知乎

《搜索与推荐中的深度学习匹配》之搜索篇

黄冠
五月天歌迷；机器学习算法工程师

关注他

623 人赞同了该文章

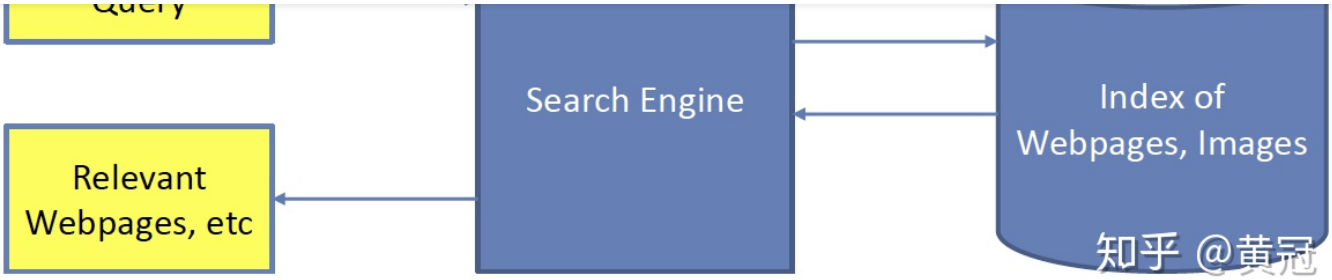
讲真，很久没看过能让我这么兴奋的资料了，这个tutorial 链接: pan.baidu.com/s/1UaHiZb... 提取码: itz8 ，简直就像一个博士论文，能让我对这个方向有足够深入的了解。而我最近又恰好从事这个方向，恰好也是落地到搜索引擎和推荐系统中，刚看到这个tutorial的时候，简直开心得不要不要的。

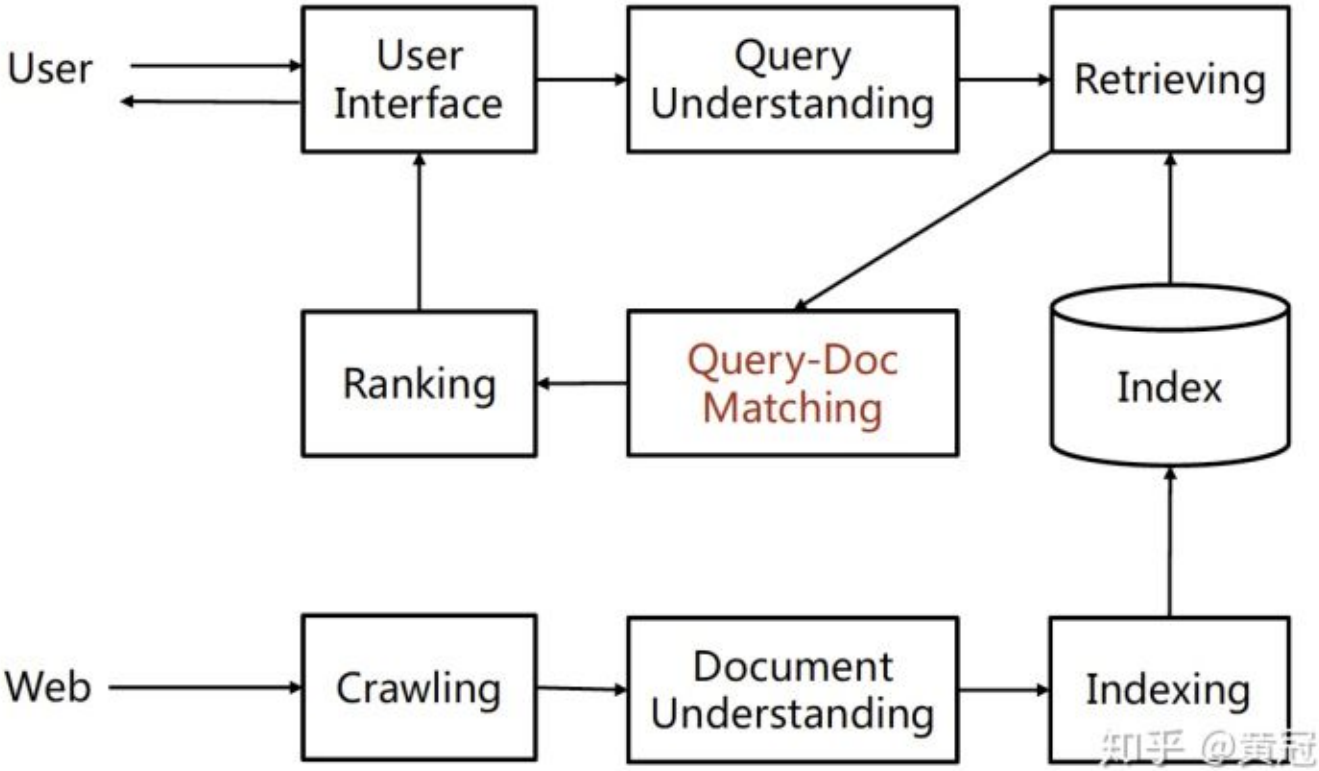
本篇blog的纲要：

part-1 搜索和推荐的概述

- 搜索

搜索是一个用户主动输入query，用query比较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。

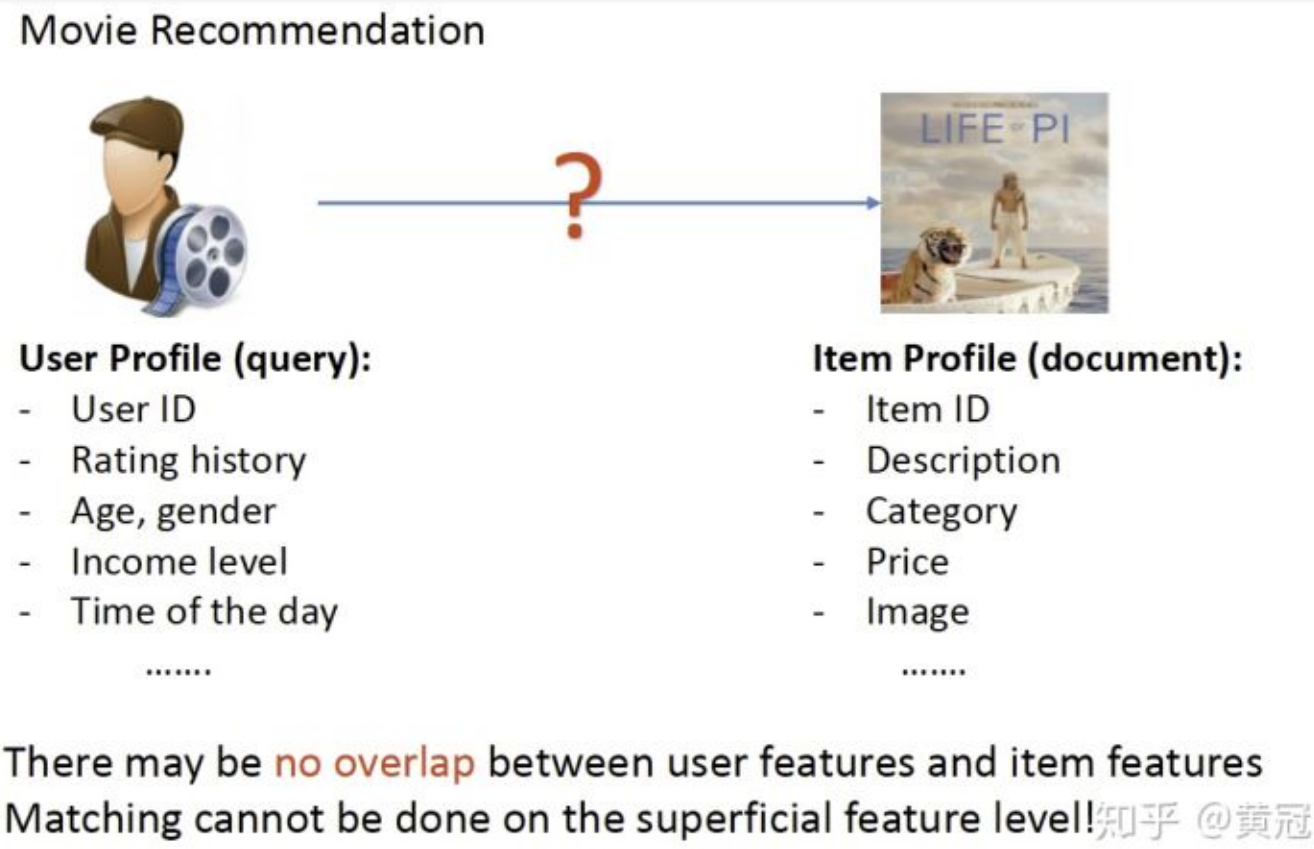




- 推荐

推荐一般是非主动触发的，一般没有用户输入的query，用户的兴趣由他过去的行为、用户的一些属性（例如职业、年龄、性别等）、以及当前的上下文来隐式地表达，推荐的实体可以是电商的宝贝、电影、标签、好友等。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。

