进行验证，事实上只能验证其性能，并不能从其结构的本质上说明问题。并且需要花费较多的训练时间。

lesson3

unigram estimation平均估计

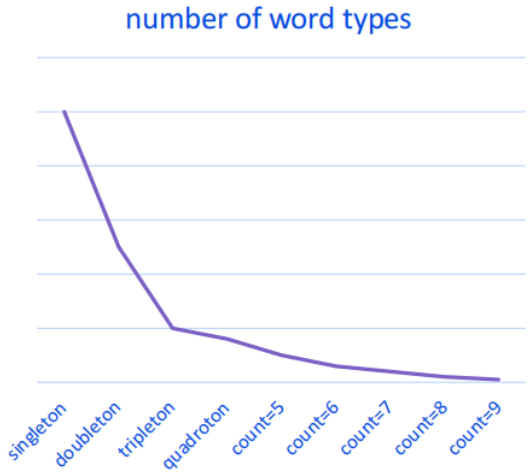
1. 对于一个词，我们希望估计这个词的频率，或者说出现的概率。对于一个语料库和一个词汇表，我们定义如下情况。

$$\text{语料库：}x_1, x_2, \ldots, x_N \quad \text{词汇表：}w_1, w_2, \ldots, w_V$$

$$\text{一个词的概率为：}\hat{p}_{ML}(w_j) = \frac{c(w_j)}{N}$$

   其中这个c函数是指这个词的统计次数。

2. **n-gram**：指的是连续n个词组成的一串，可以以n-gram为单位统计词频。
3. 两个**challenge**：
   - 未出现的单词：在语料库里面没有出现的单词，并不应该认为他的单词概率就是0，只是他没在这里出现而已。
   - 词频极低的单词：虽然他们的概率并不为0，但是会存在较高的方差，比如你统计的是1/1000，但是其实应该是3/1000，相对的概率变化就相差了很多。（一般情况下，**因为语料库一般不够大，低频词会被过高的估计。**）
4. 解决方法（初步）
   - unknown label的使用，比如我们构建这样的一个词汇表，这个词汇表是语料库中至少出现两次的单词构成的表。然后其他的全部标记为unknown，然后统计词频。
   - 缺点：
     - 只是简单地替换，并没有考虑不同的unknown word之间的差别。
     - 需要大语料库，因为你不要那些低频词
     - 对于命名实体识别，QA任务不利，因为有些专有名词或者主语只会出现一次，而在QA中主语是重要的。
     - 没有改进低频词的统计。
5. **古德图灵估计（Good-Turing Estimation）**
   - Good-Turing Estimation用于优化NLP中对于低频词和未出现的词的概率统计。
   - 预定义：我们首先统计词频的频率（比如，只出现一次的词的总数量，word-type）



     y轴是词的总数量。

     对于这幅图，我们有如下的估计
     - 长尾分布，有可能在非常远的地方，一直是0，然后突然出现一个1.比如单词the出现了十万次。
     - 整体上是递减规律，出现一次的词要比较多。

     我们是希望改进最左边这些低频词的估计。
   - 性质:这幅图有如下性质

$$\sum_{n=0}^{\infty} c_n = V \quad \sum_{n=0}^{\infty} nc_n = N$$

我们考虑下一个词是singleton的概率，显然是$\frac{c_1}{N}$，但这个估计是错的，**事实上实验中会低很多。出现这一问题的原因分析如下**：

- 只是使用了一个数据集去估计所有数据集上面的分布，我们这个估计是**依赖于数据集**的
- 所以产生偏差的原因不是因为最大似然估计MLE的问题，他还是无偏估计

- **古德图灵估计方法**
  - 两个假设和推断
    - 预测下一个词的是否是singleton，和你观察的前面的单词数有关，但是你**观察的前面的单词数相差不多的时候，可以近似认为条件概率是相等的。**

      P(next word is a singleton | N words observed) ≈ *P*(next word is a singleton | N − 1 words observed)
    - 如果我这时候看到的词是singleton，那么包括他在内的的前面的总单词中，他一共出现了两次。因为在N-1的部分出现了一次。
  - 产生公式

$$\begin{aligned} &P(\text{next word is a singleton}|N-1 \text{ words observed}) \\ =&P(\text{a word occurred twice}|N \text{ words observed}) \\ \approx&\frac{2c_2}{N} \end{aligned}$$

