**主观题自动评分**

**问题描述：**

答到关键点得分，关键词匹配

由于中文语义多样性，关键点可能存在多种描述，需要对语义进行正确识别

分析语句结构

对常见的解释性词语及语句及其解释之间的匹配

0-10的评分机制

**技术清单：**

1. NLP
2. 文本相似度计算
3. 词嵌入
4. 分词
5. 命名实体
6. 可视化展示界面设计

方案一：网站（Flask/Django后端，前端套模板）

方案二：跨平台GUI框架（Electron PyQT）

**二．文本相似度计算**

**1.传统的计算模型，**不借助神经网络，利用传统的统计词频，和相似度计算公式实现

**TF-IDF：**用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度，总结就是一个词语在一篇文章中出现次数越多，同时在所有文档中出现次数越少，越能够代表该文章。

TF=某一类中词条出现的次数/该类中所有的词条数目

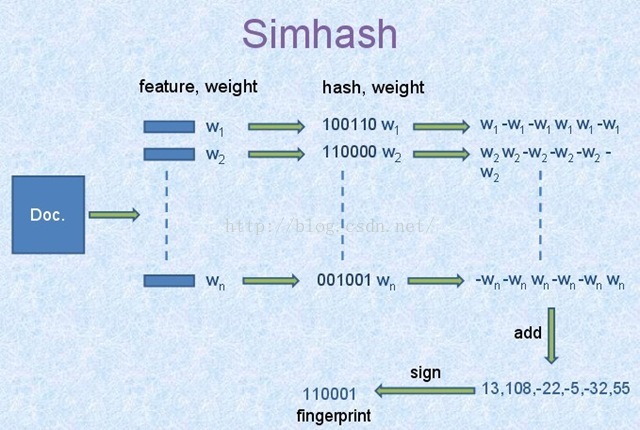
TF指的是某一个给定的词语在该文件中出现的次数，一般会被归一化（词频/文章总词数）

IDF的主要思想是包含词条t的文档越少，IDF越大，则说明词条具有很好的类别区分力。

IDF=log（总文件数/包含该词条的文件数+1）

TF-IDF=TF\*IDF，因此，TF-IDF倾向于过滤掉常见的词语保留重要的词语。

**Simhash：**（只能检测出文本一致或者不一致，停留在文字表面？？）



feature指一篇文档分词后的某个词，weight是这个词的权重，可以是这个词在句子中出现的次数，hash即传统的hash算法

通过simhash算法可以计算出文档的simhash值，通过各个文档计算出的二进制值来计算文档间的汉明距离，然后根据这个距离来比较文档之间的相似度。汉明距离指的是两个相同长度的字符串相同位置上不同的字符的个数

Simhash算法分为五个步骤：

分词：给定一段语句，进行分词，得到有效的特征向量，为每一个特征向量设置1-5 五个级别的权重

Hash：通过hash函数计算各个特征向量的hash值，hash值由二进制数01组成，这样就将字符变成了一系列数字

加权：在hash值的基础上，给所有特征向量进行加权，即W=Hash\*weight，且遇到1则hash值和权值正相乘，遇到0则hash值和权值负相乘。例如给“CSDN”的hash值“100101”，加权得到W（CSDN）=100101\*4=4 -4 -4 4 -4 4，“博客”的hash值“101011：加权得到W（博客）=101011\*5=5 -5 5 -5 5 5

合并：将上述各个特征向量的加权结果累加，变成只有一个序列串.例如将前两个特征向量累加，得到“9 -9 1 -1 1 9“

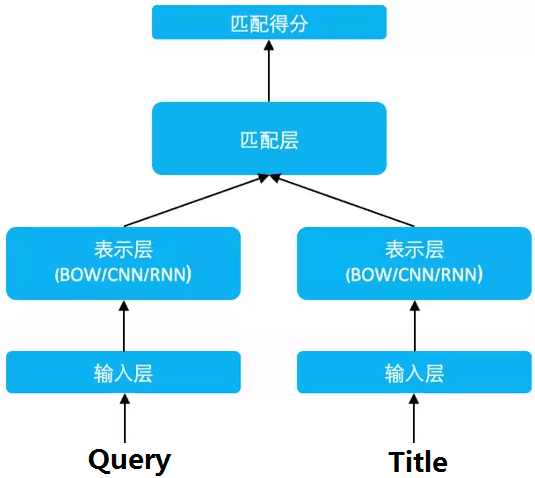
降维：对于累加结果，如果大于0则置1，否则置0，从而得到该语句的simhash值，最后我们便可以根据不同语句simhash的汉明距离来判断他们的相似度。