知乎



part-2 什么是语义匹配

例如在搜索中，同样想知道iPhone手机的价格，两个query:“iphone多少钱”和“苹果手机什么价格”，这两个query的意思是完全一样的，但是字面上没有任何的重叠，用bm25和tf-idf来计算，他们的相似度都是0。语义匹配就是要解决这种字面上不一定重叠，但是语义层面是相似的文本匹配问题。

part-3 匹配问题的重要性

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：

- 搜索：query vs doc 的匹配
- 广告：query vs ad 的匹配 或者 user vs ad 的匹配
- 推荐：user vs item 的匹配

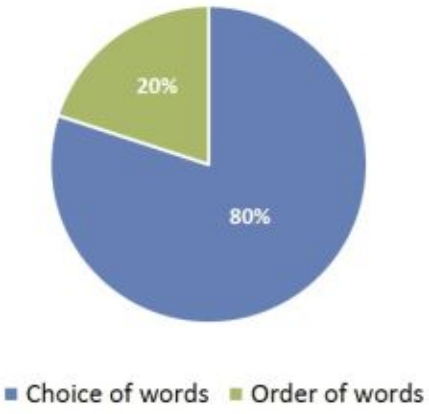
- query: Down the ages noodles and dumplings were famous Chinese food
- doc: ... down the ages dumplings and noodles were popular in China ...

那么要做好query和doc的匹配任务，要从以下几个点入手：

- 建立词和词之间的语义匹配：
  - 精确匹配的词： Down~down，dumplings～dumplings
  - 语义相似的词：famous ~ popular, Chinese ~ China
- 识别词序信息和大粒度匹配：
  - N-grams: “down the ages” ~ “down the ages”
  - N-terms: “noodles and dumplings” ~ “dumplings and noodles”

那么到底是词的出现信息更重要，还是词的顺序更重要呢？slide中很好的说明了这个问题，80%由出现信息决定，20%由词的顺序决定：

- Assume:
  - Size of vocabulary = 10,000
  - Average sentence length = 20
- Rough contributions of information in bits
  - From the selection of words: $\log_2(10000^{20})$
  - From the order of words: $\log_2(20!)$
- “Over 80% of the potential information in language being in the choice of words without regard to the order in which they appear ”
  - 80%: choice of words
  - 20%: order of words

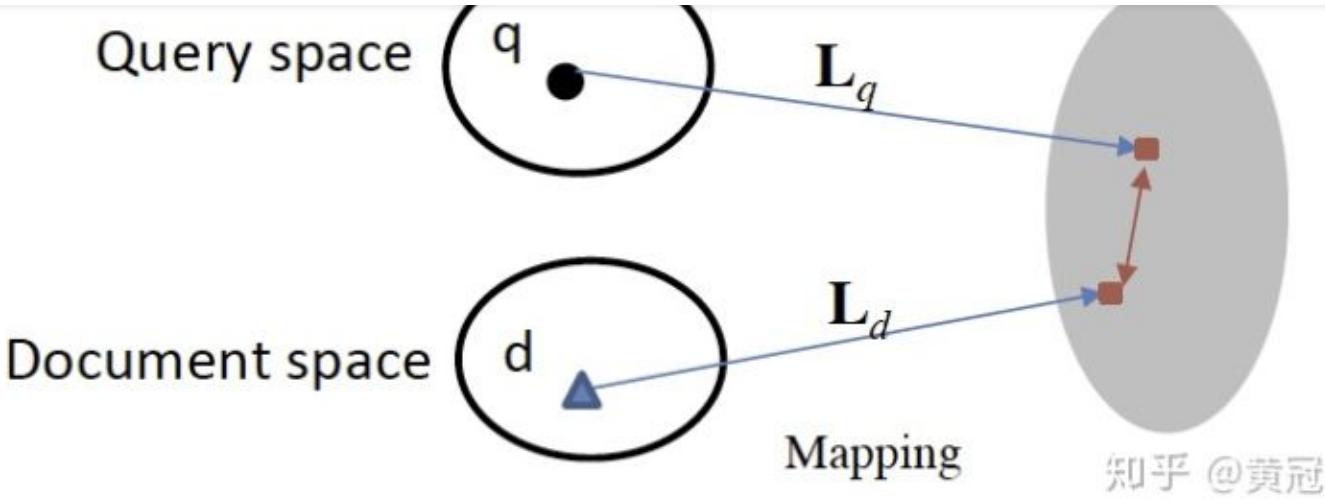


part-5 搜索中的传统匹配模型

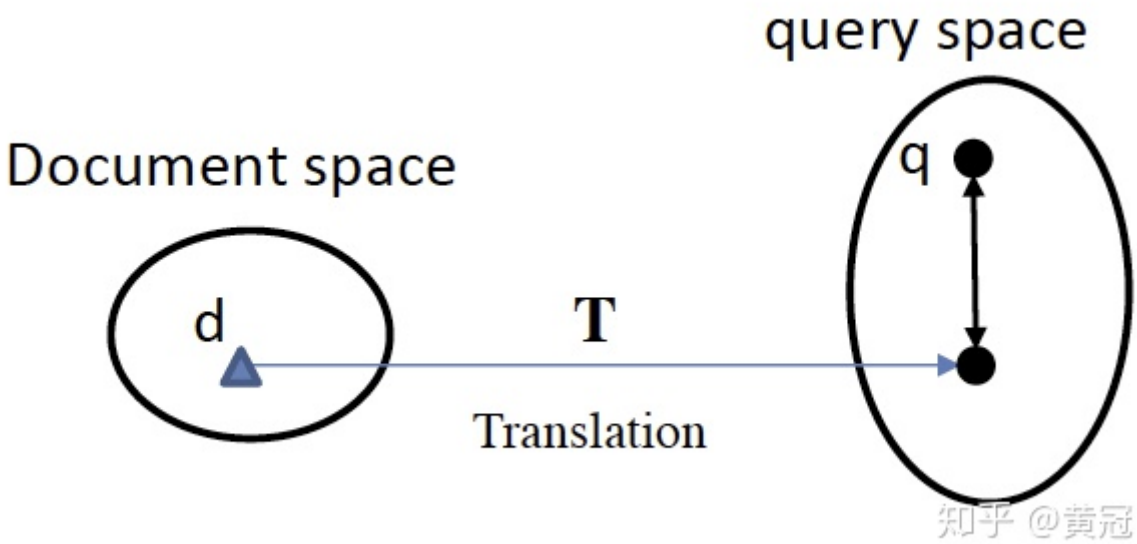
这里不详细展开，但举几个例子

- 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

知乎



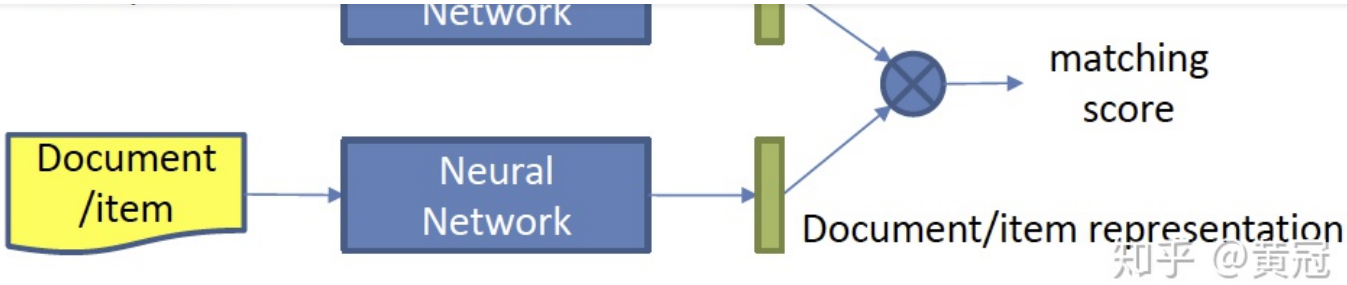
- 翻译、转换模型：将doc映射到query空间，然后做匹配；或者计算将doc翻译成query的概率(同语言的翻译问题)。



part-6 搜索里的深度学习语义匹配的方法分类

- **representation-based**

这类方法是先分别是学习出query和doc的语义向量表示，然后用两个向量做简单的cosine（或者接MLP也可以），重点是**学习语义表示(representation learning)。**