2c2是出现两次的词的总数，一共有c2种词，每个词出现两次。

  - 递推得到

$$\hat{P}_{GT}(next\ word\ has\ occurred\ n\ times) = \frac{(n+1)c_{n+1}}{N}$$

  - 因此，我们通过这个得到了**没有出现的单词应该被赋予的概率**
  - 对于图灵估计方法，可以看做是乘上了一个图灵因子，对于每一个单词w_j，计算他的概率，等价于除以c_n,因为一共有c_n种,**这个w_j一共出现了n次。**

$$\hat{P}_{GT}(w_j) = \frac{(n+1)c_{n+1}}{Nc_n} = \frac{(n+1)c_{n+1}}{nc_n}\hat{P}_{ML}(w_j) = \frac{G_n}{c_n}$$

- 图灵估计方法适用于：大量类别的多项式分布；长尾分布，小计数样本占主导的分布。
- 图灵估计有时候会给G_n分母底下减掉n，是对其的进一步调整。

6. **Dirchlet smoothing**平滑处理

对于这个词频统计与单词概率计算的问题，我们可以认为是一个多类别的多项分布，可以使用贝叶斯估计的方法。参数μ是对应的词的概率，给定μ求x显然是好求的。

$$p(x|\mu) = \frac{N!}{x_1!\dots x_k!}\mu_1^{x_1}\cdots\mu_k^{x_k}$$

- 我们一般认为贝叶斯估计的先验概率，即参数的分布，由狄利克雷分布给出。在这里，参数μ的分布由α决定。

$$p(\mu|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\dots\Gamma(\alpha_k)}\mu_1^{\alpha_1-1}\cdots\mu_k^{\alpha_k-1} \quad \text{where} \quad \alpha_0 = \sum_{i=1}^{k}\alpha_i$$

- 最后得到后验概率，希望是已知结果求原因，x是结果，概率μ是产生x的原因。

$$p(\mu|x,\alpha)=\frac{p(x|\mu)p(\mu|\alpha)}{p(x|\alpha)}\propto p(x|\mu)p(\mu|\alpha)\propto \mu_1^{x_1+\alpha_1-1}\dots\mu_k^{x_k+\alpha_k-1}$$

- 这个实际上还是一个beta分布，可以求他的MLE

$$\hat{\mu}^{\mathrm{ML}}=\arg\max_{\mu} p(x|\mu)\Rightarrow \hat{\mu}_i^{ML}=\frac{x_i}{\sum_{i'=1}^{k}x_{i'}}$$

$$\hat{\mu}^{\mathrm{MAP}}=\arg\max_{\mu} p(\mu|x,\alpha)\Rightarrow \hat{\mu}_i^{\mathrm{MAP}}=\frac{x_i+\alpha_i-1}{\sum_{i'=1}^{k}(x_{i'}+\alpha_{i'}-1)}$$

- 经过后验概率的计算可以算出平滑处理后的概率，相当于把每一中单词都加上或者减去了一定的值，调整了概率。当α全部为2的时候就是laplace平滑处理，**把所有的单词的出现次数都加上1，这样就没有没出现过的单词了。**
- 我们也可以对于概率进行操作。把$\frac{c(w_j)+1}{N+V}$作为概率，然后单词出现次数等于概率乘上总数即可。**相比直接对于出现次数加1对于低频词的出现次数增加更多**
- 存在的问题：**先验不一定准确**；

Language modeling语言模型

1. n-gram语言模型
   - 假设下一个词只依赖于前面的n-1个词，会损失长距离依赖关系
   - 注意，每一句话都用<s>.....<\s>这样的形式构建。
     - 使用start-of-sentence symbol，是因为这样使得在数学上表示会比较一致（都是条件概率的形式）；同时会给网络提示，这里应该产生token了
     - 使用 end-of-sentence symbol，是因为如果不使用句子结束符，那么语言模型只能对于固定长度的序列进行概率分布的建模。比如如下情况。

       考虑一个简单却直观的情形，假设词表 $V=\{a,\ b\}$，训练集给定如下

       > 共四个序列，每个序列的长度为2(不包括起始符)
       > $W_1:\ <s>\ a\ b$
       > $W_2:\ <s>\ b\ b$
       > $W_3:\ <s>\ b\ a$
       > $W_4:\ <s>\ a\ a$
       > 注：该例取自Speech and Language Processing 3rd ed