缺点：传统的文本相似性无法发现识别语义，对相似度的计算只停留在文字表面。在排序时，一些细微的语言变化往往带来巨大的变化。使用他们进行语义匹配，效果不是很好。

**2.基于神经网络的计算模型：**

**DSSM：**用神经网络把Query和Title表示为低维语义向量，并通过cosine距离来计算两个语义向量的距离，最终训练出语义相似度模型。

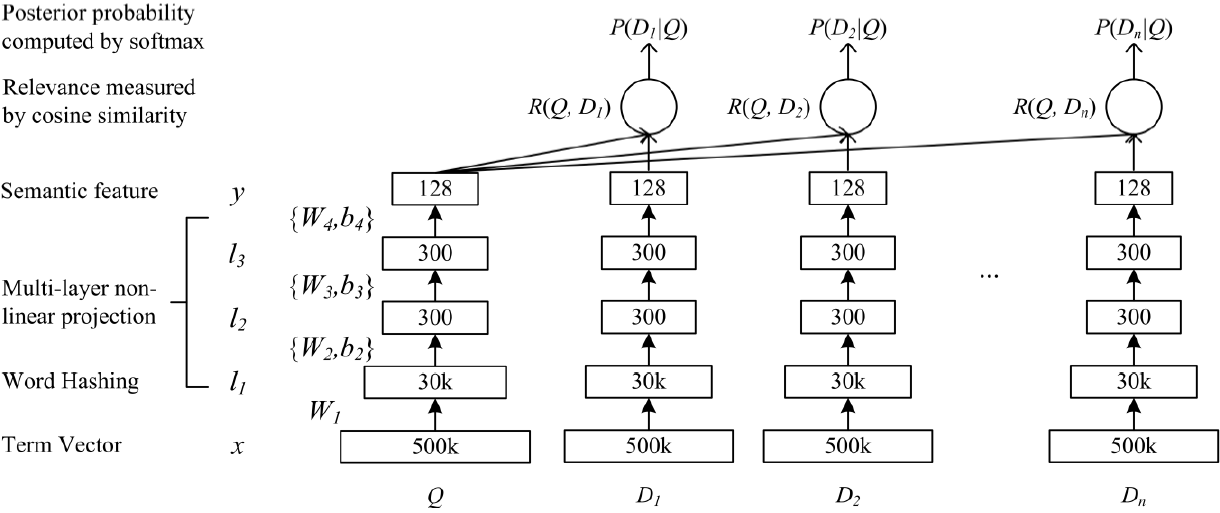
DSSM从上往下可以分为三层结构：输入层，表示层，匹配层



输入层：把句子映射到一个向量空间并输入到神经网络当中。中文处理方式一般采用字向量（one—hot）作为输入

表示层：DSSM的表示层采用BOW（词袋模型）的方式，相当于把字向量的位置信息抛弃了，整个句子的词都放在一个袋子里，不分先后顺序。

紧接着是一个含有多个隐藏层的DNN

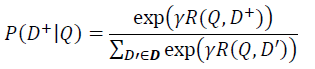


最终得到一个128维的低维语义向量

匹配层：Query和Doc的语义相似性可以用这两个语义向量的cosine距离来表示：

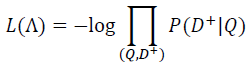


通过softmax函数可以把Query与Doc的语义相似性转化为一个后验概率、



其中r为softmax的平滑因子，D+为Query下的正样本，D‘为Query下的负样本（随机负采样），D为Query下的整个样本空间

在训练阶段，通过极大似然估计，最小化损失函数

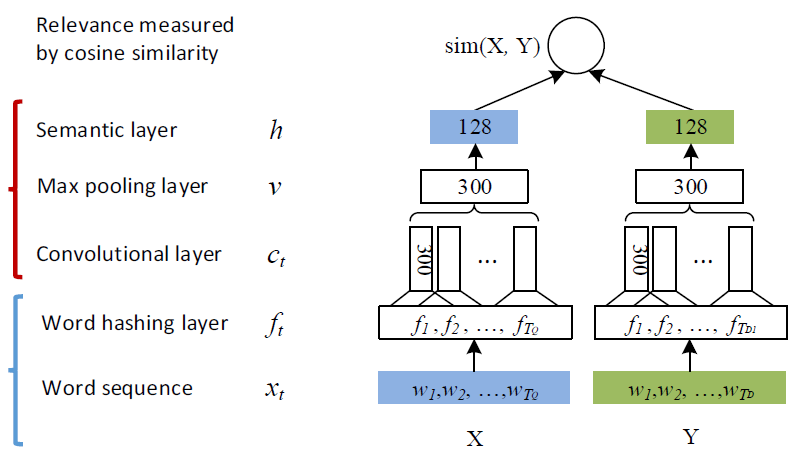


优点：DSMM用字向量作为输入既可以减少切词的依赖，又可以提高模型的范化能力，因为每一个汉字所能表达的语义是可以复用的。另一方面，传统的输入层是用Embedding的方式来做词的映射，再把各个词的向量累加或者拼接起来，由于Word2vec是无监督的训练，这样会给整个模型引入误差，DSMM采用统一的有监督训练，不需要在中间过程做无监督模型的映射，因此精准度会比较高。

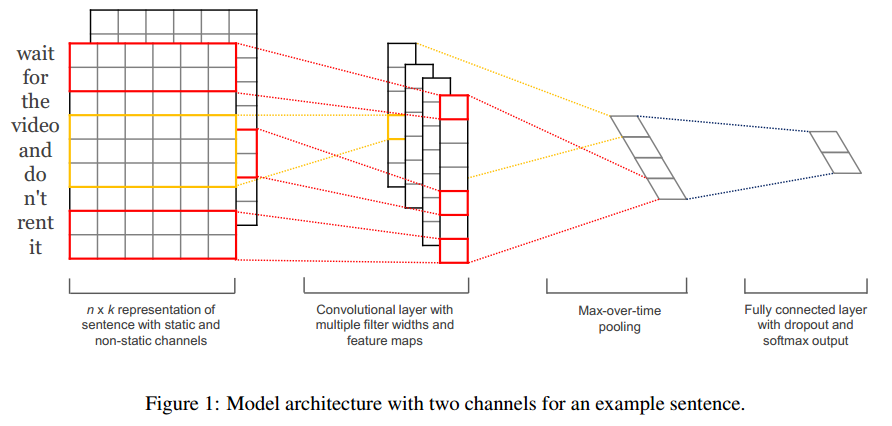
缺点：DSMM采用了词袋模型，丧失了语序信息和上下文信息。另一方面，DSMM采用弱监督，端到端的模型，预测结果不可控。