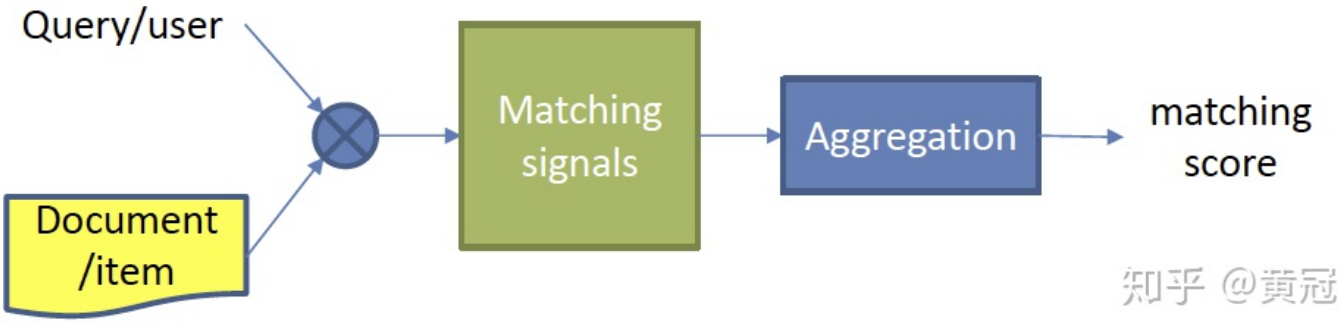
这种方法的模型分两个步骤：

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

这种方法和市面上提的双塔模型差不多是一个意思， 只是市面上说的双塔的模型，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**

这种方法是不直接学习query和doc的语义表示向量，而是在底层，就让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：

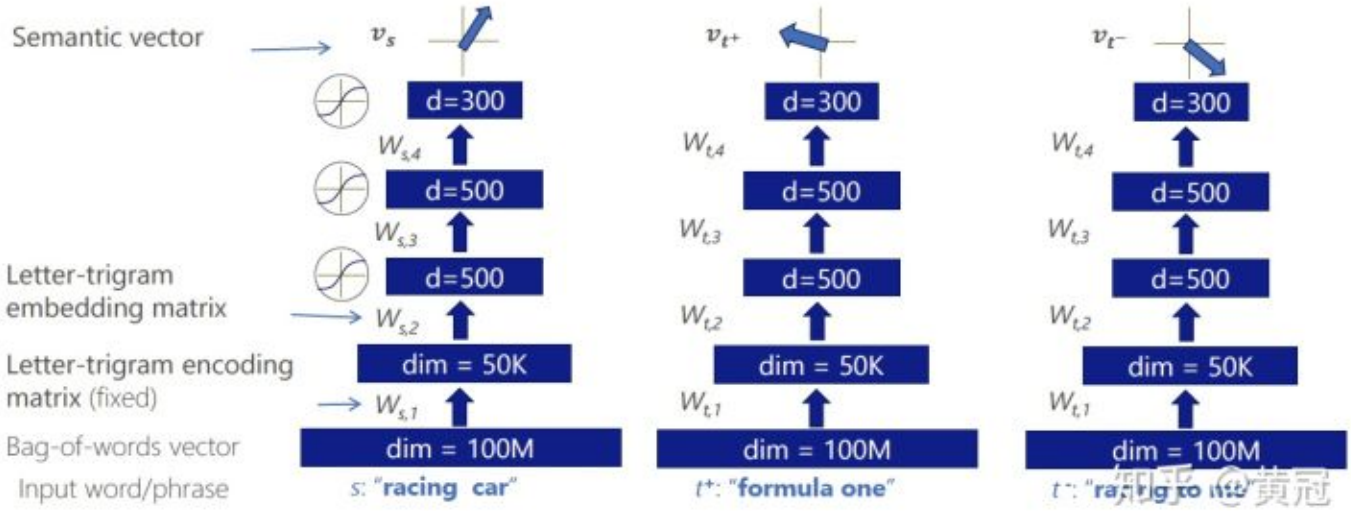
- Step 1: construct basic low-level matching signals
- Step 2: aggregate matching patterns

知乎

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

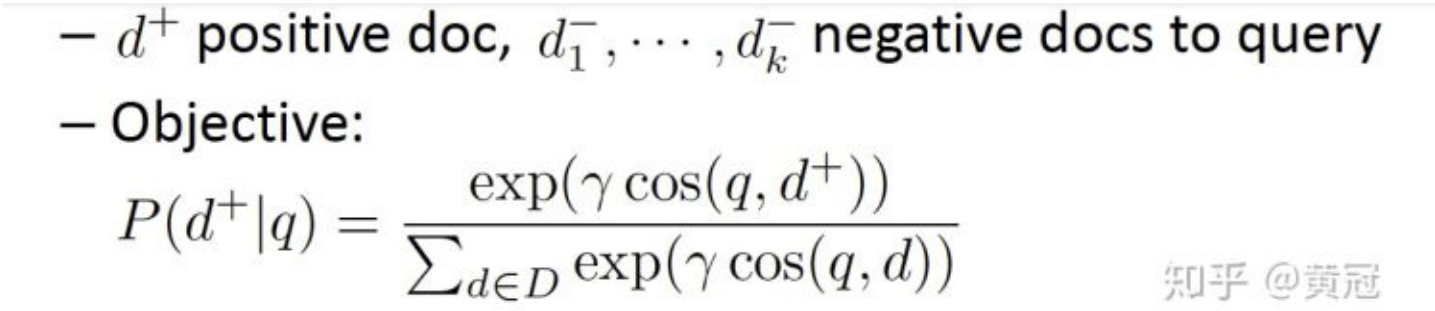
语义匹配的开山之作。这个模型，用MLP来分别学习query和doc的语义向量表示：

