匹配模型的进展

陌路小北 deep炼金实验室 2020-02-29

本文将介绍NLP中常见的匹配模型，也应用在搜索和推荐等领域，笔者曾经的排序思路基本都是受到匹配模型的启发去应用的，因此我认为匹配的思想非常重要。

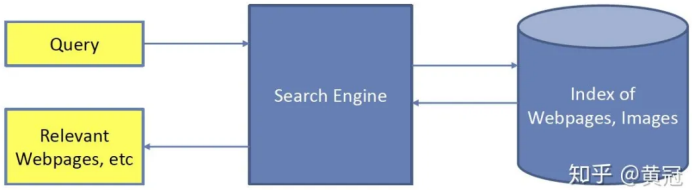
搜索和推荐本质其实都是匹配，搜索的本质是给定query，匹配doc；推荐的本质是给定user，推荐item。本文主要讲推荐系统里的匹配问题，包括传统匹配模型和深度学习模型。

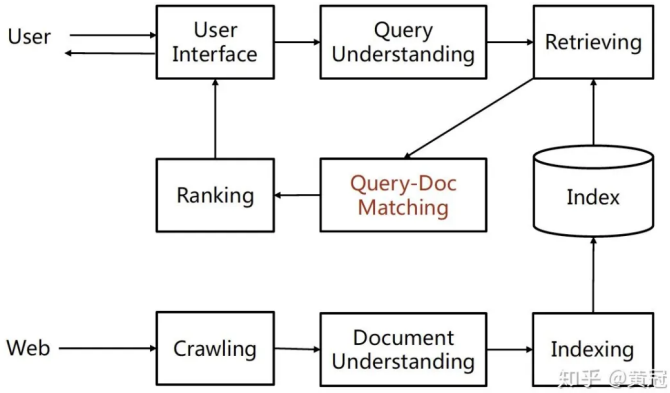
1-匹配综述

1.1 搜索和推荐的综述

- **搜索**

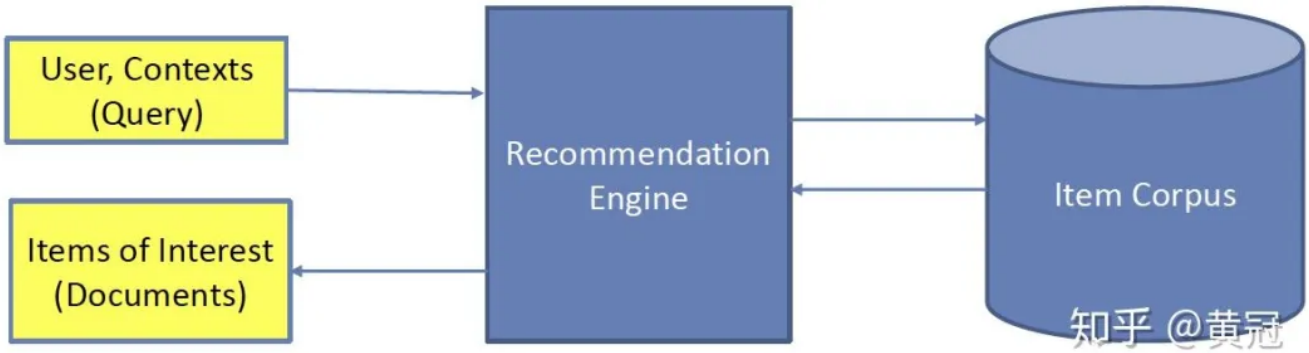
搜索和推荐本身差异不大，但是搜索是一个用户主动输入query的过程，用query明确的表达自己需求，然后从搜索引擎的网页库中获取自己想要的信息的过程。





- **推荐**

推荐一般是非主动触发的，推荐的用户主动交互较少，用户的兴趣由他过去的行为、用户的一些属性以及当前的上下文来隐式地表达，推荐的实体丰富多样。推荐里的两大实体：user和item，他们是不同类型的东西，不像搜索里的query和doc一般都是文本。

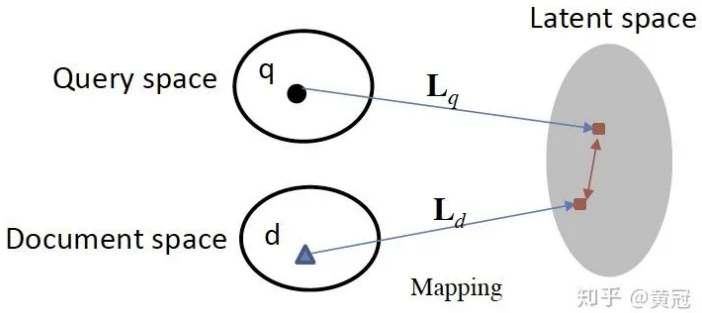


大部分的机器学习所应用的系统，例如搜索、推荐、广告，其实都可以分为召回和排序两个阶段，召回是一个拉取候选集的过程，往往就是一个匹配问题，而且很多匹配特征会是排序阶段的重要依据。再进一步说，搜索、推荐、广告本身其实就是一个匹配问题。