**CNN-DSSM：**主要针对DSSM词袋模型丢失上下文信息的特点。与DSMM的主要区别在于表示层

表示层：CNN-DSSM的表示层由一个卷积神经网络组成



卷积层：作用是提取滑动窗口下的上下文特征



最大池化层：为句子找到全局的上下文特征。

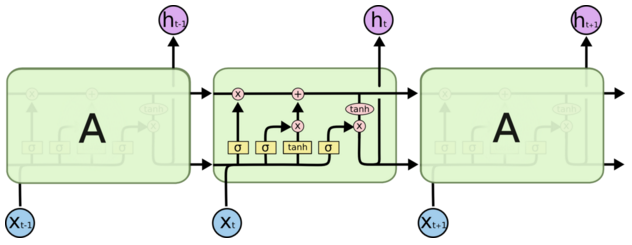
全连接层：通过全连接层把一个300维的向量转化维一个128维的低维语义向量，全连接层采用tanh激活函数。

优点：CNN-DSSM通过卷积层提取了滑动窗口下的上下文信息，又通过池化层提取了全局的上下文信息，上下文信息得到较为有效的保留。

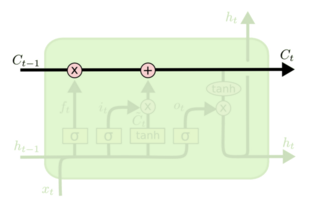
缺点：由于卷积核大小的限制，对于间隔较远的上下文信息，难以有效保留。

**LSTM-DSSM：**用来解决CNN-DSSM无法捕获较远上下文特征的缺点

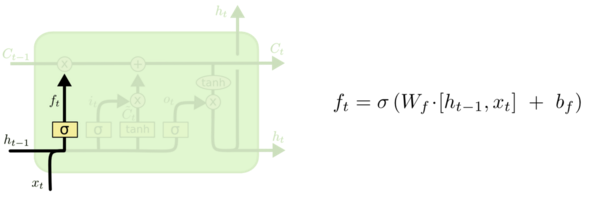
在这之前，我们先介绍一下**LSTM：**LSTM是一种RNN特殊的类型，可以学习长期依赖信息。我们分别来介绍它最重要的几个模块。



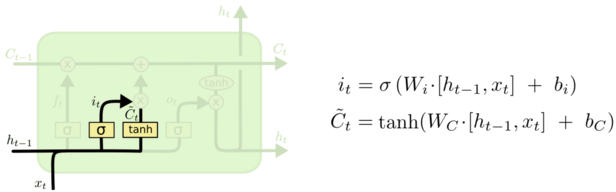
细胞状态：这条线可以理解成是一条信息的传送带，只有一些少量的线性交互，可保持信息的不变性。



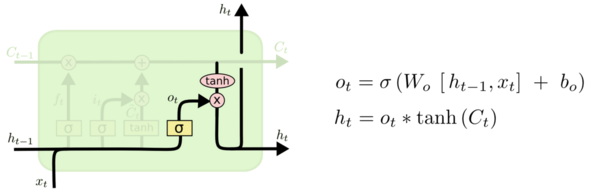
遗忘门：用来控制细胞状态cell有那些信息可以通过，继续往下传递。如下图所示，上一层的输出h（t-1）联结上本层的输入x\_t，经过一个sigmoid网络（遗忘门）产生一个0-1的数值f\_t，然后与细胞状态C（t-1）相乘，最终决定有多少细胞状态可以继续往后传递。



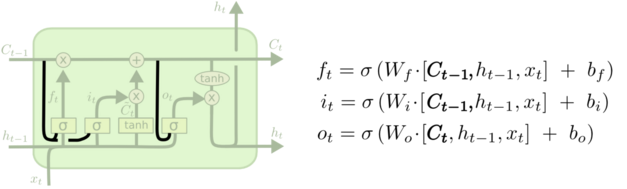
输入门：输入门决定要新增什么信息到细胞状态，这里包含两部分，一个sigmoid输入门和一个tanh函数。Sigmoid决定输入的信号控制，tanh决定输入什么内容。如下图所示，上一层的输出h（t-1）联结本层的输入x\_t，经过一个sigmoid网络产生一个0-1的数值i\_t,同样的信息经过tanh网络做非线性变换得到C\_t，sigmoid的结果和tanh的结果相乘，最终决定有那些信息可以输入到细胞状态里。



输出门：输出门决定细胞状态要输出什么信息，这里也包含两部分，一个sigmoid和一个tanh函数，sigmoid决定输出的信号控制，tanh决定输出什么内容。如下图所示，上一层的输出h（t-1）联结本层的输入x\_t,经过一个sigmoid网络产生一个0-1的数值O\_t,细胞状态C\_t，经过tanh网络做非线性变换，得到结果再与sigmoid的结果O\_t相乘，最终决定有哪些信息可以输出，输出的结果h\_t作为这个细胞的输出，传递个下一个细胞。