- 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



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

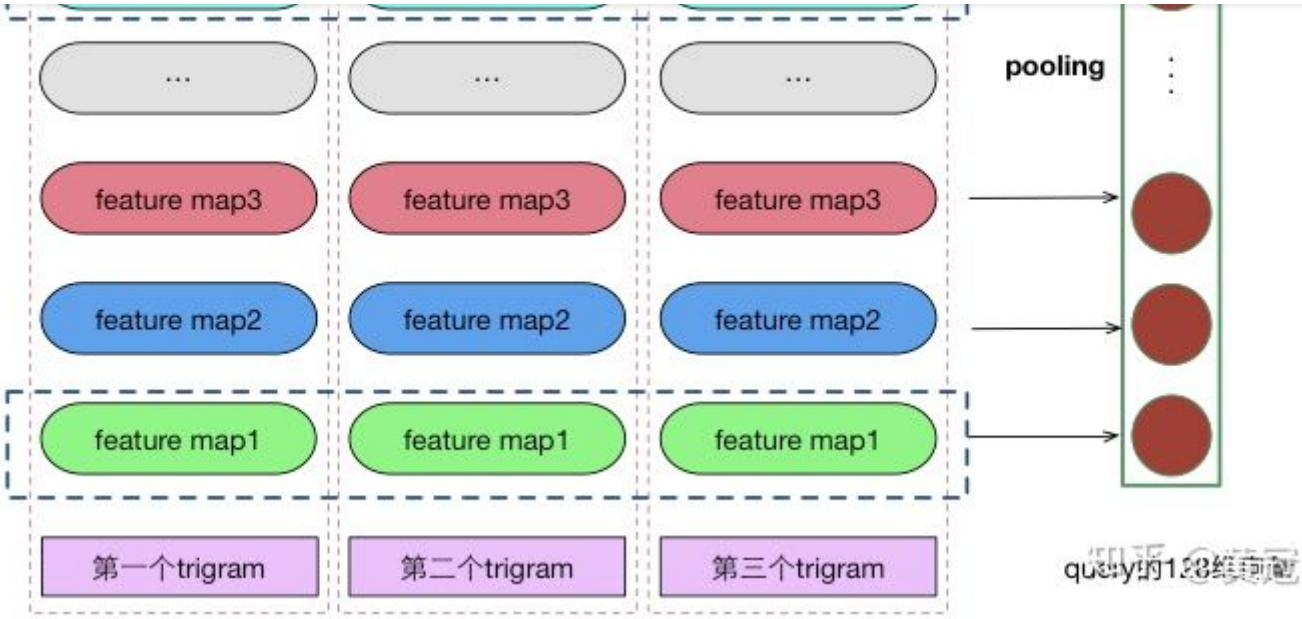
dssm的缺点：

- 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。

那么怎么获取词序信息呢？答案是可以使用CNN和RNN。

- **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

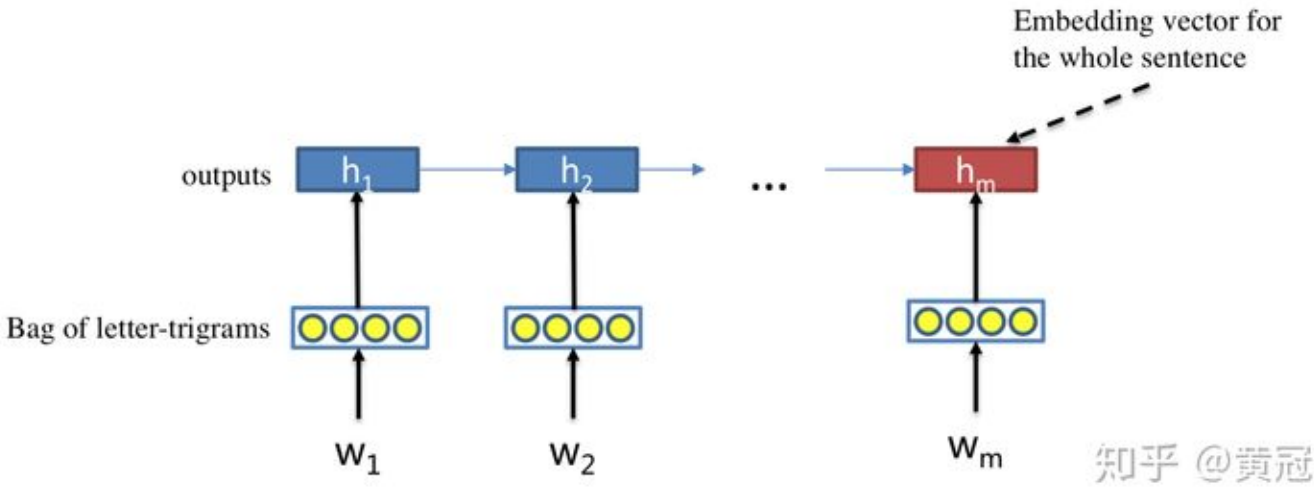
例如，我们要提取query的句子向量表示，可以表示为下图：



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）。但是max-pooling的时候，又把次序信息丢失了，max-pooling类似一个全局统筹的操作，抓取最强的语义信息。也可以用其它的pooling操作，可以部分的保留次序信息，这里先不展开。另外，pooling还有一个作用，可以将变长的东西，变成定长的，因为神经网络的神经元数目需要预先设定好，所以这个pooling的操作特别巧妙。但是max-pooling有个缺点是，例如query是ABCD，很有可能query是ABCDEFG的时候，max-pooling的结果不变，这个特性和很多匹配任务的场景是不一致的。

- **RNN**

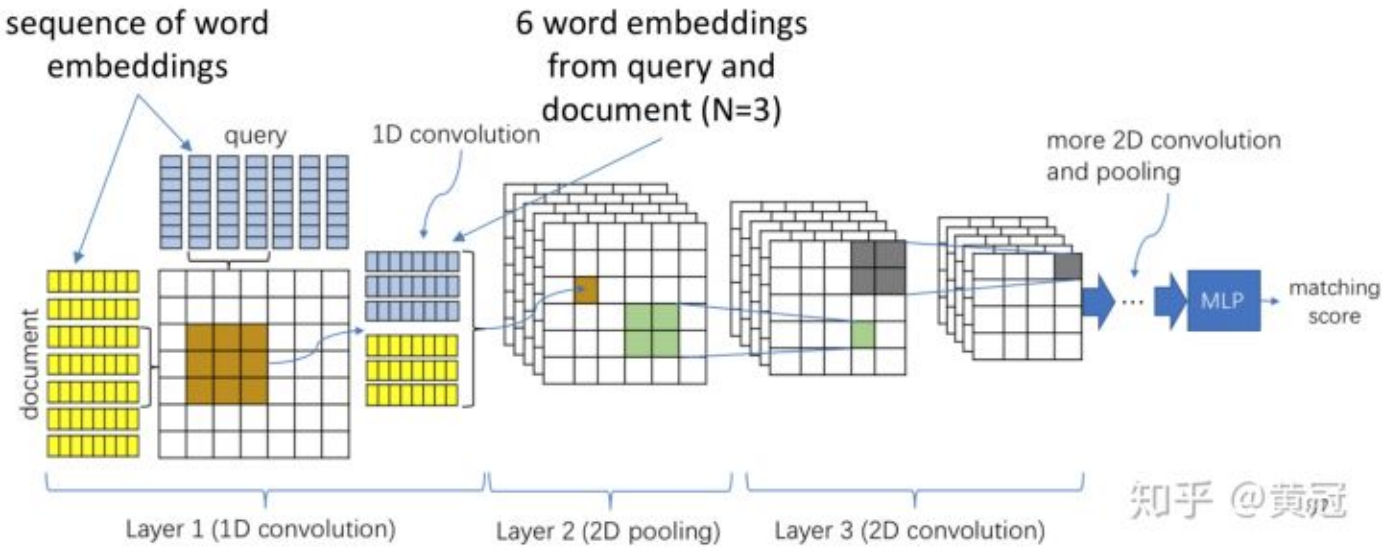
RNN可以很好的对序列进行建模，保留序列信息。每个timestamp喂入一个词的embedding，那么最后一个time_stamp的隐状态就可以当成整个句子的语义表示向量，然后正反向rnn一起用的话，效果有很好的提升：



- cosine
- 内积
- 接一个或者多个MLP，最后一层的输出层只有一个节点，那么最后一层的输出值就可以表示匹配分

part-6.2 Interaction based的一些方法介绍

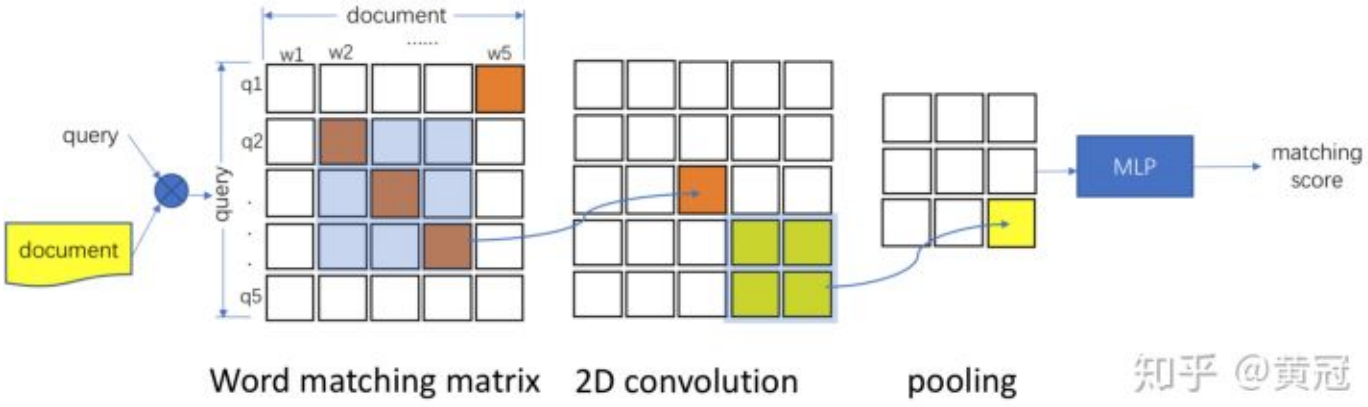
- **ARC-II (Hu et al., NIPS '14)** hangli-hl.com/uploads/3... :
  - 让两个句子在得到它们各自的句子向量表示之前，提前交互，使用1D conv:
    - 例如窗口大小为N=3，那么每次从两个句子中分别取1个trigram，然后把两个trigram的向量concat起来，然后使用一个卷积核进行卷积得到一个值
    - 那么每两个trigram进行卷积，就会得到一个矩阵，这个矩阵是两个句子的基础的匹配信号，这个矩阵类似图像，是个2维的向量。
    - 使用多个卷积核，就会得到多个矩阵，即tensor
  - 得到多个有关两个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)

- **MatchPyramid (Pang et al., AAAI '16)** cn.arxiv.org/pdf/1602.0...:
  和上文类似，只是这里不是trigram的基础匹配信号，而是使用unigram的匹配信号，即