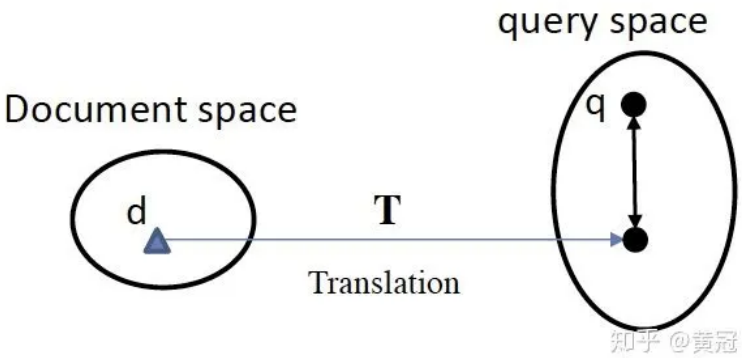
1.2 搜索中的传统匹配模型

举几个例子

- tf-idf
- bm25
- 隐式模型：一般是将query、title都映射到同一个空间的向量，然后用向量的距离或者相似度作为匹配分，例如使用主题模型：



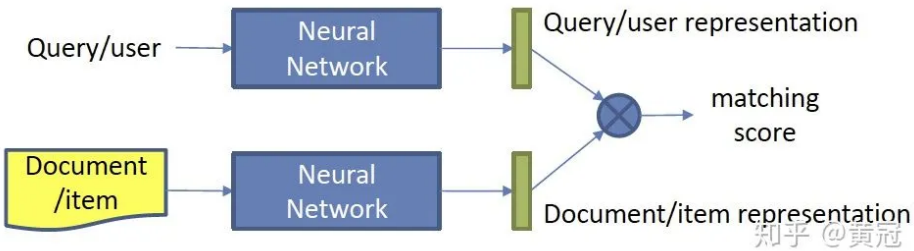
- 翻译、转换模型：将doc映射到query空间，然后做匹配；或者计算将doc翻译成query的概率(同语言的翻译问题)。



搜索里的深度学习语义匹配的方法分类可以分为以下几类：

- **representation-based**

这类方法是先分别是学习出query和doc的语义向量表示，然后用这两个向量做简单的计算如cosine/MLP，重点是**学习语义表示(representation learning)。**



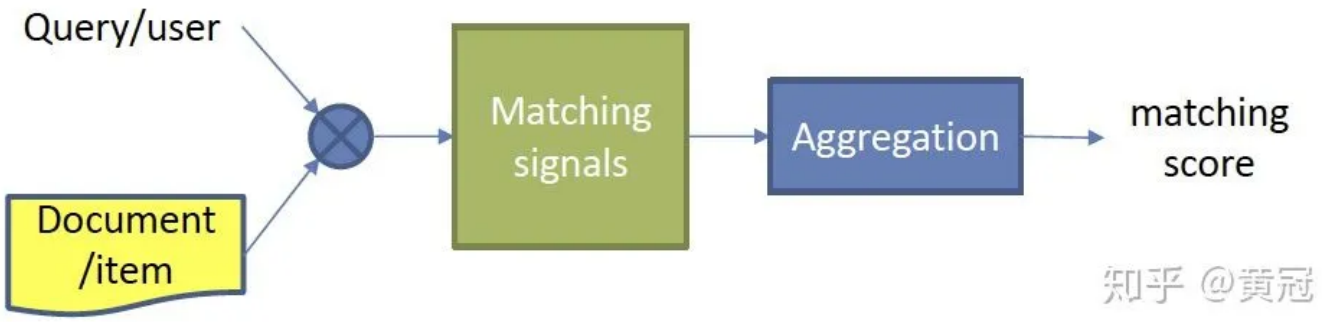
这种方法的模型分两个步骤：

- Step 1: calculate representation $\phi(x)$
- Step 2: conduct matching $F(\phi(x), \phi(y))$

这种方法和双塔模型类似，一般算出query和doc的向量后，直接就cosine或者内积，一般不是接MLP的。

- **interaction based**

这种方法关键在于在底层，让query和doc提前交互，建立一些基础的匹配信号，例如term和term层面的匹配，再想办法把这些基础的匹配信号抽取融合成一个匹配分。



这种方法也分为两个步骤：

- Step 1: construct basic low-level matching signals
- Step 2: aggregate matching patterns

