Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

1.背景：解决语义模型中话题和文档的对应，具体为将两者映射到一个隐含空间，在这个隐含空间中进行相似度匹配

2.已有模型：

BLTM：话题和文档同分布，每个话题分配词组

DPM：提取两者“概念向量”，计算距离

使用自编码器：语义hashing提取关键向量 问题：参数更新不直接针对分类、关键向量提取受大小限制及运算限制

3.改变方法：

DNN映射到相同语义空间进行余弦相似度计算

根据点击数据进行有监督学习的分类

引入词哈希降维（字母的n-gram，简单粗暴）

4.模型：

高维文本特征向量->词哈希->多层神经网络->cos相似度->softmax输出

5.训练方法：

传统的对数似然加上平滑系数的超参

6.创新点：

总结一下，原方案的优化是把文档中的词更好的建模（像k-means点的投影），再根据全部词的投影分类，这里就引入一些限制（数据处理部分提及）直接划为类似多分类问题（标题和话题对应），直接对分类参数进行优化，自然而然的使用了DNN。

还有就是引入了词哈希这个小trick。

BiLSTM+CRF实验

两篇经典的论文：

Bidirectional LSTM-CRF Models for Sequence Tagging

1.Abstract：

state of the art (or close to) accuracy on POS, chunking and

NER data sets， less dependence on word embedding

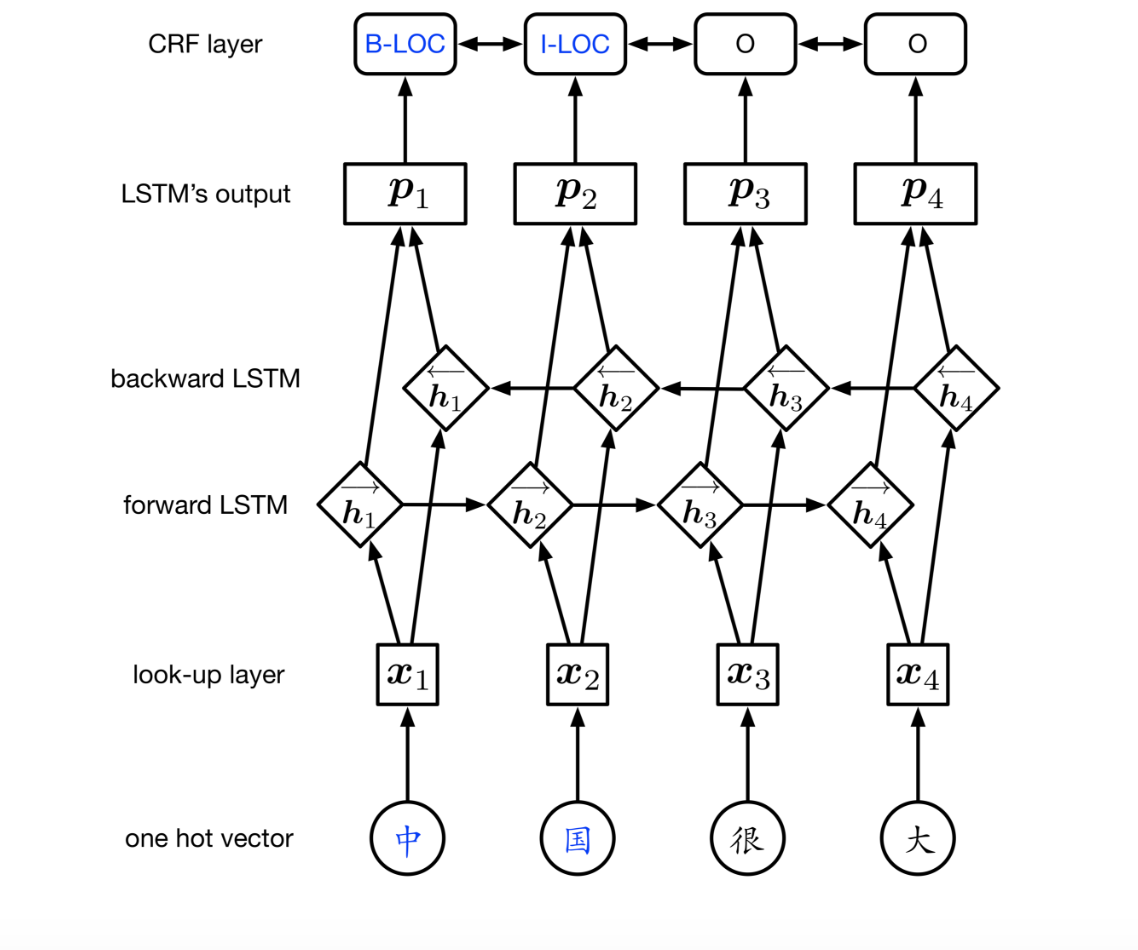
2.Introduction：

LSTM:use both past and future input features

CRF:use sentence level tag information

3.Model:

Bi-LSTM:BPTT、dropout



Neural Architectures for Named Entity Recognition

1.Abstract：

models rely on  character-based word representations learned from the supervised

corpus and unsupervised word representations learned from unannotated corpora

2.Introduction:

NER problem: a very small amount of supervised training data available,few constraints on word

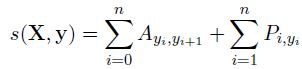
(1)NER consist of multiple tokens BiLSTM+CRF和stack-LSTM 字向量