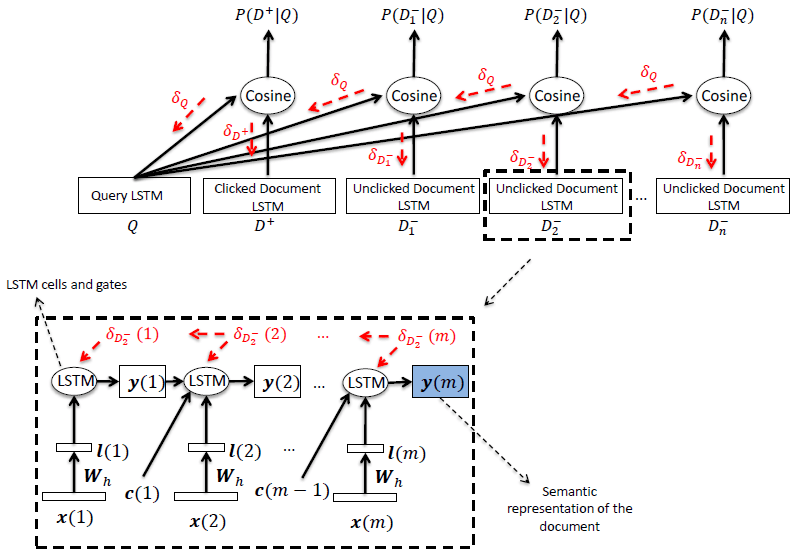


LSTM—DSSM用的是一个加入了peephole的LSTM，如下图所示



这三条黑线就是peephole，传统的LSTM中遗忘门，输入门和输出门只用了h（t-1）和x\_t,来控制门缝的大小，peephole的意思是说不但要考虑h（t-1）和x\_t,也要考虑C\_t-1和C\_t，其中遗忘门和输入门考虑了C\_t-1,而输出门考虑了C\_t。总体来说需要考虑的信息更丰富了。

接下来来看一下LSTM—DSSM整体的网络结构：



DSSM客观存在的缺点：

DSSM是端到端的模型，虽然省去了人工特征转化，特征工程和特征组合，但端到端的模型有个问题就是效果不可控。对于一些要保证较高的准确率的场景，用有人工标注的query’分类作为打底，再结合无监督的word2vec进行语义特征的向量化，显然比较可控。

DSSM是弱监督模型，需要有海量的训练样本，且需要大型的GPU集群。

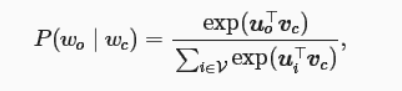
三．**词嵌入（word2vec）：把词映射为实数域向量**

介绍：Word2vec包括了一组用于word embedding的模型，这些模型通常都是用两层神经网络训练词向量。Word2vec的模型以大规模语料库作为输入，生成一个向量空间。词典中的每一个词都对应了向量空间中的一个独一的向量，而且语料库中拥有共同上下文的映射到空间向量中的距离会更近，也就是使这些向量能够较好地表达不同词之间的相似和类比关系。

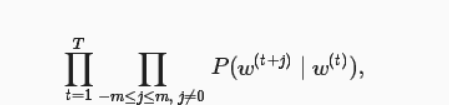
Word2vec的两个模型：

Skip gram：

这个模型用某个词来生成它在文本序列周围的词。它所关心的是，给定中心词后，生成与中心词距离不超过背景窗口大小的背景词的概率。

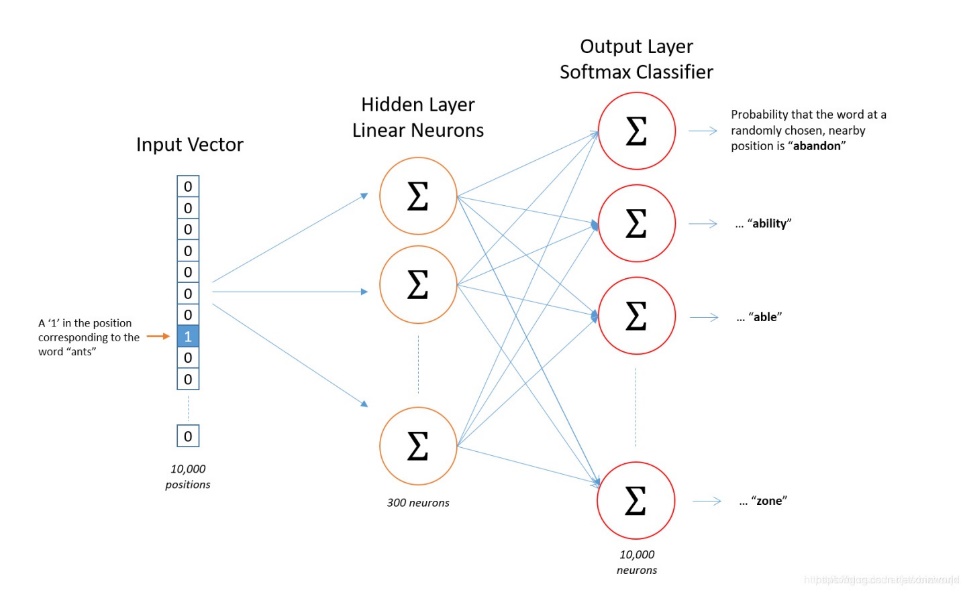
在跳字模型中，每个词都被表示成两个d维向量，用来计算条件概率。假设这个词在词典中索引为i，则当它为中心词时向量表示为v\_i∈R^d，而为背景词时向量表示为u\_i∈R^d,。设中心词W\_c在词典中的索引为c，背景词W\_o在词典中的索引为o，给定中心词生成背景词的条件概率可以通过对向量内积做softmax运算而得到：

