一种结合local和distributed文本相似度算法

> by joeyqzhou

基于频次的信息检索方法

信息检索(IR), 简单说，就是给一个query, 返回与其最相关的doc. 传统的IR方法有tfidf, BM25, 它们主要考虑的是query中词语的**确定性**匹配(不能匹配到类似词）。即词在某篇候选doc中出现的次数(term frequency)和在所有doc中出现的频次(idf)。这种方法的局限有: 不能把词语的相似性考虑进来，比如搜索"iphone"，苹果手机不会命中的

基于向量表达的信息检索方法

简单的稀疏表达

由于上述局限，一个term(词或字，后面简单用词表达term.)需要用向量来表达。简单的方法如

- onehot：词表数量是向量长度，这个词的位置为1，其他为0
- 出现在doc表达: 向量长度是doc的数量，值是该词在每个doc中出现的次数
- 词语共现表达: 词表数量是向量长度，值是该词与每个位置词的共现次数

等其他方法，但是这里方法局限是高维稀疏，长度不定，不方便接后续神经网络计算

Embedding表达:

又经常被称为: 隐语义、潜语义表达等。其含义是用低维稠密的向量来表达每个词. 这类方法根据有无标注数据，又可区分为无监督、有监督方法. 我们先简述无监督方法.

主题模型方法PLSA, LDA

PLSA： 每篇文章d都有多个主题c服从$p(c|d)$, 每个主题c在每个词语上符合分布$p(w|c)$, 那么对于现有的文档及其词语的联合分布为:

$$p(w,d) = p(d)p(w|d) = p(d)\sum_c p(c|d)p(w|c)$$

设置 topic数目为K, PLSA就是求解参数$p(c|d)$ （参数个数：文档数 x K),$p(w|c)$（参数个数:词典数xK），来使得这个联合分布的最大.

LDA 是在PLSA的基础上，认为参数不应该是fixed and unkonwn的，而是符合一个分布，该分布是Dirichlet Distribution

word2vec方法

划分一个固定长度的窗口，利用中心词预测周围词（skip-gram）,或周围词预测中心词（ cwob). 以skip-gram为例, 其目标函数为:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0} P(w_{t+j}|w_t)$$

其中:

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

注意u，v是两套向量体系，一般v是用来代表词向量在后续中使用。值得注意的点，word2vec目标使得**周围词语相似的词语更相似**，而主题模型是使得**相同文档内的词语更相似**。word2vec有如下两个优势：

- 窗户的文档长度的区别。一篇文档可能很长，距离很远的词关系并没有那么大
- Word2vec 使得“周围”词语相似的词语更相似。同义的词语可能很少共现。比如 “计算机, 电脑" 类似的近义词语共现比较少，但是周围的词语很相似

glove方法

也是划分窗口，统计任意两个词语的频次X_{ij}, 任意一个词语由v_i向量和一个bias b_i表示，其目标函数是:

$$J(\theta) = \sum_{ij} f(X_{ij})(v_i^T v_j + b_i + b_j - log(X_{ij}))$$

其中$f(X_{ij})$是根据频次给的一个权重

更复杂的embedding表达

最近几年由 LSTM, GRU这种有序的RNN网络 -> Elmo（也是时序的) -> transformer(基于attention, 不需要时序) -> bert 。 由对词语的表达， 单一的表达（上述的方法词的embedding都与上下文无关), 到针对句的表达（不需要分词），或词语根据上下文其embedding有变化。 网络不断变得复杂，输入的数据越来越大，表达能力越来越强。当然，针对计算量无论是训练、预测都是极大的挑战(最近也有针对bert网络进行蒸馏来优化效率的文章)

上面总结了无监督的常见embedding表达，词语或query/doc拿到了embedding表达，就可以通过cos相似度或类似方法，计算相似度了。这种基于向量表达的相似度计算，比tfidf这种传统方法的泛化能力更强，召回能力更强.