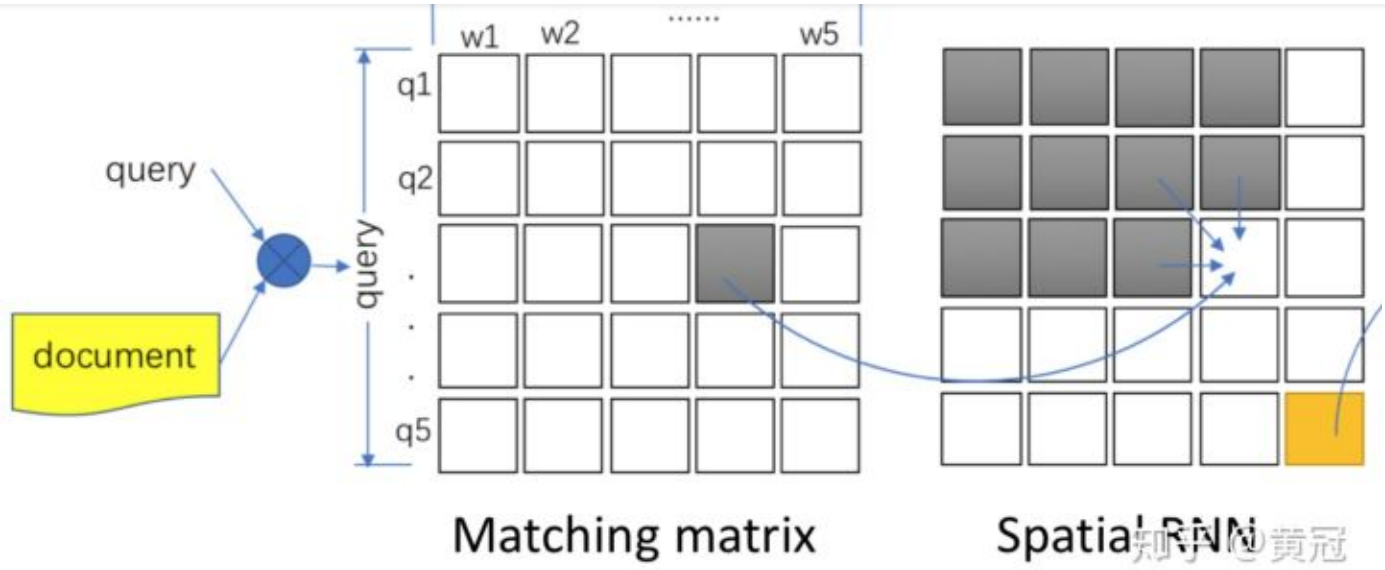
同理，这里匹配矩阵是保留词序信息的。

- **Match-SRNN (Wan et al., IJCAI '16)** arxiv.org/pdf/1604.0437...:
  - 这篇论文利用了和动态规划类似的思想。假设我们有了基础的word-level(或者说unigram粒度)的匹配信号矩阵后，怎么得到这两个句子最终的匹配分呢？假设query的长度是n，doc的长度是m，f[i][j]表示query的前i个词和doc的前j个词的匹配分，那么f[n][m]就是最终的query和doc的匹配分。那么怎么得到f[i][j]呢？

由动态规划的转移方程，我们可以得知，f[i][j]和以下信息有关：

- f[i-1][j]
- f[i][j-1]
- f[i-1][j-1]
- query[i]和doc[j]的相似度打分，query[i]表示query的第i个词，doc[j]表示doc的第j个词，不妨用s[i][j]表示这个打分。

那么f[i][j] = func(f[i-1][j], f[i][j-1], f[i-1][j-1], s[i][j])，论文中用一个2D的gru拟合这个func。



- **Decomposable Attention Model for Matching (Parikh et al., EMNLP '16)**aclweb.org/anthology/D1... ：
  - 这个论文的主要思路是使用attention：
    - 假设query的每个词对应的词向量是：query=[a1, a2, a3]，doc的每个词对应的词向量是：doc=[b1, b2, b3]
    - attention类似一种对齐机制，query的每个词都会去跟doc做attention，例如a1去跟doc去做attention，得到的向量表示是A1。那么query的每个词都去跟doc做一遍attention，就得到[A1,A2,A3]
    - 同理，doc的每个词都去跟query做一遍attention，就得到[B1, B2, B3]
    - 使用一个function，例如一个简单的神经网络，例如一层的FC作用到每个词和它的attention向量上，例如f(a1, A1) = X1，f(b1, B1) = Y1
    - 然后做一个简单的累加，v1=sum(X1, X2, X3)，v2=sum(Y1，Y2，Y3)
    - 然后再在v1、v2上，使用简单的FC层，就可以获得它们的相似度打分

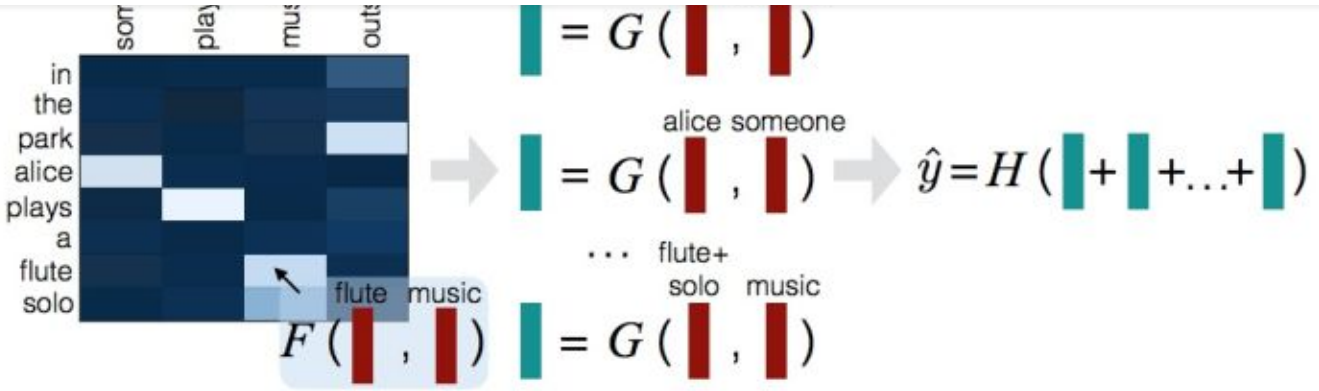
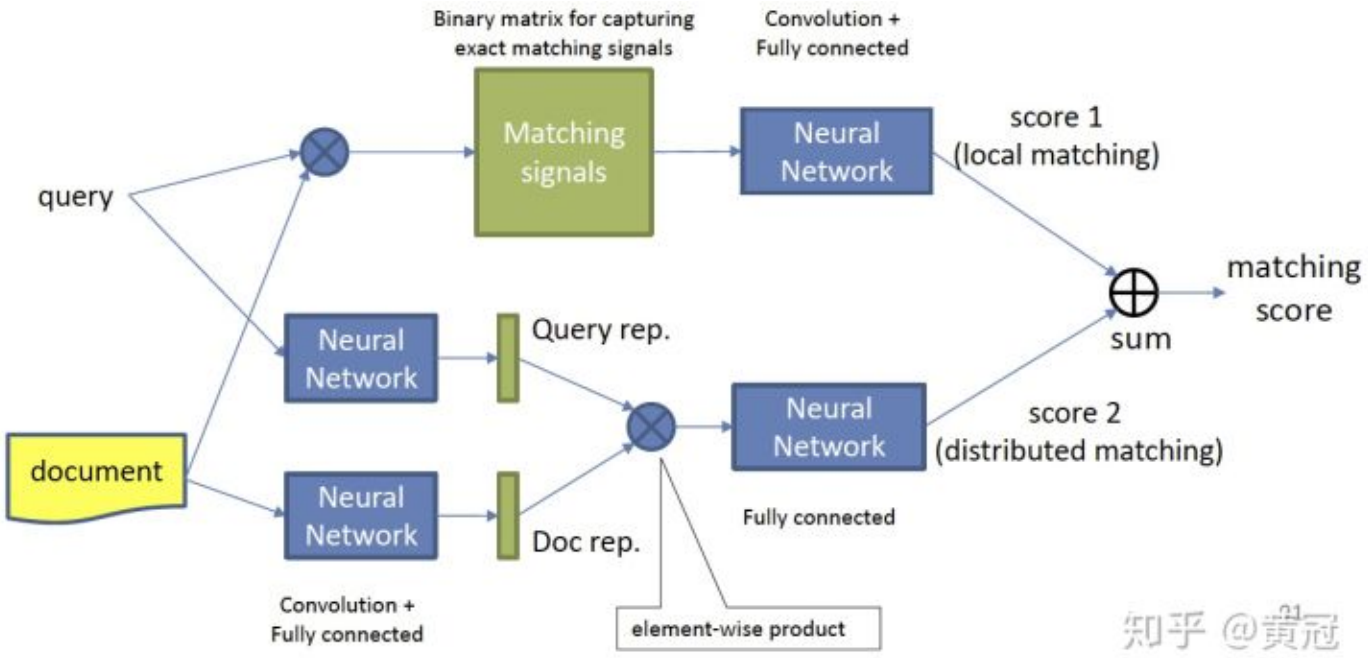


Figure 1: Pictoral overview of the approach, showing the *Attend* (left), *Compare* (center) and *Aggregate* (right) steps.

part-6.3 representation-based 和interaction-based方法的融合

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



这篇论文的卖点是，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。例如query "what channel are the seahawks on today"，包含这个问题的答案的网页才是比较好的结果，而不是命中“channel” or “today”这两个term的答案，这种