设词典集V={0,1,……,V-1}。假设给定一个长度为T的文本序列，设时间步t的词为W^(t).假设给定中心词的情况下背景词的生成相互独立，当背景窗口为m时，跳字模型的似然函数即给定任一中心词生成所有背景词的概率



可以看出，word2vec的核心方法就是认为每个词都是相互独立的，用连乘来估计最大似然函数，求解目标函数就是最大化似然函数。

Skip-gram可以表示为由输入层，映射层和输出层组成的神经网络

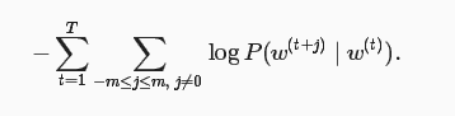


输入：每个词用one-hot编码方式表示，所有词都是一个N维向量

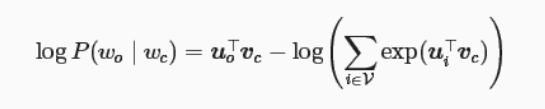
网络中传播的前向过程：输出层向量的值可以通过隐含层向量（K维）以及连接隐藏层和输出层之间的K\*N维权重矩阵计算得到。输出层也是一个N维向量，每维与词汇表中的一个单词相对应。最后对输出层向量应用softmax激活函数，可以计算每一个单词的生词概率。

Skip-gram模型的训练

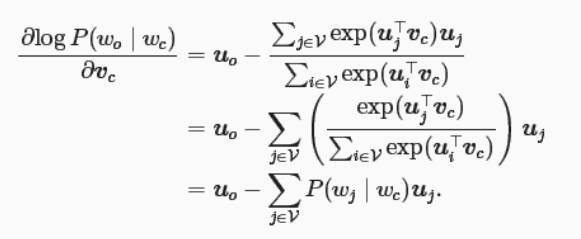
训练中我们通过最大化似然函数来学习模型参数，这就等价于最小化以下目标函数



如果使用梯度下降的话，那么在每一次迭代里我们都随机采样一个较短的子序列来计算有关该子序列的损失，然后计算梯度来更新模型参数。梯度计算的关键是条件概率的对数有关中心词向量和背景词向量的梯度。



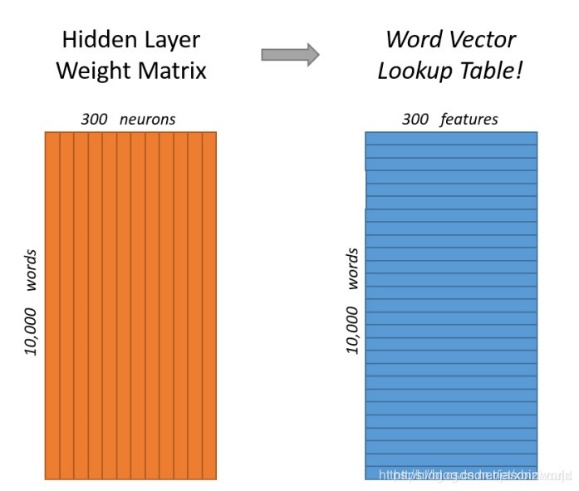
通过微分，我们就可以得到V\_c的梯度



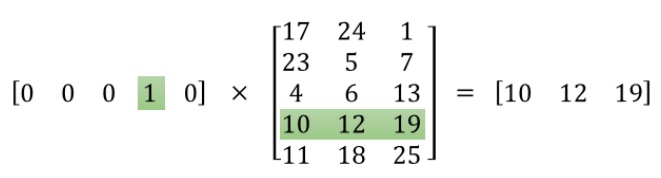
可以看出，它的计算需要词典中所有词以W\_c为中心词的条件概率。

训练结束后，对于词典中的任一索引为i的词，我们均得到该词作为中心词和背景词的两组词向量v\_i和u\_i。一般使用中心词向量作为词的表征向量。

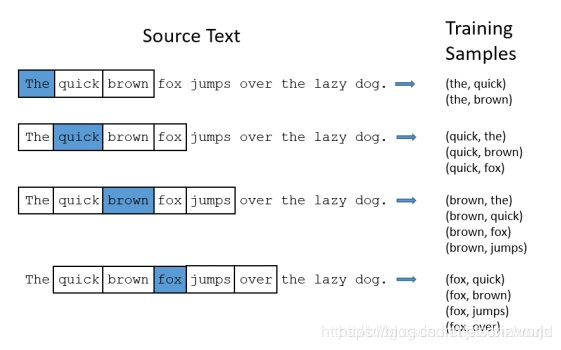
词向量即上图中的input层到hidden层中间的weight矩阵。



下面是对这个过程的简单可视化示意图。左边矩阵为词汇表第四个单词的one-hot表示，右边矩阵包含3个神经元的隐藏层的权重矩阵，做矩阵乘法就是从权重矩阵中选取了第四行的权重，因此，这个隐藏层的权重矩阵就是我们最终想要获得的词向量