有监督的表达

一般情况下，无监督的效果是没有有监督的效果好的。所以最好的还是利用检索场景的反馈数据来进行有监督的训练。针对文本相似度（还没有考虑到个性化，点击率等因素），有监督训练的样本是(query, 点击/曝光, doc), 目前doc使用的是doc标题。

DSSM[1]（distributedf方式相关模型)

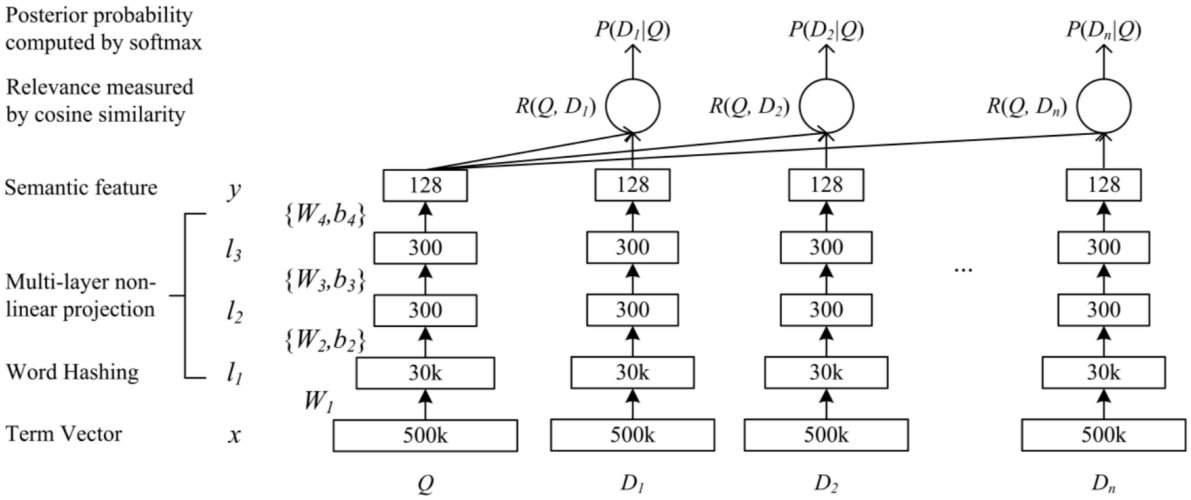


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear projections. The final layer's neural activities in this DNN form the feature in the semantic space.

word-hashing

原文中采用word-hashing方法，来对输入的词语进行处理。举例有输入词语"#good#", 那么"#good#"就被trigram分为"#go, goo, good, od#", 这样做的好处是避免覆盖的词语更全，避免出现out of vocab的情况，比如goooood job, 这种口语化的词语. 另外，减小了特征字典的大小，从500k, 将为30k.

因为trigram会出现碰撞，但是这种碰撞完全是可以接受的，另外模型训练会根据数据做一定适应。甚至对"goooood job"这种例子反而带来一种迁移的效果。

但是我们训练的场景是中文分词，所以就不采用该技术了。而是在分词的时候，结合了分词+bigram对中文进行全面的切割

模型

query和doc共享参数，都是走若干层全连接，最后一层是一个128维的向量。相似度由cosine计算:

$$R(Q,D) = \cos(y_Q, y_D) = \frac{y_Q^T y_D}{||y_Q|| ||y_D||}$$

那么doc D与query Q的相关的概率是:

$$P(D|Q) = \frac{\exp(rR(Q,D))}{\sum_{D' \in \mathbf{D}} \exp(rR(Q,D'))}$$

r是经验参数，**该参数设置也比较关键。如果该参数设置比较小，比如1，loss收敛的很慢，甚至不会收敛**。原因是R这个相似度是(-1,1)的，如果限制在（-1，1）那么正负样本的预测值的softmax无法拉开差距，所以r需要设置的较大，我们设置的是20.

其中$\mathbf{D}$ 可以代表和这个Q相关的所有doc集合。这个集合可能很大，可以采取随机采样.

其目标函数最大似然正样本的概率乘积，负样本在分母起作用:

$$J(\theta) = -\log \prod_{(Q,D^+)} P(D^+|Q)$$

DSSM的后续优化

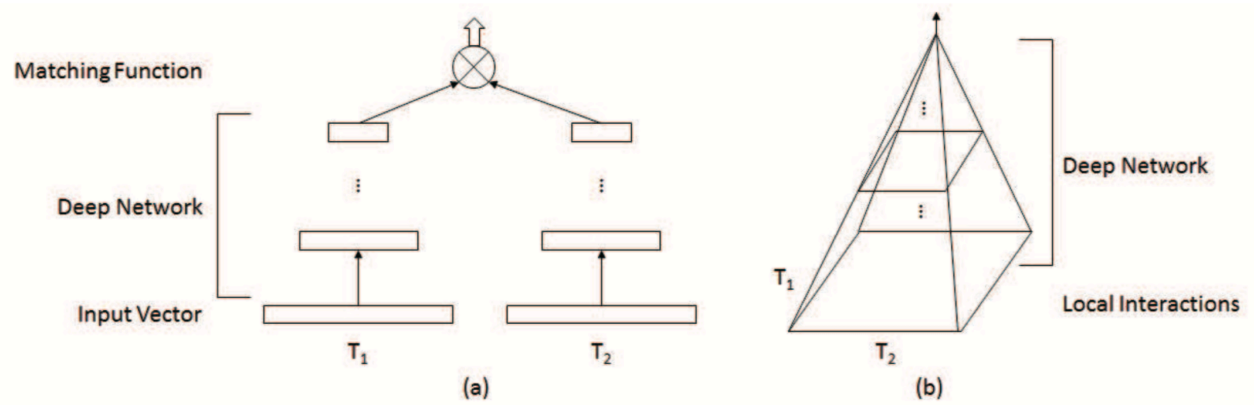
DSSM在2013年发表，产生了不错的效果。后续也有陆续的其他优化，如把DSSM后的全连接网络加入CNN层(CDSSM)，或结合LSTM把时序信息也加入进去，对效果有进一步提升。

不过针对，零售搜索领域，query一般比较简短一到两个词语，doc也经常是词语的拼接，不是完整的一句话。通过离线实验，所以增加了CNN, LSTM层对DSSM在零售的语料离线提升有限。

DRRM(local interaction方式)

文本相似度建模一般有如下两种方式，(a) 先分别计算query和doc的embedding, 再计算相似度， (b) 先计算query和doc的local interaction, 然后数据神经网络计算相似度，参考[2]一般匹配方式：

$$\text{Match}(T_1, T_2) = F(\phi(T_1), \phi(T_2))$$



T_1,T_2 表示等待对比两句话

A: distributed方式, (ϕ 复杂，F 简单)

两句话先拿复杂的网络表示为(ϕ), 最后一层简单结合(F)。DSSM 即属于distributed方式, 其ϕ 由若干层全连接表示，F 是cosine相似度

B: local interaction方式。ϕ 简单，F 复杂

简单组合两句话，输入复杂的模型计算。

代表方法: DRRM

为什么DSSM等distributed相关方法取得了不错的效果，还需要local interaction的方法呢？【2】中说明，"distributed are not a typical ad-hoc retrieval setting"，【3】文章中说明，distributed方式针对一些长尾的query表现的不如local方式好，所以local方式是仍然需要的。下面简单介绍下Deep Relevance Matching Model(DRRM)【2】

这个方法的特点:

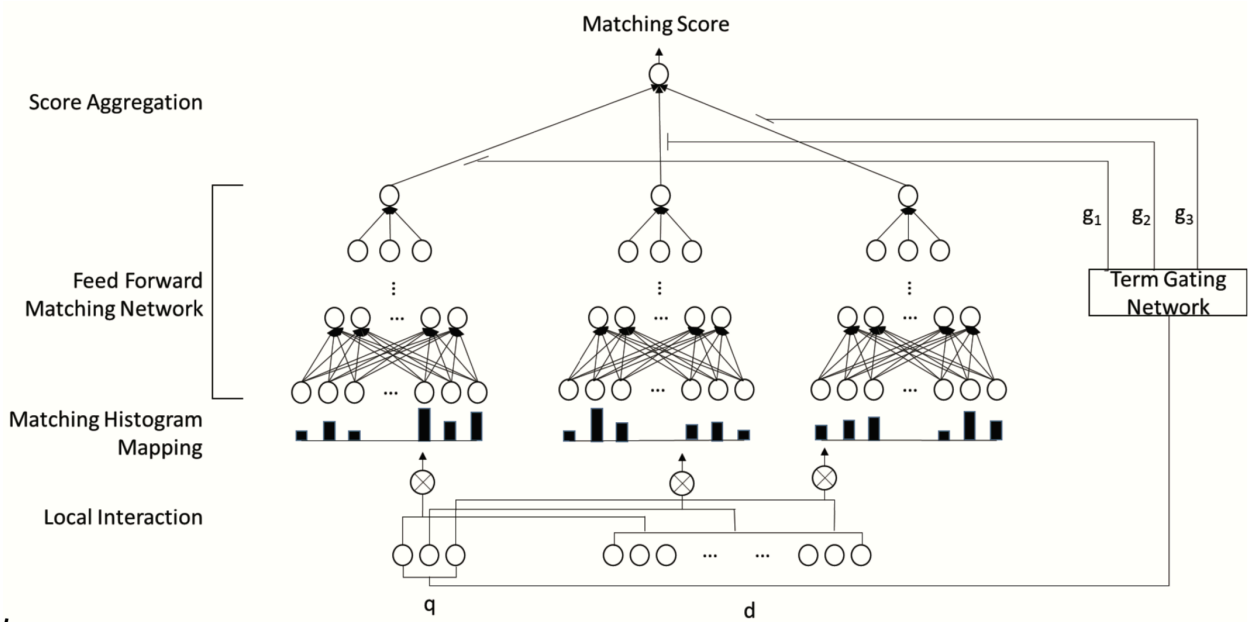
1. 其输入层是query word embed和doc word embed的相似度的直方图。举例[2],query是car, doc的标题分词是:(car, rent, truck, bump, injunction, runway), query词语car和doc的每个词语的相