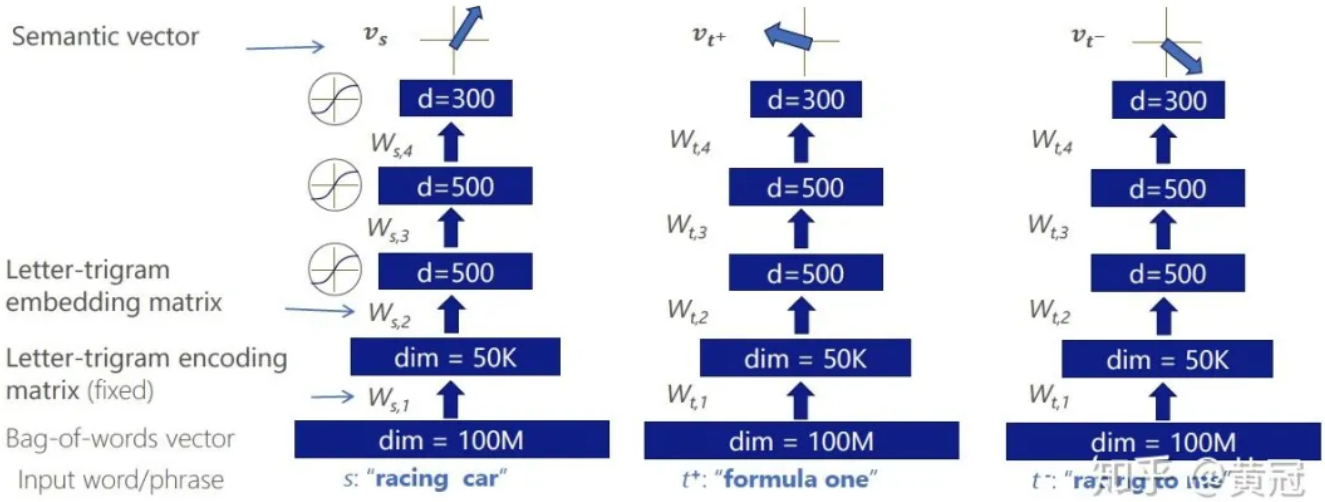
1.3 Representation based的一些方法介绍

- **Deep Structured Semantic Model (DSSM)**

语义匹配的开山之作。这个模型，用MLP来分别学习query和doc的语义向量表示：

- 首先，这是模型使用letter-trigrams，即letter级别的trigram来表示词（即在每个单词前后插入一个#，然后使用每一个letter-trigrams来表示单词）：
  - “#candy# #store#” --> #ca can and ndy dy# #st sto tor ore re#
  - Representation: [0, 1, 0, 0, 1, 1, 0, ..., 1]
  - 将每一个letter-trigram映射成一个d维的向量
  - 将query内所有的letter-trigram的向量相加得到一个d维的向量（**bag of words，词袋模型**）；因为是相加，忽略了顺序，所以是词袋模型；相加还有一个作用是，将变长的向量转化为定长
  - 将doc内所有的letter-trigram的向量相加得到一个d维的向量（**bag of words，词袋模型**）
  - 后续继续接几个FC来分别学习query和doc的语义向量

- 然后用query和doc的语义向量的cosine表示它们的相似度打分



dssm的loss function：

point-wise的loss，例如log-loss，即和ctr点击率预估一样，相关的文档的label为1，不相关的文档的label为0

– A query q and a list of docs $D = \{d^+, d_1^-, \cdots, d_k^-\}$

– d^+ positive doc, $d_1^-, \cdots, d_k^-$ negative docs to query

– Objective:

$$P(d^+|q) = \frac{\exp(\gamma \cos(q, d^+))}{\sum_{d \in D} \exp(\gamma \cos(q, d))}$$

知乎 @黄冠

使用bag of letter-trigrams的好处：

- 减少字典的大小，#words 500K -> letter-trigram: 30K
- 减缓out of vocabulary的问题，提高了泛化性
- 对于拼写错误也有一定的鲁棒性

dssm的缺点：

- 词袋模型，失去了词序信息
- point-wise的loss，和排序任务match度不够高（排序任务一般是pairwise）。可以轻松的扩展到pair-wise的loss，这样和排序任务更相关。

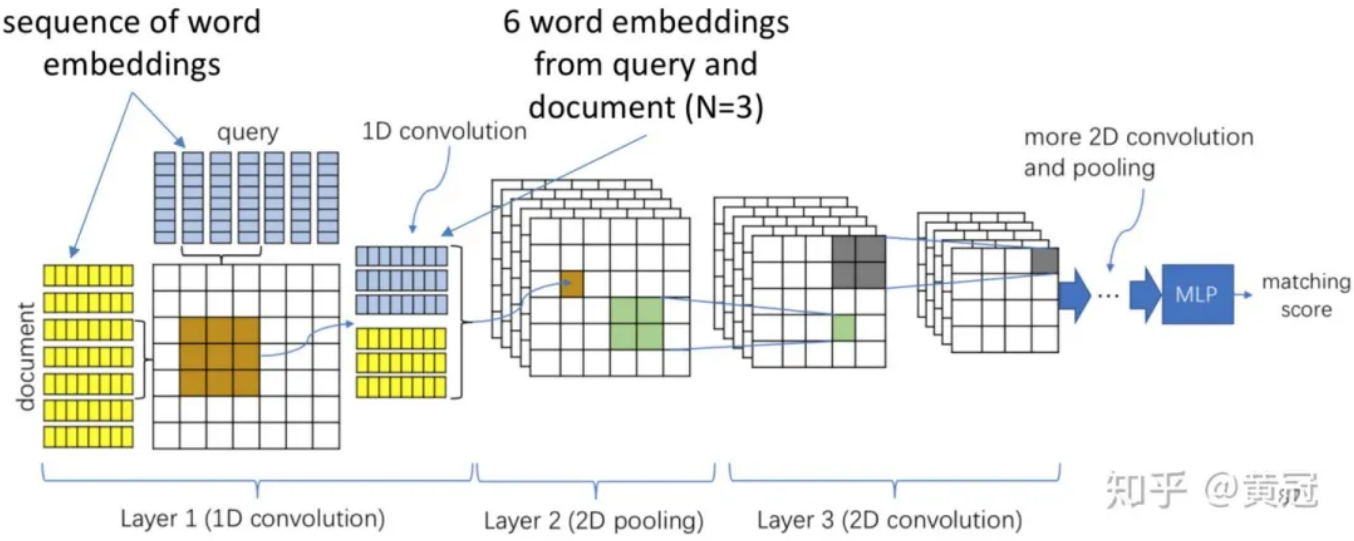
- **CNN**
  - **CNN对文本建模的常见操作，以CNN如何获得一个句子的向量为例：**

- 假设输入句子是sen=[A B C D E]，每个单词映射成一个100维的向量
- 假设卷积核窗口大小为k=3，那么每次对3个单词的向量（concat，得到3*100的一个300维的向量）进行卷积，卷积结果得到一个值，类似以下过程：
  - out_1 = w_1*x_1 + w_2*x_2+.....w_300*x_300
- 那么依次对ABC、BCD、CDE进行卷积，就会得到三个值：out_1、out_2、out_3
- 对out_1、out_2、out3取最大值， 即所谓的max-pooling，就会得到一个值，不妨叫max_out_1
- 上面是只有一个卷积核的情况，如果我们使用128个卷积核，就会得到128个值，这128个值就可以当成句子的向量