总体上，我在自己的工作中也发现，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。

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。

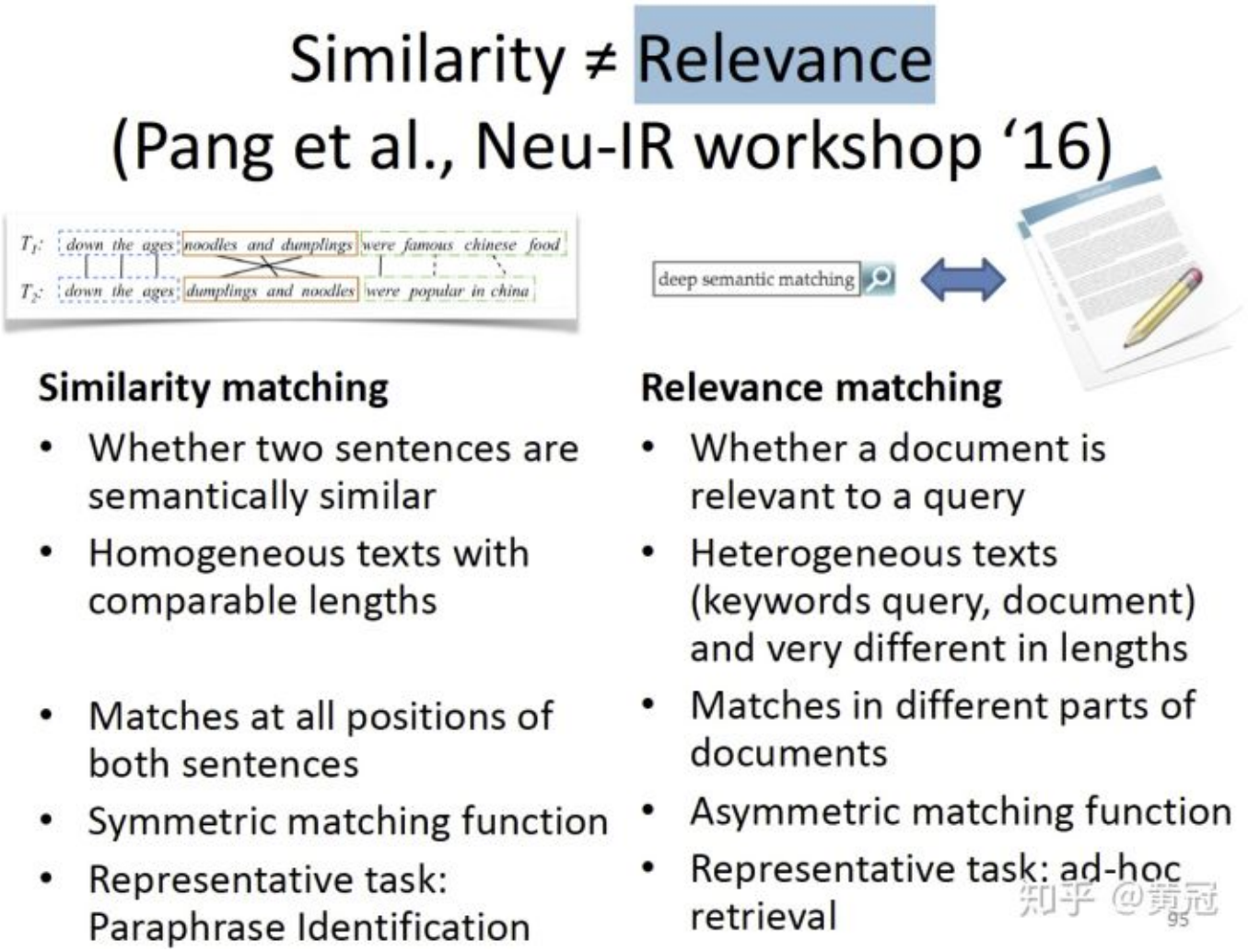
part-7 搜索里的query和doc的相关性匹配

这一part和前文有点跳跃性的转弯了，突然跳到一个相关性匹配（前文是语义匹配）的问题。但两者相关性高，语义匹配是相关性匹配的基础技术，例如，在工业界的搜索引擎中，网页doc往往可

知乎

首先，作者阐明相似度!=相关性，区别是：

- 相似性：
  - 判断两个句子语义、意思是否相似
  - 一般是同质的两段文本，例如两个句子、两个文章（长度接近，属性类似）
  - 在两段文本的不同位置进行匹配
  - 匹配函数是对称的，因为是同质的文本进行匹配
  - 代表性任务：同义句子识别
- 相关性：
  - 判断文档和搜索query是否相关
  - 不同质的两段文本，例如一个是query，一个是网页，长度不一样
  - 在网页的不同部分进行匹配，例如title、锚文本(链接对应的文本)、正文
  - 匹配函数不是对称的，因为不是同质的文本进行匹配
  - 代表性任务：query-网页检索



Query-Document Relevance Matching的方法也主要分为两大类：

知乎

part-7.1 based on Global Distribution of Matching Strengths的一些方法介绍

这种方法的基本步骤：

1.对于query中的每个term:

a.计算它在doc中的匹配信号

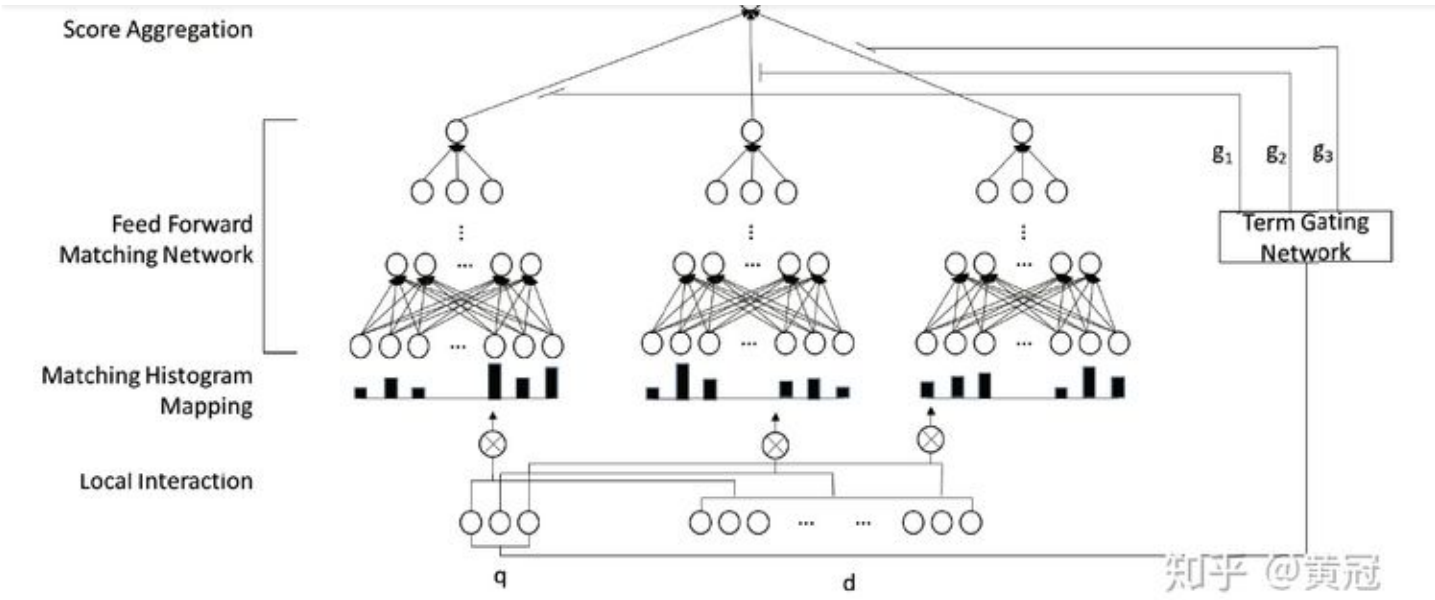
b.计算整体的匹配强度的分布

2.累计匹配强度的分布

- **Deep Relevance Matching Model (Guo et al., CIKM ’ 16)**bigdatalab.ac.cn/~gjf/p...
  - 这篇论文更像interaction-based的方法:
    - 和match matrix类似，使用cosine计算两个单词的相似度，这个模型直接用使用word2vec的词向量，不在模型中学习。这样可以使用少量的标注样本，专注于匹配函数的学习。而无监督的数据容易大量获得，所以学习word2vec是一件容易的事

对于query中的每个term：

- 将它和文档的所有单词的匹配分，离散化分桶。特别的是，为精确匹配单独划分一个桶。统计在每个桶上的次数，即得到一个关于这个term和文档匹配分的一个直方图，即一个向量。
- 得到上述向量后，使用全连接层学习匹配分。注意，不同的单词 ，这些全连接层的参数是共享的。
- 将上述的匹配分加权求和，这里的权重论文中也介绍了两者方法，其中一种是使用简单的IDF。