       不难发现，上述四个序列已经囊括了词表 V 下长度为2的所有可能序列。

       不考虑结束符，先利用上述极大似然估计的第一个等式可直接计算出

       $$P(a|<s>)=1/2,\ P(b|<s>)=1/2$$

       $$P(a|a)=1/2,\ P(a|b)=1/2,\ P(b|a)=1/2,\ P(b|b)=1/2$$

       然后，基于 bigram 语言模型，惊奇的发现

       $$P(W_1)+P(W_2)+P(W_3)+P(W_4)=1$$

       只有加了结束符，整个状态空间才完整。

2. 语言模型的challenge

- Simple conditional distributions *P(X = x | Y = y)*
  - Bigram model
  - Case 1: y has been seen, but (x, y) was not seen. Good-Turing and Dirichlet are applicable.
  - Case 2: y has not been seen
- Complex conditional distributions *P(X = x | Y = y, Z = z)*
  - Trigram model
  - Case 1: (y, z) has been seen, but (x, y, z) was not seen. Good-Turing and Dirichlet are applicable.
  - Case 2: the context (y, z) has not been seen

    - 当xy的组合没有出现时，case1，可以用前面的两种方法改进。
    - 当y或者是yz的组合没有在语料库中出现的时候，**case2，分母会是0**，最后得到的概率没有定义。这点有待解决

3. 解决办法

    - **线性内插法：**我们可以用加权平均的方法，一个很自然的想法就是，**如果前面的n-1个词构成的chunk没有出现，我们就看n-2个词构成的chunk，依次类推地分配权重。**

$$\begin{aligned}&\tilde{P}(x_i|x_{i-1},x_{i-2})\\=&\lambda_3\hat{P}(x_i|x_{i-1},x_{i-2})+\lambda_2\hat{P}(x_i|x_{i-1})+\lambda_1\hat{P}(x_i)\\&\text{for}\quad\lambda_1+\lambda_2+\lambda_3=1\end{aligned}$$

    一个不足之处是这个λ还是要训练或者人为确定。

    - **绝对折扣法：Absolute Discounting**：我们构建如下的公式

$$P_{AD}(x_i|x_{i-1},x_{i-2})=\frac{max(c(x_{i-2}x_{i-1}x_i)-D),0)}{c(x_{i-2}x_{i-2})}+(1-\lambda)P_{AD}(x_i|x_{i-1})$$

    分析，如果分子里面count不等于0，那么等式右边的左端项就非零，同时右端项贡献出bigram的影响；如果等于0，就完全看bigram。相比之下，这个λ可以直接求取。

    我们假设有S个x_i出现在后面，对上式求和。左端是1，但是右端我需要知道$P_{AD}(x_i|x_{i-1})$的取值，右端的左端项求和等于1-$\frac{SD}{c(x_{i-2}x_{i-2})}$可以求出λ。

$$1-\lambda=P(U)=\frac{SD}{c(x_{i-2}x_{i-1})}$$

    - **Kneser-Ney (KN) Smoothing**

    一个思维是一个词出现的频率或者次数不应该是有他的绝对频率决定，而是应该由他上下文的种类决定。

$$P_{\text{KN}}(x)=\frac{|\{v:c(vx)>0\}|}{\sum_{x'}|\{v:c(vx')>0\}|}$$

Number of contexts in which x appears

Number of word types v which precede x

Normalization

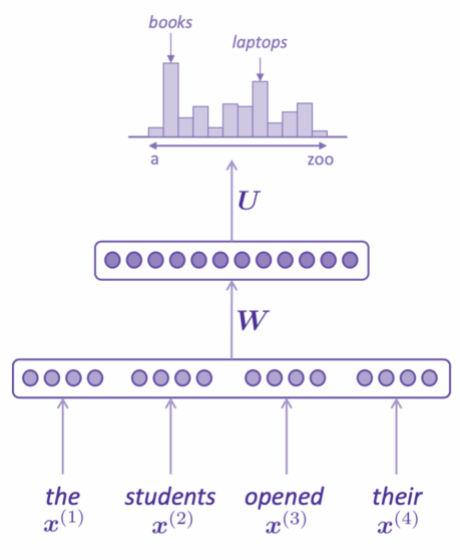
    Kneser-Ney (KN) Smoothing仍然是使用Absolute Discounting，但是在计算单个词的概率的时候使用了$P_{KN}(x)$

lesson4

神经网络语言模型

对于统计的语言模型，**稀疏性是其本质的问题，无法解决。**我们希望通过神经网络的方法，更加连续地表示原先的目标（条件概率）。

- fix-window model



比如我给定一个上文长度，和n-gram的规则是一样的。

首先把词通过词向量矩阵等方法表示为词向量，然后concat，通过乘以W，变成隐藏层向量h，隐藏层向量乘上U再softmax得到概率分布。我们甚至可以认为，U是一个词的表示矩阵。

- fix-window的问题
  - 首先无法关注到长距离信息，即使我们增大window-size，但是也不能够无限大。并且window-size的增大会增大参数量
  - 对于不同位置的处理，比如$x^{(1)}, x^{(2)}$,他的处理不是一致的，从感性上是不合理的。**No symmetry**