(2)one token:词向量

dropout提高泛化

3.Model：

LSTM直接输入对POS较有效，对NER这种对外部语料依赖性很大的效果较差，因此把LSTM输出作为打分矩阵，表示某个词被标注为某个标签的概率



打分系数，其中A为标签转移得分，训练时使用极大似然估计

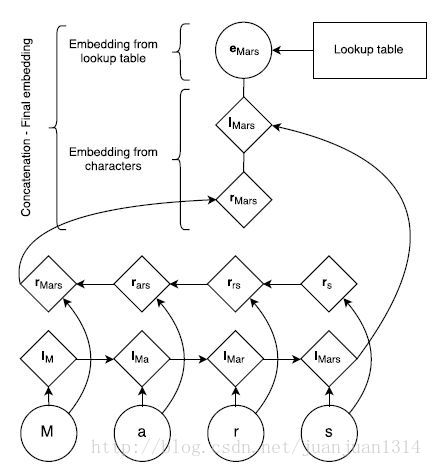
4.Parameterization and Training

变量：打分矩阵A,BILSTM计算P参数,词向量

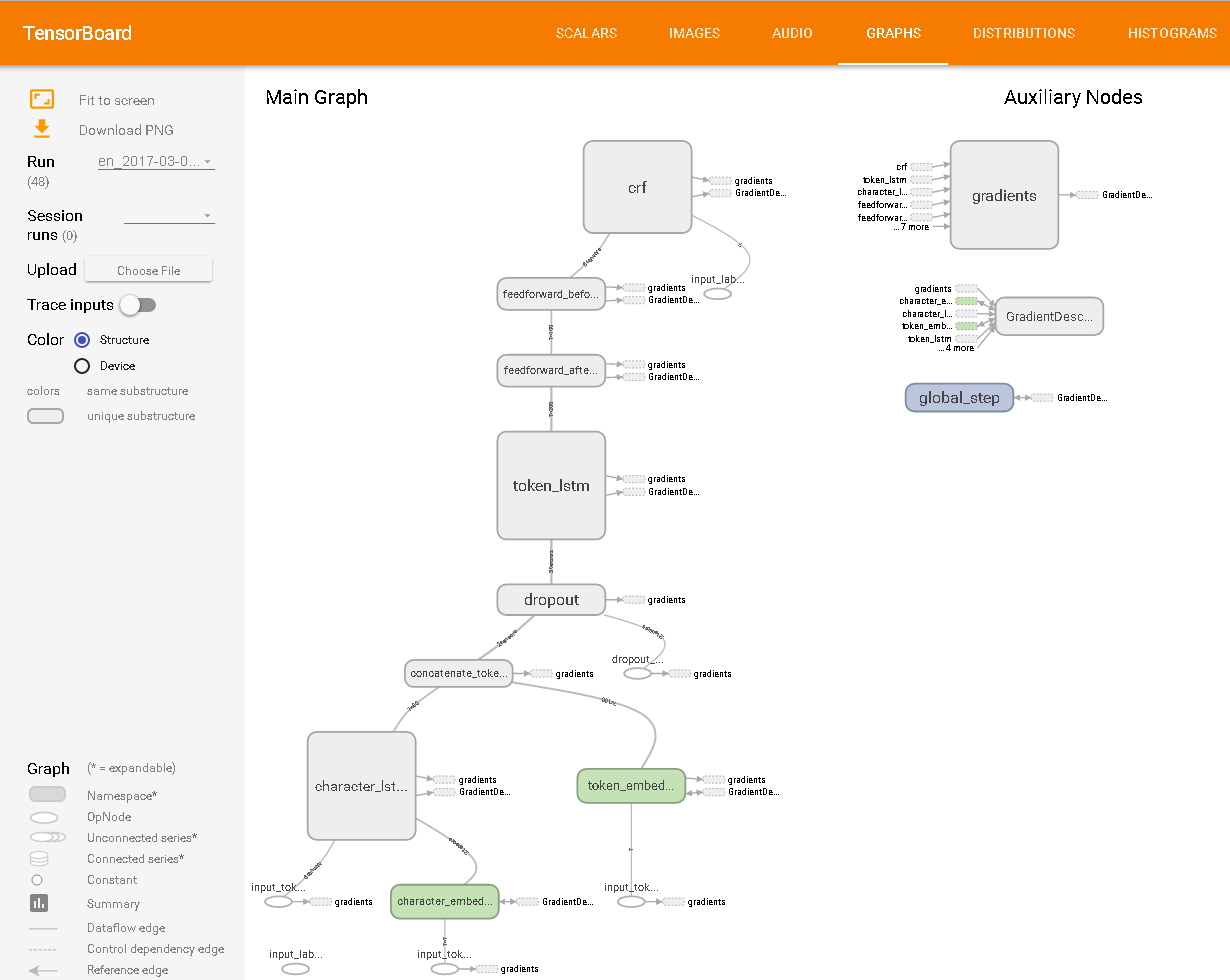
若A随机生成 似乎效果也不是很好

为了更好的效果，把LSTM输出全连接接入线性隐层

5.word embeddings



随机生成字母向量表，根据LSTM正序逆序加上原先词向量获得最后的词向量（也可用cnn提取前缀后缀特征）



tensorboard实现图

进阶（未阅读论文）：attention based

《Attending to Characters in Neural Sequence Labeling Models》

《Phonologically aware neural model for named entity recognition in low resource transfer settings》

LSTM+CRF+attention概念 达到了瓶颈

迁移学习《Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks》和半监督学习。

《Semi-supervised sequence tagging with bidirectional language models》