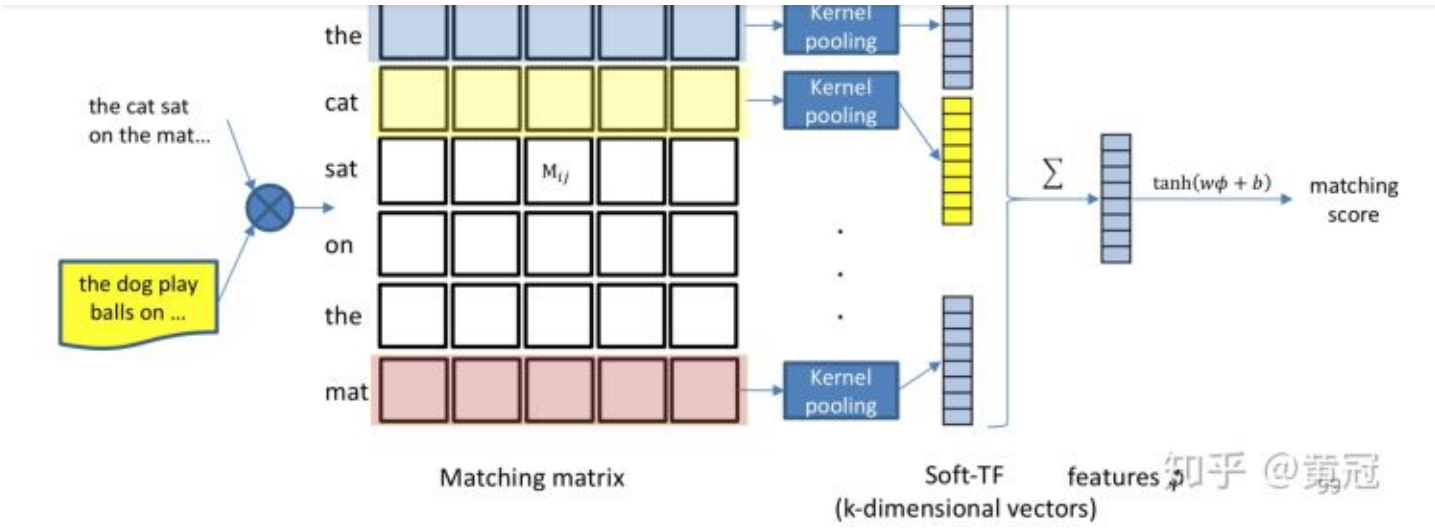
这个模型的优点是：

- 区分精确匹配和普通的相似度匹配信号
- 使用直方图，不用像卷积那样子使用padding
- 相比原始的匹配信号，直方分布图更鲁棒

缺点是：

- 失去了位置信息。但这篇论文要解决的是Ad-hoc Retrieval的问题，位置信息相对没那么重要。

- **K-NRM: Kernel Pooling as Matching Function (Xiong et al., SIGIR '17)**arxiv.org/pdf/1706.0661...
  - 这篇论文也是interaction-based的方式，步骤如下：
    - 先用cosine计算query和doc的match matrix
    - 对于match matrix的每一行，应用K个pooling-kernel，得到一个K维的向量
    - 然后将每一行的K的向量加和，得到一个K维的向量，然后后面就可以接FC层，得到匹配分



现在介绍一下这个pooling函数。首先，整个模型的流程用数学符号表示如下：

$$f(q,d) = \tanh(w^T \phi(M) + b) \quad \text{Learning to Rank} \quad (1)$$

$$\phi(M) = \sum_{i=1}^{n} \log \vec{K}(M_i) \quad \text{Soft-TF Features} \quad (2)$$

$$\vec{K}(M_i) = \{K_1(M_i), \ldots, K_K(M_i)\} \quad \text{Kernel Pooling} \quad (3)$$

$$K_k(M_i) = \sum_j \exp(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}) \quad \text{RBF Kernel} \quad (4)$$

$$M_{ij} = \cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d}) \quad \text{Translation Matrix} \quad (5)$$

$$t \Rightarrow \vec{v}_t . \quad \text{Word Embedding} \quad (6)$$

其中M表示匹配矩阵，M[i][j]表示匹配矩阵里的第i行的第j个元素。

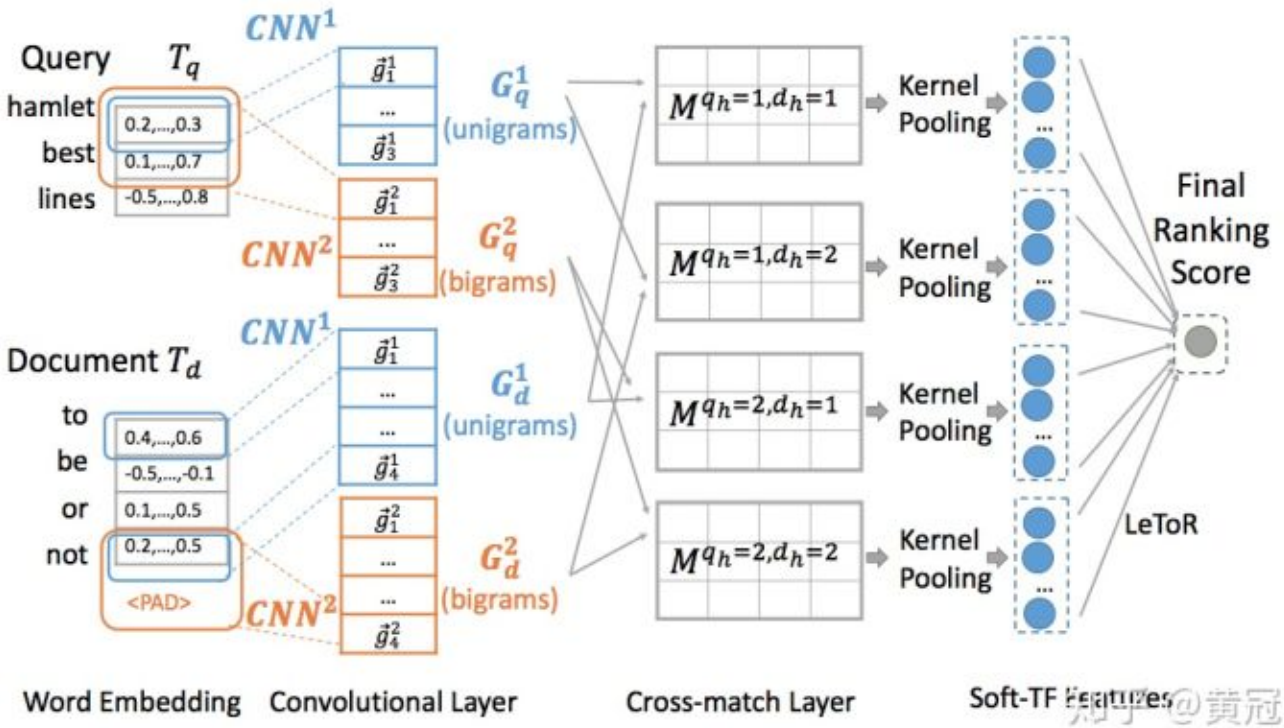
论文中采用RBF的pooling-kernel函数：

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

- μ = 1 and σ → 0，那么除非M[i][j]=1，否则 $-\frac{(M_{ij}-\mu_k)^2}{\sigma^2}$ 都是负无穷大，exp后的值就是0，所以只有M[i][j]=1才会得到非零值，所以这个时候，kernel只会抓取精确匹配的信号
- 如果 σ → ∞， $-\frac{(M_{ij}-\mu_k)^2}{\sigma^2}$ 基本趋于0，所以所有值的贡献都一样，类似捕捉的是doc的term的数目。论文中说这时，kernel函数类似于mean-pooling，这是我暂时还理解不了的。

同理，这个模型是忽略了顺序信息的。

- **Conv-KNRM (Dai et al., WSDM '18)**cs.cmu.edu/~callan/Pape...
  看懂了KNRM后，这篇论文就比较简单，只是用Conv来分别抓取unigram和bigram的信号，然后分别做match-matrix: q-unigram vs d-unigram、q-unigram vs d-bigram、q-bigram vs d-unigram、q-bigram vs d-bigram，然后四个矩阵的匹配信号concat起来即可。



part-7.2 based on Local Context of Matched Terms的一些方法介绍

based on Global Distribution of Matching Strengths的方法是，对于query中的每个term，直

知乎

- 找出它要匹配的doc的局部上下文
  - 匹配query和doc的局部上下文
- 累计每一个term的匹配信号

这种方法的好处是：

- 鲁棒性强，可以过滤掉doc和query无关的部分
- 在doc的局部上下文，可以很好的考虑顺序信息

- **DeepRank (Pang et al., CIKM ’ 17)**arxiv.org/pdf/1710.0564...