RNN模型

事实上已经学习过一遍了，这里仅记录一些新知识

一些分析

- 每一个隐藏层经过矩阵乘法后会作用上一个激活函数，其实并不是为了映射到概率，只是单纯增加一个非线性。
- 对于一个RNN，其实就像他名字说的，就是一个**隐藏层使用循环反复作用**。把它展开形成一个序列化结构。所以在这个序列化结构中，嵌入矩阵和隐藏层矩阵W_e W_h都只有一个。在梯度传递的过程中，要把所有时间步的梯度加起来一起去改变这几个矩阵。（**这样会不会参数量太少了？在空间上并不够深**）

优点

- 相比之前已经可以处理任意处理长度的文本输入，并且模型参数量保持恒定，不随输入长度增加而增加。
- 可以解决长距离依赖问题（理论上）
- 来自上文的特征提取信息可以被重复计算，这是因为$x_1, x_2, x_3, x_4 \rightarrow x_2, x_3, x_4, x_5$对于n-gram需要重新计算，但是RNN都保留到上一个时间步里面了

缺点

- 难以并行，因为有时间步。
- 梯度消失难解决

梯度消失再分析

- 问题定义

    - 一个模型主要有以下三个方面的问题：Capacity（能否准确表达目标函数）、Optimization（能否实现目标函数的优化（在训练集上的优化））、Generalization（能否实现模型在测试集上的泛化）
- 对于梯度消失，他其实是一个优化问题，因为RNN给足够的参数他是有能力去拟合目标的，但是因为这个issue，使得不可能得到优化的（在具体操作上）

各种batch构建方法

因为这个RNN语言模型直接输入的是一个语料库，我们希望将这个语料库变成batch从而实现较好的训练。

Sentence-level batching（句子级别的批次构建）

- 必须是一句话一句话地构建，需要去识别句子结束符，为了变得等长，需要给定一个最大长度，然后把所有的句子都padding到那个长度，比如可以添加</s>。这样需要有mask向量，这个mask向量保证损失函数不计算padding部分。

this	is	an	example	</s>
this	is	another	</s>	**</s>**

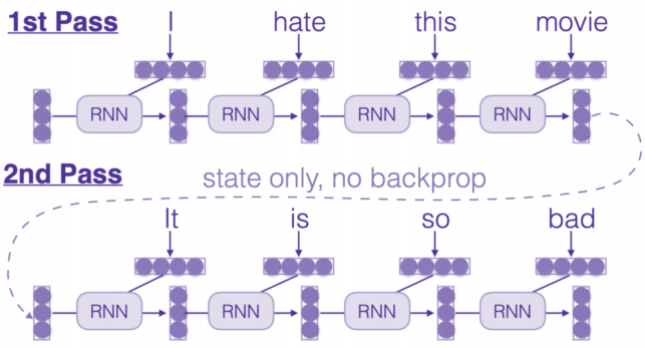
- 一些小trick：我们可以首先对于句子长度进行聚类，把长度比较一致的放到一起，然后合理地去调batch size。

Fixed-Length Batching（固定长度的批次构建）

- 可以不用padding和mask，操作简便
- 缺点是缺乏对于语言边界的建模，（这样产生的模型最后可能不知道一句话应该怎么结尾？结尾不畅？）

Continuous Batching（连续的批次构建）

- 输入的时候不改变语言的顺序（不进行shuffle），使用上一个批次的最后一个时间步的隐藏层状态对于下一个批次的第一个时间步初始化。



Shuffled Batching（随机打乱的批次构建）

- 批次间的句子没有上下文关系，已经被打断了。
- 缺点在于无法构建批次之间的关系。

LSTM模型

Input to cell: $x^{(t)}, h^{(t-1)}, c^{(t-1)}$, output of cell: $h^{(t)}, c^{(t)}$

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)} + \mathbf{b}_f)$$
$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{b}_i)$$
$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_o \mathbf{h}^{(t-1)} + \mathbf{U}_o \mathbf{x}^{(t)} + \mathbf{b}_o)$$

Three gates

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$
$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(i)} \odot \tilde{\mathbf{c}}^{(t)}$$
$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh \mathbf{c}^{(t)}$$

Gating operations for updating cell and hidden states

构成：由长时记忆和短时记忆构成，可以认为cell-state中记忆的是长时记忆。每一个时间步都会对于长时记忆进行更新。正是因为LSTM保存了长时记忆，所以即使他有梯度消失的问题，也不会影响他对于长距离的关注。

RNN的应用

- Represent a sentence（表示一整句话，模型需要理解这一整句话的意思）
- Represent a context within a sentence（比如表示一个单词的上下文，模型需要理解这个单词在上下文的含义和词性）

Lecture5

文本分类

NLP中三种文本分类的模式

- **feature vector+feedforward nets**, the key point is how to build the feature vector. Method: BOW(bag of word) , hand crafted vector.
- **word vector + FC**: use embedding and FC layers
- **word vector + RNN/CNN**: RNN can extract the sequential information, CNN can extract the **n-gram information** by using the **different size of filters**.