以上的操作中，卷积的时候，就类似抓取trigram信息，这个过程是保持局部的词序信息的（局部统筹）。

1.4 Interaction based的一些方法介绍

- **ARC-II (Hu et al., NIPS '14)** hangli-hl.com/uploads/3 ：
  - 让两个句子在得到它们各自的句子向量表示之前，提前交互，使用1D conv:
    - 例如窗口大小为N=3，那么每次从两个句子中分别取1个trigram，然后把两个trigram的向量concat起来，然后使用一个卷积核进行卷积得到一个值
    - 那么每两个trigram进行卷积，就会得到一个矩阵，这个矩阵是两个句子的基础的匹配信号，这个矩阵类似图像，是个2维的向量。
    - 使用多个卷积核，就会得到多个矩阵，即tensor
  - 得到多个有关两个句子的基础匹配信号的矩阵后，就可以像处理图像一样，处理每一个矩阵。常见的操作就是使用2D的卷积核。不断的卷积，就会得到一个定长的向量，然后再接MLP，最后一层的输出节点数目只有1，就得到了它们的匹配分。



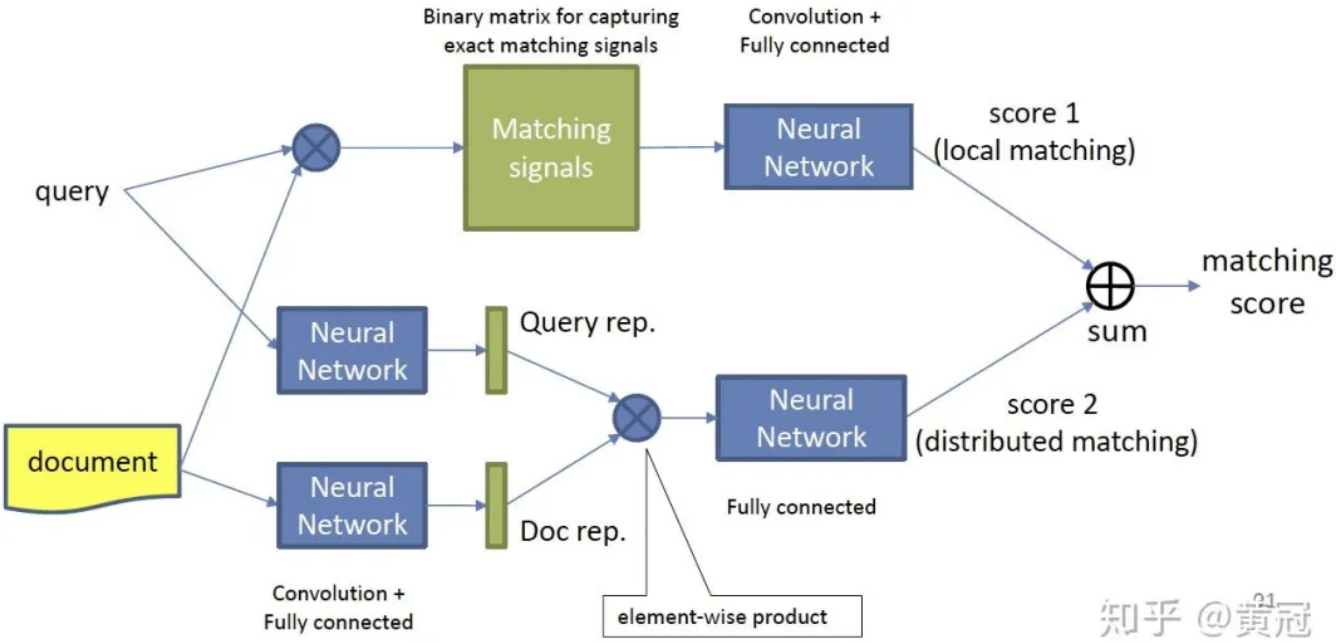
这个模型的优缺点：

- 优点是，有基础的匹配信号矩阵，可解释性强，而且卷积操作是保留次序信息的
- 缺点是，基于trigram的匹配，没有unigram的匹配信号，不过这个稍微改一下就可以了；另外没有特别区分精确匹配（sim=1.0）和普通匹配信号(sim<1.0)

同理，这里匹配矩阵是保留词序信息的。

1.5 representation-based 和interaction-based方法的融合

- **Duet, Mitra et al., WWW '17**microsoft.com/en-us/res：



这篇论文将interaction based的模型和representation based的模型融合。这篇论文指出，interaction based的模型和representation based的模型在处理不同的类似的query，各有千秋。总体上，interaction based的方案对长的、冷门的query更有效，representation based的模型对短的、热门的query更有效。这点经验仅供参考。

1.6 Representation based和Interaction based的效果、优缺点

	Method	P@1	MRR
Traditional IR	BM25	0.579	0.457
Representation Learning methods	ARC-I	0.581	0.756
	CNTN	0.626	0.781
	LSTM-RNN	0.690	0.822
Matching Function Learning	ARC-II	0.591	0.765
	MatchPyramid	0.764	0.867
	Match-SRNN	0.790	0.882

Based on Yahoo! Answers dataset (60,564 question-answer pairs)

效果上是interaction based好一些，但是representation based的方法，可以提前把doc的embedding计算出来，建网页库的时候就存进去，不用实时计算。

总结一下：

- Representation based：
  - 重点是学习文本的句子表示；可以提前把文本的语义向量计算好，在线预测时，不用实时计算。
  - 在学习出句子向量之前，两者没有任何交互，细粒度的匹配信号丢失。学习出来的向量可能是两个不同向量空间的东西，通过上层的融合层和loss，强制性的拉近两个向量。
- interaction based：
  - 有细粒度、精细化的匹配信号，上层进行更大粒度的匹配模式的提取；可解释性好
  - 在线计算代价大。