如何获取训练样本：按照上下文窗口的大小从训练文本中提取出词汇对，然后将词汇对的两个单词使用one-hot编码就得到了训练用的train\_data和target\_data。下面给出了一个例子，设定窗口大小为2,，也就是说我们仅选取词前后各两个词和输入词进行组合。下图中，蓝色代表输入词，方框内代表窗口内的词。

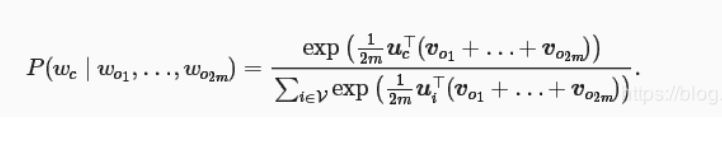


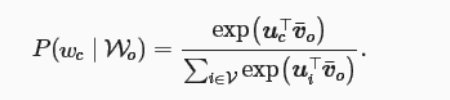
CBOW模型（连续词袋模型）

连续词袋模型与跳字模型最大的不同在于，连续词袋模型假设基于某中心词在文本序列前后的背景词来生成该中心词。他关心的是，给定背景词来预测中心词的条件概率。CBOW对小型数据库比较合适，而Skip-gram在大型预料中表现更好。

因为连续词袋模型的背景词有多个，所以我们将这些背景词向量取平均，然后使用和跳字模型一样的方法来计算条件概率。设v\_i∈R^d和u\_i∈R^d分别表示词典中索引为i的词作为

背景词和中心词的概率。设中心词W\_c在词典中的索引为c，背景词W\_o1……W\_o2m在词典中索引为o1……o2m,那么给定背景词生成中心词的条件概率为

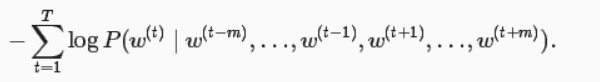


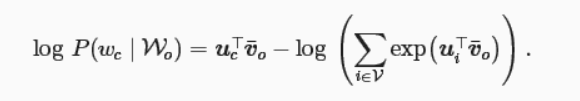
我们设，，那么上式可以简写成

给定一个长度为T的文本序列，设时间步t的词为W\_(t),背景窗口大小为m，连续词袋模型的似然函数是由背景词生成任一中心词的概率。

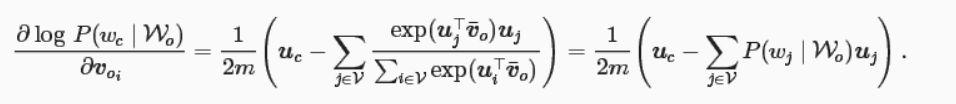
训练连续词袋模型

连续词袋模型的最大似然估计等价于最小化损失函数



注意到

通过微分，我们可以计算出上式中条件概率的对数有关任一背景词向量v\_oi的梯度



我们一般用词袋模型的背景向量作为词的表征向量。

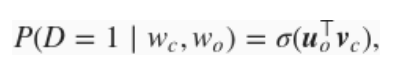
不论是跳字模型还是连续词袋模型，我们实际上都是取得input—hidden这个词向量（weight矩阵）。

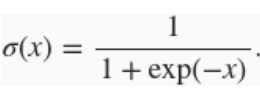
近似训练：

我们可以看到，无论是跳字模型还是连续词袋模型，由于条件概率使用了softmax运算，每一步的梯度计算都包含词典大小数目的累加，对于较大的词典，每次梯度计算开销可能过大，为了降低该计算复杂度，有两种近似训练的方法，即负采样和层序softmax。下面以跳字模型来介绍这两种方法。

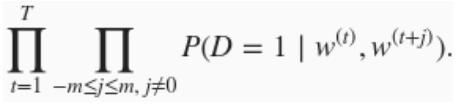
负采样：

负采样修改了原来的目标函数。给定中心词w\_c的一个背景窗口，我们把背景词w\_o出现在该背景窗口看作一个事件，并将该事件的概率计算为



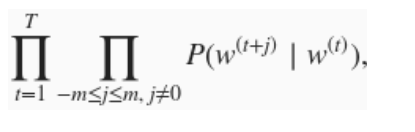
其中即sigmoid激活函数。

我们先考虑最大化文本序列中所有该事件的联合概率来训练词向量。具体来说，给定一个长度为T的文本序列，设时间步t的词为W^(t)，且背景窗口大小为m，考虑最大化联合概率

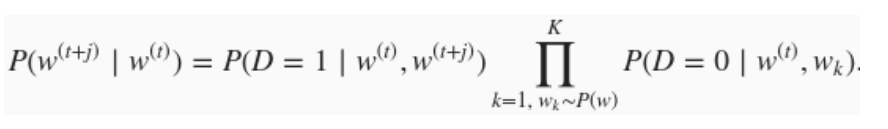


但是，以上模型中包含的时间仅考虑了正类样本，即给定中心词的背景词。这导致当所有词向量相等且值为无穷大时，以上的联合概率才被最大化为1。这样的词向量毫无意义。负采样通过采样并添加负类样本使目标函数更有意义。