四种分类方法具体实现

1. BOW

   build a vocabulary (word to index and index to word), construct the sentence vector by calculate the occurrences of word in the sentence. Add 1 to the corresponding position.

   Feed in to the FC

2. FastText

   count the n-gram information, use hash function the expand the sentence vector dimension. Each the sentence vector element represent the word index in the vocabulary.

   $$Sen_{dim} = Vocab_{size} + Ngram - hash - tabel_{size}$$

   Use embed layer to project the sentence vector to embedding, and make average of all the word embedding, finally feed into a FC.

3. CNN

Use embed layer to project the sentence vector to embedding.

use 2D filters to extract n-gram feature.

4. RNN

   Use embed layer to project the sentence vector to embedding.

   extract the sequential information

5. **Character-Level CNNs**

   based on character vocabulary. will make a long sequence.

Lecture6

Machine translation

1. statistical

   use bayes rule. suppose French x to English y. because given y to predict x is easier, we can use translation model and language model to form a French2English translation machine.

$$\arg\max_y P(y|x) = \arg\max_y P(x|y)P(y)$$

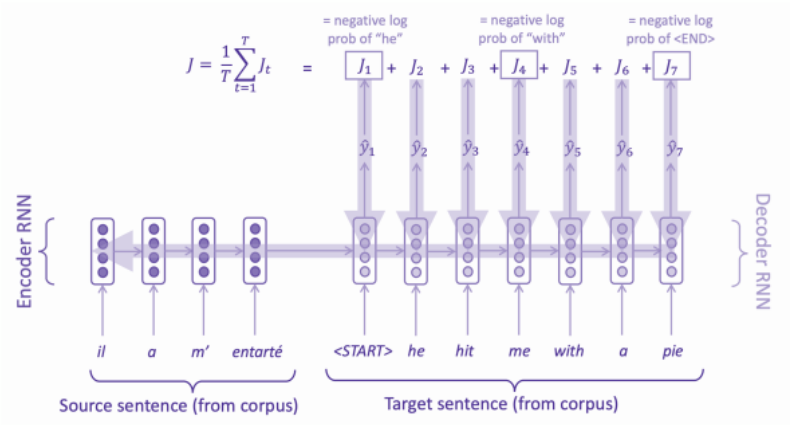
2. Alignment

   Alignment is the correspondence of words between sentence pair.

   - Problem: It is difficult to correctly find the Alignment, because of its complex relationship(many to many , many to one , one to many)
   - Solution: use phrase-Base, the correspondence of phrase is less complex.

NMT and Seq2seq

1. seq2seq structure



Seq2seq is optimized as a single system. Backpropagation operates "end-to-end".

- **Structure detail**: encoder encode the source sentence and transfer to the last hidden state. decoder use the <start> token and the former information to generate, until get the <\s> token.
- **Drawback**: less interpretable and difficult to control (not rule base)

2. use **back translation** to solve the lack of data problem

   - **Definition**: at first, we have some target language, we build a intermediate system to translate target language to source language, and **use these sentence pairs to train our desired model**.

- **Drawback:** the back translation usually use beam search which is an argmax method, this can not represent the truly distribution of data.
- **Solution:** use **probability sampling** (do not always generate word with the largest probability) and **noise** in the output sentence (that is randomly delete the word in the source sentence that intermediate system generate.)

3. **ensemble model**
    - **Definition**: To train multiple model at the same time, the output is the average of the model list.
    - **Drawback:** in the interfere mode, it will use a lot of memory
    - **Solution: parameter averaging** ,this will only work when the model is train simultaneously

Eavluation

1. BLEU

   Definition: calculate the generate sentence and the manual sentence similarity (the word overlap)

2. model base method

   Definition: use model embedding to calculate the similarity.

   Drawback: need the teacher model is well-performed than the evaluate model , lead to a boundary of translation performance.

subword representation

BPE algorithm

Definition: start from character vocabulary, add new pair (A combination of existing elements in the vocabulary) with max frequency. until get the specific size.