1.7 总结

搜索着重解决相关性、权威性、时效性等问题，深度学习匹配是解决相关性的大杀器。模型结构上，以下结构都值得尝试：

- bow
- CNN
- RNN、双向RNN、LSTM、GRU
- attention(self-attention，co-attention)
- match-matrix
  - match matrix其实是将attention的权重展开，attention是将权重用来加权求和表示，这两种方法可以配合使用，例如用attention的表示来计算match matrix等。

PLSA同样可以做到语义层面的匹配，但是深度学习可以基于有监督数据进行训练，label可以和直接的任务相关，而不是泛泛意义上的匹配。DNN使用语义平滑的word embedding，可以大大提高模型的泛化性。模型方法论上，深度学习匹配可以分为representation based的和interaction based的，两种学习范式是可以融合使用的。而具体到将匹配技术应用到搜索里的相关性问题，又可以分为based on Global Distribution of Matching Strengths的和based on Local Context of Matched Terms的。最后，安利一下这篇reference 百度NLP | 神经网络语义匹配技术 ，将这篇文章和本文的tutorial里的细节都看懂，相信你的功力会大增。

最新的attention也能用于匹配，介绍如下?

- **attention**

首先简单的介绍一下attention，attention一般我们分为self-attention和co-attention。self-attention就是自己和自己做attention，在匹配任务里对应representation learning的方法。co-attention是和对方做attention，例如query的每个term去和doc做attention，在匹配任务里对应interaction的方法。

- **transformer**

transformer是17年提出的翻译模型，其基本结构是深层的self-attention，里面有一些精细化的设计，可以参考 zhuanlan.zhihu.com/p/34 。

上面提的self-attention和co-attention的区别，可以从transformer里提的 attention的k、q、v来自哪里去理解。

- **bert**

bert是2018年谷歌提出的模型 BERT 模型详细解读， **其基本结构是transfomer**，应用到语义匹配，主要也有两种方式：

a. representation learning，即双塔的方式， 把预训练好的Bert当成一个编码器，把query输进去得到query的向量，把doc输进去得到doc的向量，然后接一个简单的match func(cosine 或者MLP)。

b. interaction based，把query和title拼接，输进去Bert，类似Bert里的NSP任务那样，然后由最后一层的表示做pooling，或者由cls的表示，然后接MLP得到匹配分。这种方式，里面做的attention既不是self-attention，也不是co-attention，因为：

i.self-attention: 只看到自己

ii. co-attention: 只看到自己的一个term和对方的所有词

iii.bert: 能看到 自己和对方所有的词

所以这种方式还是属于interaction based的方法。

用Bert来做匹配的好处是：a.基于字粒度，减少OOV（未登录词）的影响，减轻分词错误的风险，增强泛化能力。b.使用positional embedding建模顺序信号。c.深层transformer结构，有足够强的交互和建模能力。d.使用海量无监督数据预训练，在有监督样本少的任务场景就是神一般的存在。

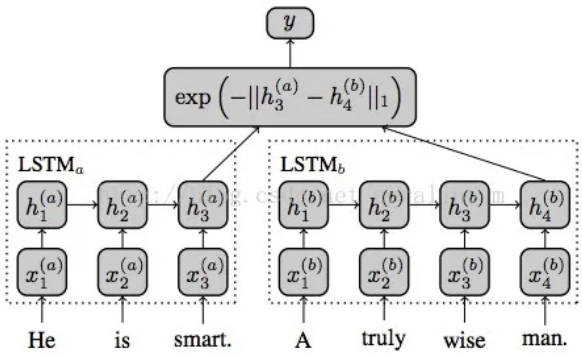
缺点是：a.模型结构复杂，预测性能是瓶颈。b.训练耗GPU。c.在有监督的训练数据本来就有海量的时候，预训练的方式优势不明显。

2-补充几个匹配模型细节

下面补充的几个是笔者曾经用过的，至于使用细节不再描述，主要讲下论文的说明。

2.1 孪生LSTM

论文题目是Siamese Recurrent Architectures for Learning Sentence Similarity，模型如下：



贡献点：1.提出新的度量方式（基于曼哈顿距离，见细节2）。优于欧几里得距离（梯度消失）、余弦相似度。【回归】 2.通过明确的指导（距离），使用简单LSTM能够建模复杂的语义。3.使用MaLSTM features输入给SVM来进行分类。【分类】

实验数据：

1.The SICK data set （10k条）：http://clic.cimec.unitn.it/composes/sick.html

- sentence_A: sentence A
- sentence_B: sentence B
- entailment_label: textual entailment gold label (NEUTRAL, ENTAILMENT, or CONTRADICTION)
- relatedness_score: semantic relatedness gold score (on a 1-5 continuous scale)

1.1 回归问题，得分在1-5

1.2 分类问题，三类【entailment, contradiction, or neutral】

细节：

- LSTM(a)和LSTM(b)权重设置一样(tied weights、主题一样)。在信息检索（IR）等其他应用场景可设置不一样(untied weights)。
- 度量方式使用基于曼哈顿距离d的dis=e^(-d),由于得分在1-5，因此做了dis*4.0+1.0的处理。简单的度量方式，让句子表示更好地表达复杂的语义关系。
- LOSS函数使用MSE。训练使用BPTT。
- 词向量预训练（实验数据只有10k条），利用同义词扩充来数据增强。
- input维度（300维）、hidden维度（50维）