这个论文的方法比较简单。模仿人类的判断相关性的过程：

- 对于query中的每个term，找出它在doc中出现的位置。
- 例如query的第二个term:q2，它在doc中三个地方出现，对于这三个位置，分别取出2k+1个词（前后各取k个），不妨假设取出来的三个句子是s1、s2、s3，然后可以用match-matrix等各种方法算出query和s1、s2、s3匹配信号，然后用rnn融合这三个匹配信号，得到一个匹配分
- 将每个term的匹配分加权求和得到最后的匹配分

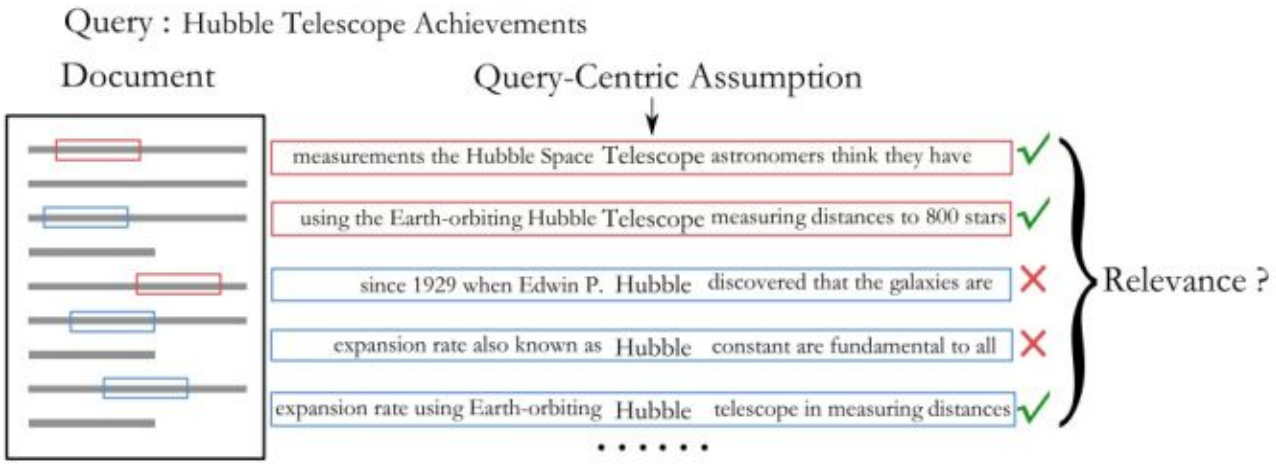
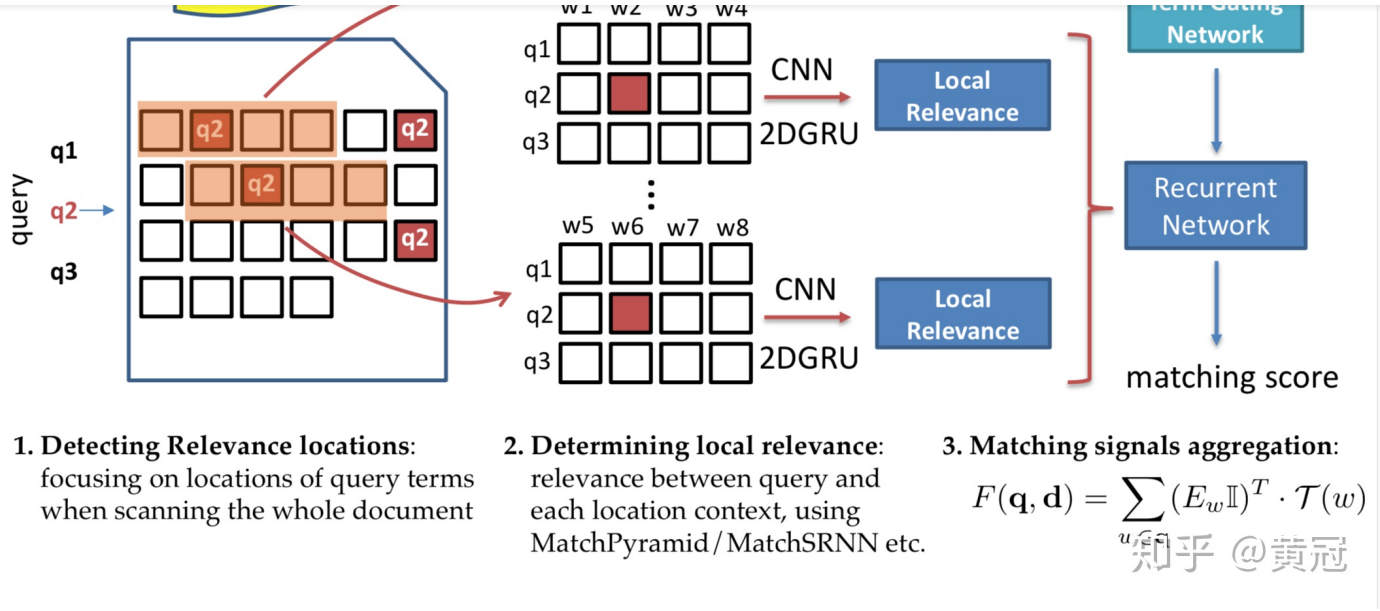


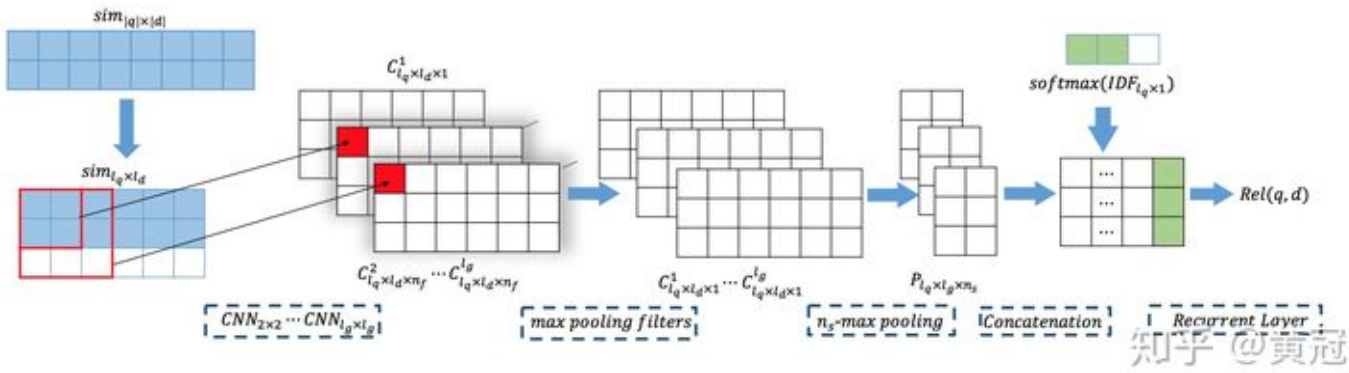
Figure 1: An example of human judgement process.



- **Position-Aware Neural IR Model (PACRR, Hui et al., EMNLP ’17)**arxiv.org/pdf/1704.0394...

其实这篇论文更多的是讲用不同大小的卷积核抓取不同粒度的匹配信号，然后把不同粒度的匹配信号concat，然后用一个rnn将query的每一个term的匹配信号（匹配信号+每个term的权重）融合，得到最后的匹配分。具体步骤：

- 先基于unigram，计算query和doc的match matrix
- 对于上述的match-matrix，使用不同大小的卷积核进行卷积，例如2*2对应获取bigram的匹配信号，3*3对应获取trigram的匹配信号，同样这样的每种粒度的匹配信号，都对应一个match-matrix
- 对于query中的每个term，将上述的不同粒度的匹配信号融合(匹配矩阵的每一行)，例如通过max-pooling(row-wise k-max pooling)、concat等，反正每个term都得到一个关于匹配信号的向量。这个向量还会插入一个新的信号，就是这个term的权重信号。
- 然后用一个rnn将上述的匹配向量融合，得到最后的匹配分。



阵转化成定长的匹配矩阵的两种方法：

- firstk-简单的取前 l_q 行和前 l_d 列，不足的补0
- kwindow-为了抓取n-gram的匹配信号，不妨假设n=3：
  - 对于doc的所有长度为n个词的片段：
    - 计算这个n-gram和query的相似度，例如：计算n-gram的第一个词和query的所有词的相似度，然后将这个n-gram的所有词计算出来的相似度max值取avg，作为这个n-gram和query的相似度打分，然后取top-k个最高分的片断，凑齐长度 l_d
    - 对于每一个n，都取出这样的一堆长度和为 l_d 的词，用这堆文本和query计算匹配矩阵，叫 $sim^n_{l_p*l_d}$ ，那么对于上述的卷积核大小为n*n的卷积，输入就是 $sim^n_{l_p*l_d}$ 。这种情况下，卷积层的stride为[1,n]，即query每次滑动一个词，doc每次滑动n个词，保证处理的是doc的连续的n个词。

不过本人还是觉得这篇论文的重点是用不同大小的卷积核抓取不同粒度的匹配信号，而不是所谓的position-aware。

part-8 short summary

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

part-9 番外篇

这部分主要补充李航的slide上没有的内容，2018开始火起来的bert。

- **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

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。

参考连接：

comp.nus.edu.sg/~xiangn...

arxiv.org/pdf/1706.0661...

[1511.08277] A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations

百度NLP | 神经网络语义匹配技术

“百度一下”背后：深度学习技术的最新应用

《attention is all you need》解读

ijcai.org/Proceedings/1...