似度分别是((1, 0.2, 0.7, 0.3, −0.1, 0.1)), 假设直方图的分桶是这样的:bins {[−1, −0.5), [−0.5, −0), [0, 0.5), [0.5, 1), [1, 1]} , 那么直方图为： [0, 1, 3, 1, 1]， 这个直方图就是神经网络的输入。 (针对tensorflow批量直方图快速计算，tf.histogram_fixed_width不支持批量的直方图计算，我的思考了一种方法供参考)

这里的直方图，可以用计数的，也可以”计数+归一化“， 也可以log(1+计数)。文章说最后一种log的方法效果最好

直观感觉，就是如果query与doc的词语越相似，在直方图的右边值越大，模型结果一般就越大。

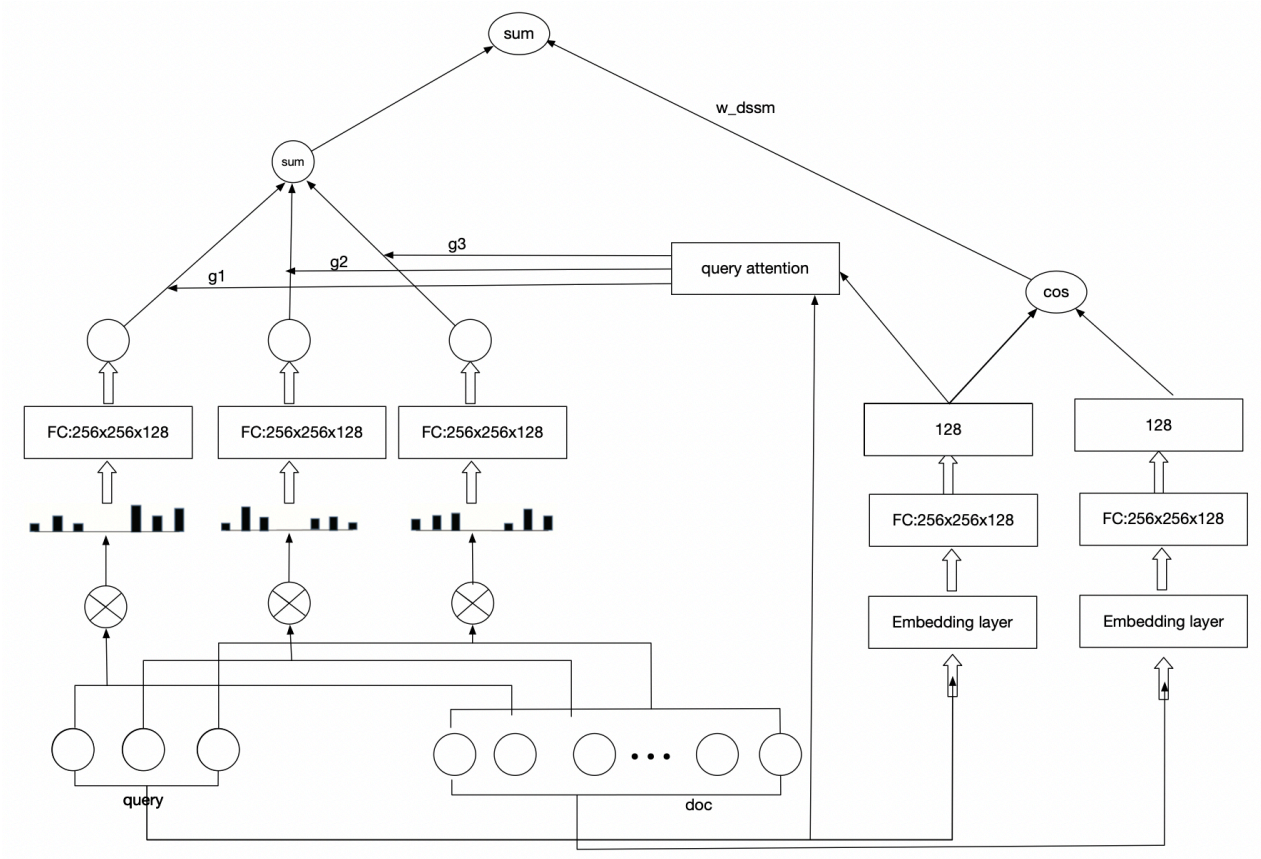
2. 区分了query中不同的词语的权重。一个词语一个网络，因为query中的词语重要性不同，在最后一层通过一个g_i 加权得到最后的分数