2.2 孪生LSTM

论文题目：Learning Text Similarity with Siamese Recurrent Networks

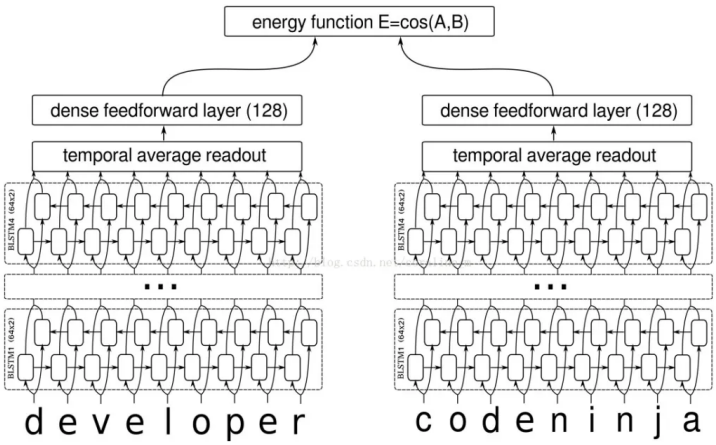


Figure 1: Overview of the Siamese Recurrent Network architecture used in this paper. The weights of all the layers are shared between the right and the left branch of the network.

贡献点： 1.语言规范化（Normalization）很重要，而规范化严重依赖于语义相似度。如（12pm，noon，12.00h）（李小龙，Bruce Lee，Lee Jun-fan）应当被归于相同的表示。present a system for job title normalization（论文阅读-文本匹配（一） 孪生LSTM是学术界研究，而这篇论文是工业界实用）2. w2v词嵌入取得不错的效果，但是使用字符级的可以更好地处理OOV问题。3.比较孪生结构（通过明确的相似性信息来学习不变性和选择性的表征）和自编码结构（增加噪声和降维来学习不变性）。4.传统job title normalization分类模型的缺点：数据标注昂贵；缺乏可控性（分类错误或新添加一条数据，模型需要重新训练）；不能够迁移学习（模型表示重用于不同的任务）。5. LOSS函数的创新。6.数据增强（Data Augmentation）【分为四种数据上看效果，详见实验数据第2点】

实验数据：1.19,927 job titles into 4,431 groups. 2.对数据的处理（Data Augmentation）2.1 拼写错误【10%的数据，随机取代20%字符，删除5%字符】2.2 同义词替换 2.3 多余的单词 2.4 随着知识增加，模型可修改。3. 长尾分布

细节：

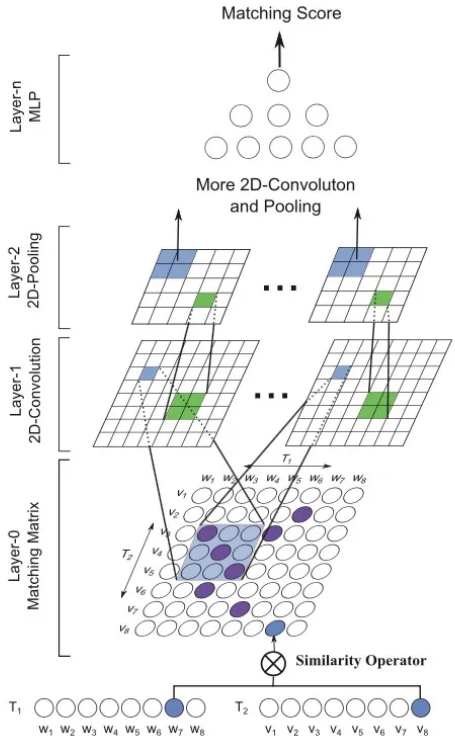
1.权重共享，度量方式使用余弦相似度。

2.字符个数padding到100，Adam优化，drop_out（recurrent: 0.2，between: 0.4）

3.正负比例1:4

2.3 MatchPyramid(构造匹配矩阵)

论文题目是Text Matching as Image Recognition



贡献点：1）受到CNN图像识别的启发（可以提取到边、角等特征），作者提出先将文本使用相似度计算构造相似度矩阵，然后卷积来提取特征。把文本匹配处理成图像识别。2）根据结果显示，在文本方面使用作者提出的方法可以提取n-gram、n-term特征。作者给了一个匹配的例子：

T1：Down the ages noodles and dumplings were famous Chinese food.

T2：Down the ages dumplings and noodles were popular in China.

PS：Down the ages是历代以来的意思。

模型可以学习到Down the ages（n-gram特征），noodles and dumplings与dumplings and noodles（打乱顺序的n-term特征）、were famous Chinese food和were popular in China（相似语义的n-term特征）。

作者基于此模型，在2017.3-2017.6 Kaggle的Quora Question Pairs 比赛上，取得了全球第四的好成绩。（YesOfCourse 团队）【效果很不错】

实验数据：

1.MSRP数据（判断两个短语是否有相同含义，4k训练，1.7k测试）

2.论文引用匹配（Paper Citation Matching）数据，约84w数据（28w正例，56w负例）。作者自己搜集，没有公开。

细节：1.Ai和Bj距离度量方式：完全一样（Indicator），余弦相似度（Cosine），点乘（Dot Product)。2.卷积，RELU激活，动态pooling（pooling size等于内容大小除以kernel大小）。3.卷积核第一层分别算，第二层求和算。可以见下图3*3的kernel分别算，2*4*4求和算。4.MLP拟合相似度，两层，使用sigmoid激活，最后使用softmax，交叉熵损失函数。5.Embedding长度较小的比较没用，用DOT比COS好。

2.4 k-nrm和Conv-knrm:

概述：构建文本与文本的相似矩阵,使用RBF Kernel进行 Kernel Pooling,取log相加后接一个全链接进行分类二分类（相似与不相似）。模型简称 K-NRM（Kernel-based Neural Ranking Model）。