实验用的是github已完整实现的命名实体识别的代码

ner语料为msra语料库

剖析代码：

data.py 数据预处理

eval.py 使用一个工具自动输出准确率、召回率、F值

main.py 主程序入口

model.py tensorflow程序

utils.py 工具函数如标签转换、训练信息记录

main.py

1.使用argparse包从控制台传参数

一些超参：是否使用预训练词向量，词向量是否更新，是否使用梯度裁剪

2.初始化词向量，输出训练信息到文件，开始训练

model.py

1.运行顺序：

self.add\_placeholders()

self.lookup\_layer\_op()

self.biLSTM\_layer\_op()

self.softmax\_pred\_op()

self.loss\_op()

self.trainstep\_op()

self.init\_op()

2.构造词向量：tf.nn.embedding\_lookup 返回词向量的特定字向量，这里为所有字的初始字向量，有300维...并使用了dropout

3.双向LSTM层：调用tf函数，返回包含前向后向输出的列表，使用了dropout

初始化权重矩阵W，偏置矩阵b

4.定义loss 使用crf\_log\_likelihood函数

tensorflow的crf函数就是按照论文Neural Architectures for Named Entity Recognition写的 分解为转移+标签的损失值

5.train 优化方法默认为Adam 梯度更新

不看源码就真的是搭积木一层一层调函数..