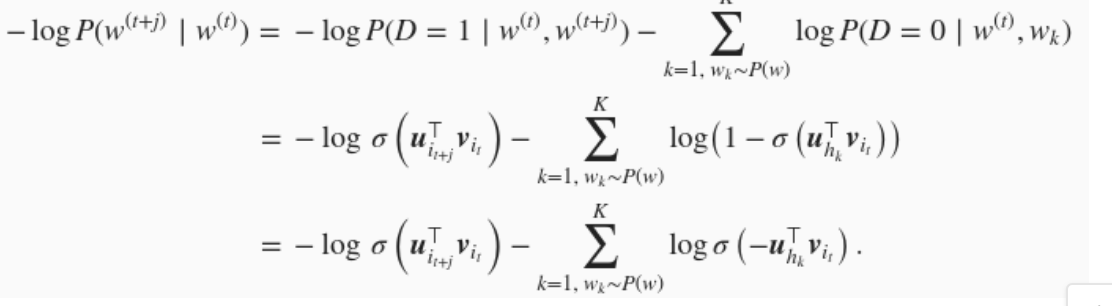
设背景词w\_o出现在中心词w\_c的一个背景窗口为事件P，我们根据分布P(w)采样K个为出现在背景窗口中的词，即噪声词。设噪声词w\_k（k=1……K）不出现在中心词的背景窗口为事件N\_k。假设同时含有正类样本和负类样本的事件P,N\_1,…,N\_k相互独立，负样本将以上需要最大化的仅考虑正类样本的联合概率改写为



其中条件概率被近似表示为



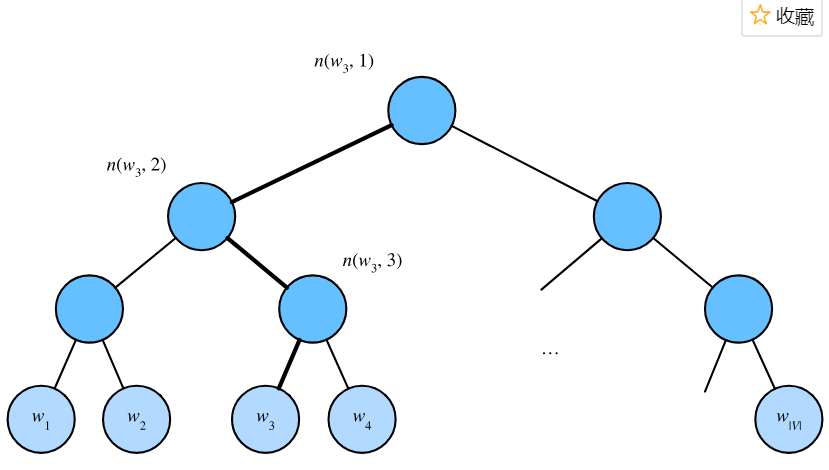
设文本序列中时间步t的词W^(t)在词典中的索引为i\_t,噪声词w\_k在词典中的索引为h\_k.以上有关条件概率的对数损失为



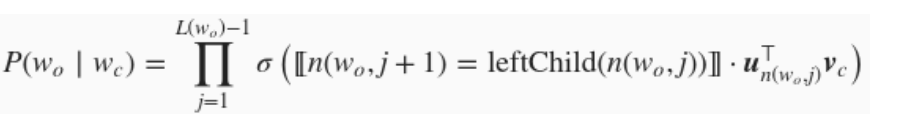
现在，词典中每一步梯度计算开销不再与词典大小相关，而与负采样的个数K线性相关。

层序softmax：

层序sofmax使用了二叉树这一数据结构，树的每个叶结点代表词典V中的每个词。

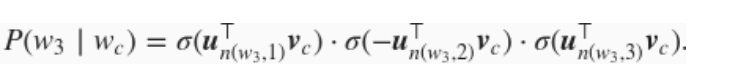


假设L(w)为从二叉树的根结点到词w的叶结点的路径上的结点数。设n(w,j)为该路径上第j个结点，并设该结点的背景词向量为u\_n(w,j)。层序softmax的条件概率近似表示为



其中，leftChild(n)是结点n的左子结点，如果判断x为真，则，反之。

比如我们计算给定词w\_c生成词w\_3的条件概率，我们需要将w\_c的词向量v\_c和根结点到w\_3路径上的非叶节点向量求内积。在二叉树中由根结点到叶结点w\_3的路径需要向左，向右再向左的遍历，所以我们得到



由于的数量级为，所以当词典V很大时，层序softmax在训练中每一步的梯度计算开销比未使用近似训练时大幅度降低。

**五．BiLSTM+CRF实现命名体识别**

基于字标注的中文分词

“我/O 爱/O 北/B 京/E 天/B 安/M 门/E”。

B | 词首  
M | 词中  
E | 词尾  
O | 单字

建立一个word2id词典，把每个汉字转换成id，再建立一个tag2id词典，把每一个字标注的类型转换为id，按照一一对应的顺序，把汉字和每个字的标签转换成id，分别存到两个数组里，一起保存到一个pki文件中，模型使用时可以直接读取。