我们的实验，针对DRRM模型做了离线实验，效果并没有超过DSSM. 正如[3]中所有，DRRM是对长尾的query表现的更好，对热门的query表现一般，综合起来离线没有beat DSSM.

local + distributed 实践

[3]中做了local + distributed 的尝试，但是其local部分，只能记录exact match(确定性匹配)和match的位置, 而不能对相似的词语起作用，所以我们的实践中local的方式借鉴了DRRM的直方图的方式，并对其进行了优化。distributed部分，[3]使用了CNN层, 我们认为对于检索场景的query，CNN或RNN反而过于复杂, 准确率提升有限。所以仍然沿用了之前的DSSM. 所以我们的网络结构最终如图, 其中左侧是



local部分:

参考DRRM，表达query与doc直接计算内积相似度，以直方图输入神经网络.

注意我们同样对不同的query进行了加权，但与DRRM不同的是，这里不是简单的用一个常量权重来代表一个query词语。单一个常量权重不能衡量query中每个词语在整个query中的重要性， 即一个词语的权重不仅和这个词语本身有关，还和它所在query的整体有关。

所以我们使用一个类attention机制， 这个attention是衡量“一个query词语的embedding输入”与“query在DSSM网络的整体的输出层”的一个相关程度，通过一个attention matrix来连接两者.

设L 为query的词语长度（不足padding）， embedding长度为K , DSSM网络最后一层长度为$\mathrm{dssm_h3}$

即query从DSSM塔的输出层是attention的”q",即 $Q_{\mathrm{dssm\_h3}}$, 而每一个query有L个词语那么输入为$Q_{L,K}$, 还有一个注意力矩阵 $W_{K,\mathrm{dssm\_h3}}$, 那么DRRM最后每个的加权向量为

$$\vec{g}_L = \mathrm{softmax}(Q_{L,K} W_{K,\mathrm{dssm\_h3}} Q_{\mathrm{dssm\_h3}})$$

distributed部分

distributed部分和DSSM网络结构一样，区别是cosine值最后需要和local部分合并，可以看local的输出不是（-1，1)的，所以需要给distributed输出乘以一个扩展因子w_{dssm} ， 这样模型可以根据数据来自行调节该权重，从而有效结合两个部分.

融合

这样，模型的输出结果不是(-1,1)，和我们之前的DSSM的结果范围不一致。目前的系统，基于DSSM的输出范围(1-,1)，后续有一系列基于绝对值的规则，比如相关性绝对值过滤等。所以我们需要把输出放在一个范围内。思路是采用sigmoid对输出进行值域的控制，这样就可以把输出控制在(0,1)。但这样我们遇到前面DSSM说的问题，正负样本的输出拉不开，结果softmax后，loss不收敛，所以在sigmoid的基础上增加了一个扩展倍数的参数，初始化20，所以最终的输出是:

$$\text{sigmoid}(\text{drrm\_output} + \text{dssm\_output} * w_{\text{dssm}}) * w_{\text{amplify}}$$

现网预测的时候会把w_{amplify}除掉. 所以最终结果就是0~1了

实验与上线

目前离线证明，该方法的loss(0.010)远远低于DSSM的loss(0.023), 不得不说一方面这得益于网络结果变得更为负责，另一方面该网络针对长尾的词语拟合的更好.

引用

1. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data
2. A Deep Relevance Matching Model for Ad-hoc Retrieval
3. Learning to Match using Local and Distributed Representations of Text for Web Search