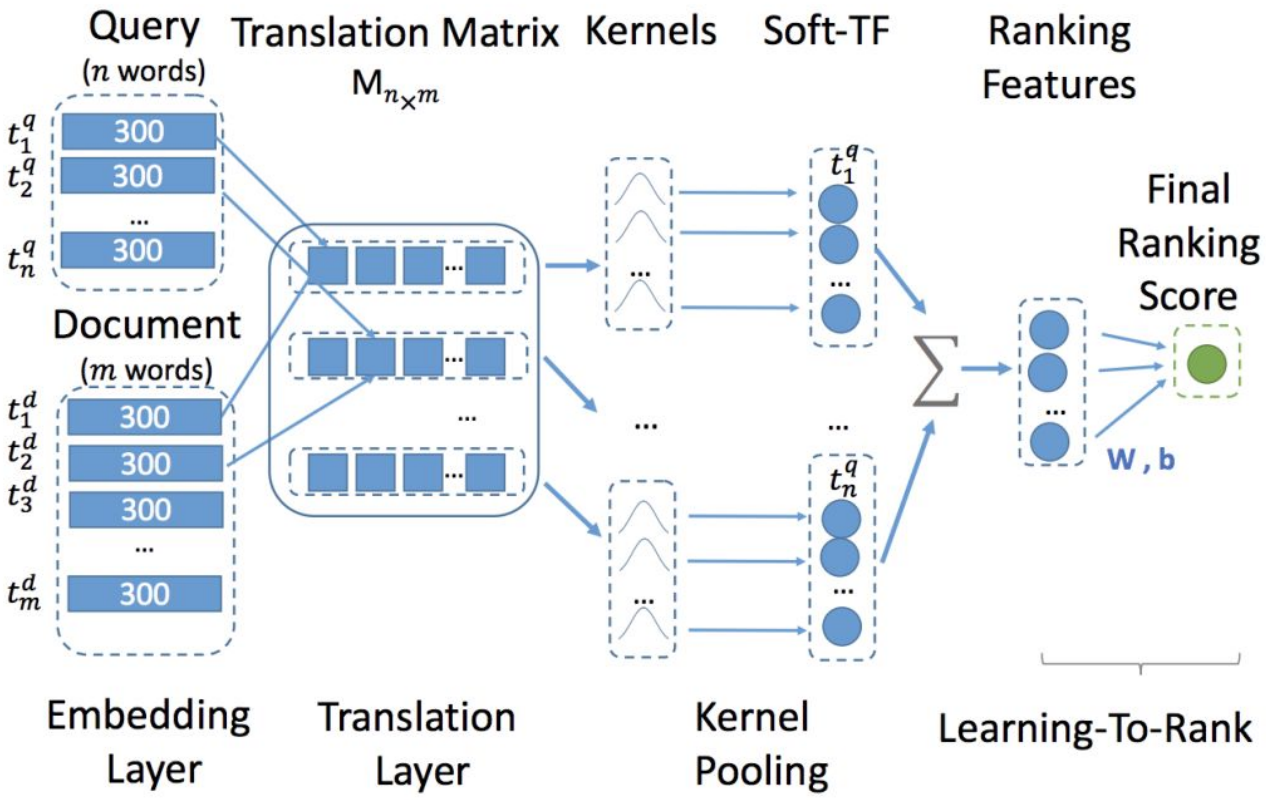


Figure 1: The Architecture of K-NRM. Given input query words and document words, the embedding layer maps them into distributed representations, the translation layer calculates the word-word similarities and forms the translation matrix, the kernel pooling layer generate soft-TF counts as ranking features, and the learning to rank layer combines the soft-TF to the final ranking score.

先把 term 映射为 Word Embedding：

$$t \Rightarrow \vec{v}_t$$

再计算Translation Matrix：

$$M_{ij} = cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d})$$

然后通过 RBF Kernel：

$$K_k(M_i) = \sum_j exp(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2})$$

得到 Kernel Pooling：

$$\vec{K}(M_i) = K_1(M_i), K_2(M_i), \ldots, K_K(M_i)$$

由 Kernel Pooling 得到 Sotf-TF Features：

$$\boldsymbol{\phi}(M) = \sum_{I=1}^{n} log(\vec{K}(M_i))$$

在用 Sotf-TF Features 做分类：

$$f(q, d) = tanh(w^T \boldsymbol{\phi}(M) + b)$$

采用 pairwise learning to rank loss 进行训练：

$$l(w, b, V) = \sum_q \sum_{d^+, d^- \in D_q^{+,-}} max(0, , 1 - f(q, d^+) + f(q, d^-))$$

使得相关的d^+排名比不相关的d^-更靠前，学习的参数为w、b和词向量V。

读后感：本文和 MatchPyramid 的核心的不同之处在于 RBF Kernel，感觉文章对高斯核函数的均值和方差怎么去取的没有写明白，只能看出来均值和方差是超参数，结合代码可以看出来，均值是从-1到1均匀均匀取值的（因为余弦相似度是在 -1 到 1 之间），方差也是取个定值，RBF Kernel得到的特征的含义是分布在均值周围的个数，可能也是 Sotf-TF 的含义（sotf term frequency）。但有个问题就是模型没有考虑文本的长度，所以我觉得他才选择 pairwise learning to rank loss 进行训练，这样对比的是相同 query 的得分，那么 query 的长度是固定的，document 一般都比较长，做截断后长度也应该相同，就解决了长度不同带来结果的不同了。