准确度判断

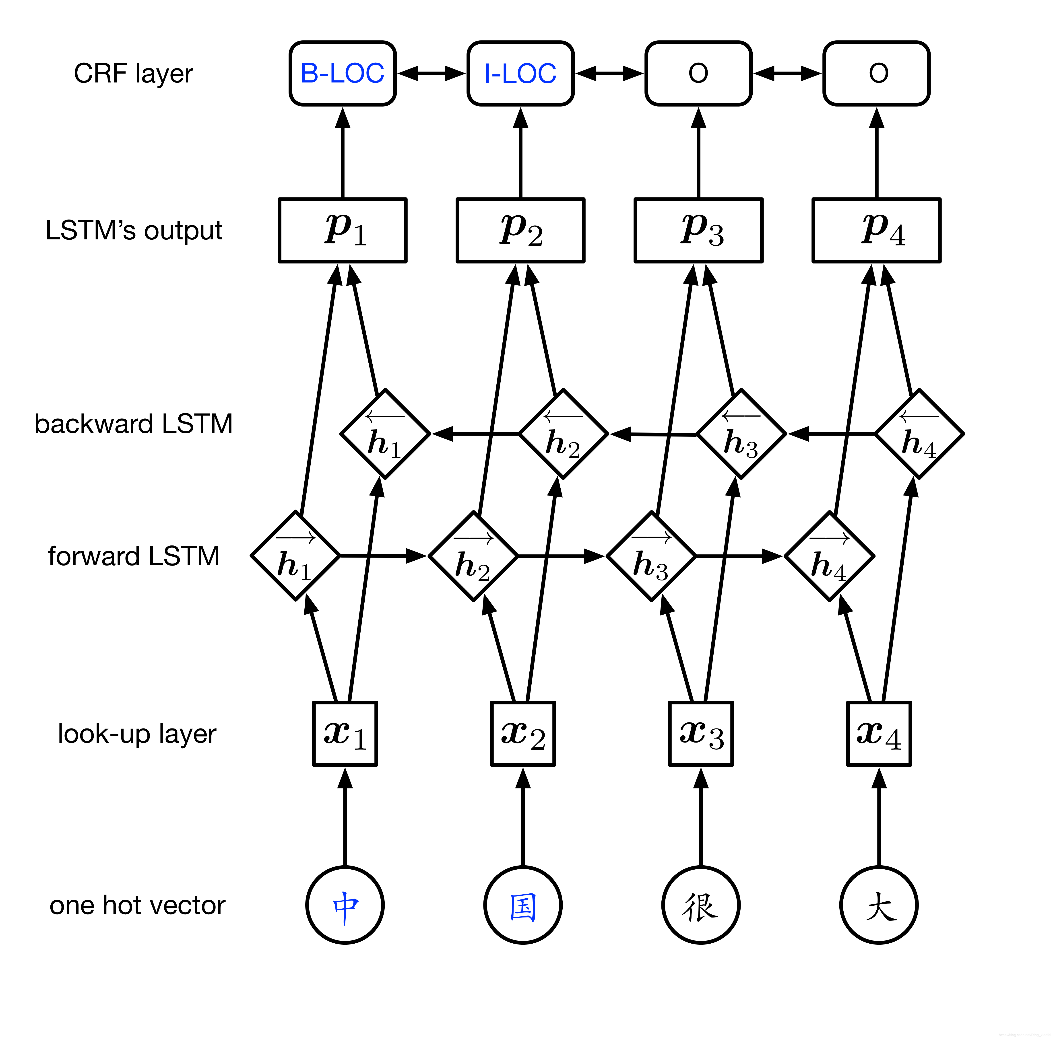
命名实体识别的准确度判断有三个值。准确率，召回率和f值。

这里需要先定义一个交集，经过模型抽取出来的实体与数据集中的所有实体取交集

准确率=交集/模型抽取出的实体

召回率=交集/数据集中所有实体

F值=2 x (准确率 x 召回率)/（准确率+召回率）



模型的第一层是look-up层，利用预训练或随机初始化embedding矩阵将句子中每个字x\_i由one-hot向量映射为低维稠密的字向量。d是embedding的维度，在输入下一层之前，设置dropout缓解过拟合。

第二层是双向LSTM层自动提取句子特征，将一个句子的各个字的字向量序列（x\_1,……x\_n）

作为双向LSTM各个时间步的输入，再将正向LSTM输出的隐状态序列与反向LSTM在各个位置输出的隐状态进行按位置拼接，得到完整的隐状态序列。



在设置dropout后，接入一个线性层，将隐状态向量从m维映射到k维，k是标注集的标签数，从而得到自动提取的句子特征，记作矩阵，可以把的每一维都视作字x\_i分类到第j个标签的打分值，如果再对P进行softmax的话，就相当于对各个位置独立进行k类分类。但是这样对各个位置进行标注时无法利用已经标注过的信息，所以接下来将进入一个CRF层来进行标注。

第三层是CRF层，进行句子级的序列标注。CRF层的参数是一个（k+2）x（k+2）的矩阵表示的是从第i个标签到第j个标签的转移得分，进而在一个位置进行标注的时候可以利用此前标注过的标签，之所以要加2是因为要为句子首部添加一个起始状态以及为句子尾部添加一个终止状态如果记一个长度等于句子长度的标签序列，那么模型对于句子x的标签等于y的打分为



由公式可以看出整个序列的打分等于各个位置的打分之和，而每个位置的打分由两部分得到，一部分是由LSTM输出决定，另一部分则由CRF的转移矩阵A决定。进而可以利用softmax得到归一化后的概率：



模型训练时通过最大化对数似然函数，下式给出了对一个训练样本的对数似然如果这个算法要式子实现的话，指数的和的对数要转换成

模型在预测过程时使用动态规划的Viterbi算法来求解最优路径：