6.其他函数

predict\_one\_batch：viterbi算法输出

默认参数：

batch size = 64

optimizer:adam

dropout = 0.5

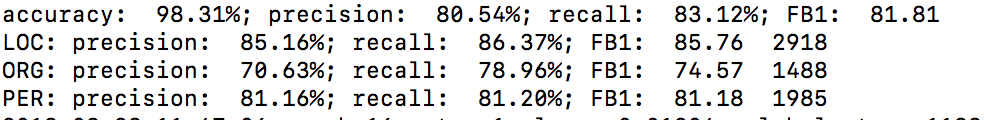
gradient clip = 5 梯度裁剪

update embedding = true

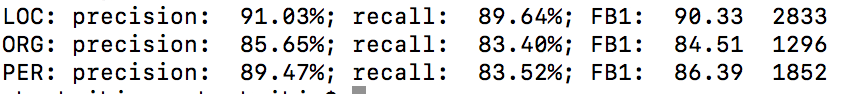
epoch = 40

作者的默认数据都是到40轮，我只跑了15轮到达相对接近的准确率就停下来了

无CRF Bilstm后接softmax输出 loss一下子就下降到很低（局部信息易拟合？）在0.03左右一直波动，准确率缓慢提高



有CRF默认数据 loss稳定变小最后到0.15左右 但准确率变化已经比较少了 比不接CRF的更快拟合 总体准确率比无CRF的更高 ORG组织名明显比无CRF的好多了 看起来对于组织名的实体识别 前后几个字的线性加权特别重要



以下单跑一个CRF++的

参数：-f 3 -c 4

跑了1小时就完成了…传统方法还是快

accuracy is 0.141521111264  recall is 0.939211555823  f is 0.245978061254

准确率出了很大问题

准确率公式记错了..导致结果出了问题 一开始还以为是crf++参数的问题 这过程中一些自己的猜想在最后进行否定和解释

CRF++结果：参数 –f 1 –c 4

地名：accuracy is 0.845963416311  recall is 0.9573540856  f is 0.89821845794

组织：accuracy is 0.852735323525  recall is 0.9008601177  f is 0.87613736425

人名：accuracy is 0.916380494505  recall is 0.9549114331  f is 0.93524927714

实验对比：

三个准确率为地名、组织、人名

BILSTM+softmax(15轮)准确率分别为85%、70%、81%

BILSTM+CRF(15轮)准确率分别为84%、85%、91%

作者跑了40轮准确率分别为91%、85%、87%

CRF++跑准确率分别为91%、85%、86%

如下表

|  |  |  |  |
| --- | --- | --- | --- |
| 模型/实体类型 | 地名 | 组织 | 人名 |
| BILSTM+softmax | 85% | 70% | 81% |
| BILSTM+CRF | 84% | 85% | 91% |
| 作者(40轮) | 91% | 85% | 87% |
| CRF++ | 91% | 85% | 86% |

实验效果总结：

目前来看BILSTM+CRF效果略优于CRF++，只有BILSTM效果明显最差

时间来看，普通语料CRF++1 2个小时肯定能跑完，而用深度学习模型跑了15轮大概用了4个小时（只是接近稳定而已），是传统模型的好几倍。

结论：

1.训练集和测试集差距比较大，训练集普通句子多，命名实体少，测试集命名实体多了很多，但是这并不影响实验结果

2.在网上看到其他人用crf++做NER达到90%准确率，是先提取出了更多特征:词性、是否为实体结尾词、是否为句子结尾词等等，准确率能进一步提高

3.NER训练语料特别“稀疏”，但这并不会导致欠拟合，传统方法、深度学习方法都是这样，CRF++只使用出现3次以上特征对结果影响也不大

4.调参对传统方法CRF++影响并不大，正则化系数、特征选择等参数改变对结果影响不到1%，但深度学习方法参数选择不好效果远远不如传统方法CRF++

5.论文作者使用了预训练词向量，不清楚预训练的过程而且也不清楚是否对结果有比较大的提升，跑的代码是随机初始化的，对比CRF++效果提升并不大

6.第一次看完整的tensorflow程序，就算是一层层调包和写工具函数也感觉离自己能完整复现好像还挺遥远...