Conv-knrm相比k-nrm，最大的改变就是它添加了n-gram的卷积，增加了原先模型的层次，它能够捕捉更加细微的语义实体，交叉的粒度也更加细。

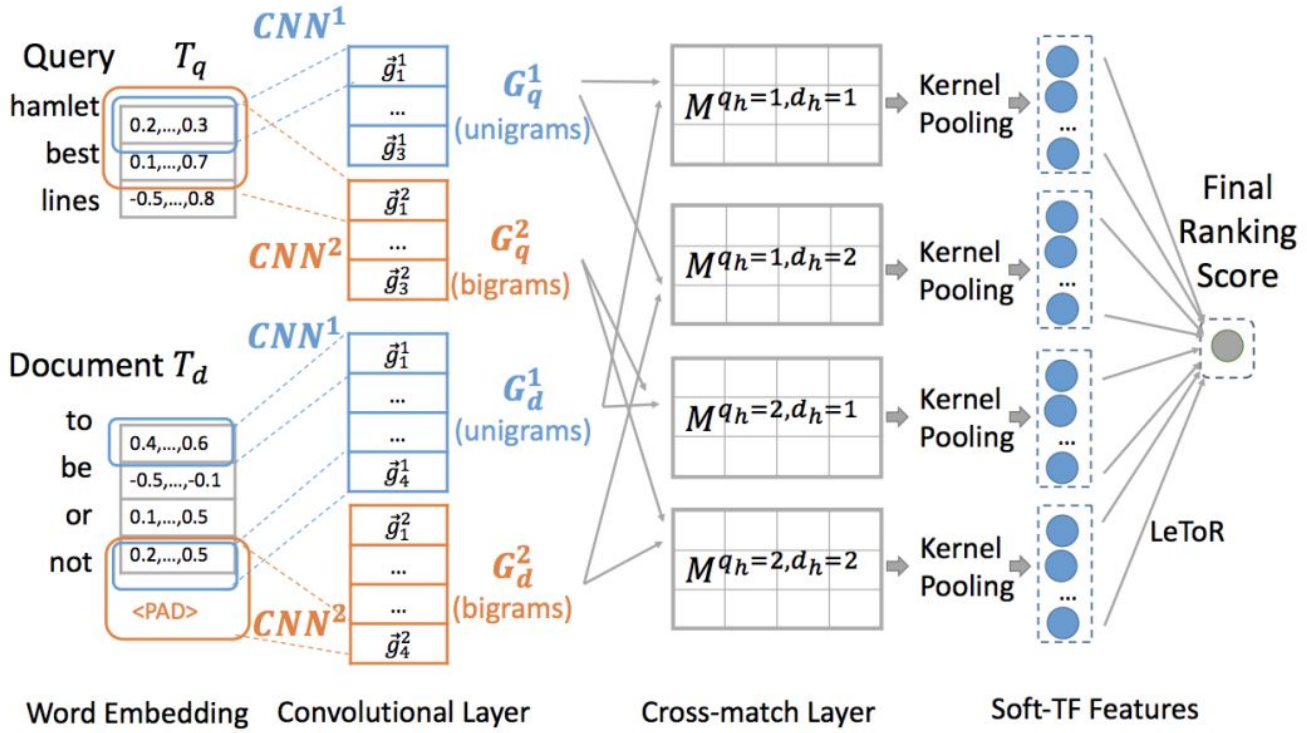


Figure 1: The Conv-KNRM Architecture. Given input query and document, the embedding layer maps their words into distributed representations, the convolutional layer generates n-gram embeddings ; the cross-match layer matches the query n-grams and document n-grams of different lengths, and forms the translation matrices; the kernel pooling layer generates soft-TF features and the learning-to-rank (LeToR) layer combines them to the ranking score. The case with Unigrams and Bigrams ($h_{max} = 2$) is shown.

过程也是比较简明的：

先把 term 映射为 L-dimensional 的 Word Embedding：

$$t \Rightarrow \vec{v}_t$$

对于有m个 term 的文本，可以得到一个$m \times L$的矩阵：

$$T = \begin{bmatrix} \vec{v}_1 \\ \dots \\ \vec{v}_m \end{bmatrix}$$

再用卷集计算 h-gram 的特征(向后补0)：

$$\vec{g}_i^h = relu(W^h \cdot T_{i:i+h} + b^h)$$

然后计算 cross-match：

$$M_{ij}^{h_q,h_d} = cos(\vec{g}_i^{h_q}, \vec{g}_i^{h_d})$$

和 k-nrm 一样，对每一个M_{ij}计算 Sotf-TF Features，再把所有的 Sotf-TF Features 拼起来得到$\phi(M)$，然后再做分类：

$$f(q,d) = tanh(w^T \phi(M) + b)$$

读后感：感觉上与 knrm 增加了 n-gram 的特性，但是感觉参数增加了很多，可能会很容易过拟合。

划重点

1.推荐和搜索的本质其实都是匹配，前者匹配用户和物品；后者匹配query和doc。具体到匹配方法，分为传统模型和深度模型两大类。

2.对于深度模型，主要分为基于representation learning的深度模型以及match function learning的深度模型。基于representation learning的深度模型学习的是用户和物品的表示，然后通过匹配函数来计算，这里重点在与representation learning阶段，可以通过CNN网络，auto-encoder, 知识图谱等模型结构来学习。对于match function learning的深度模型，也分为基于协同过滤的模型和基于特征的模型。对于match function learning另一种模型框架，是基于特征层面的，

有基于fm模型的，基于attention的，以及高阶特征捕捉的，另外还有基于时间序列的文章中只提到了DIEN模型。

参考资料：

1）[论文]深度相关性匹配模型-DRMM

https://zhuanlan.zhihu.com/p/94195125

2）检索式问答系统的语义匹配模型（神经网络篇）

https://zhuanlan.zhihu.com/p/67132780

3）推荐系统中的深度匹配模型