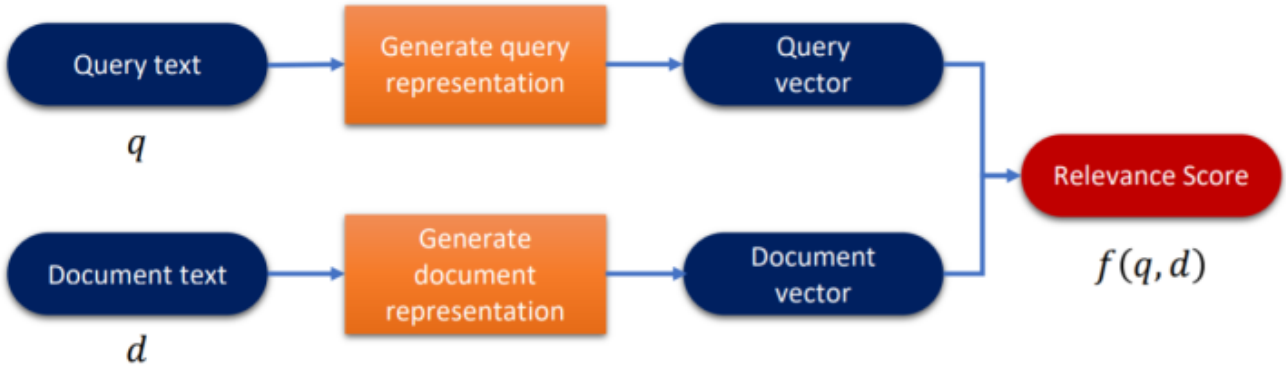
经典IR模型

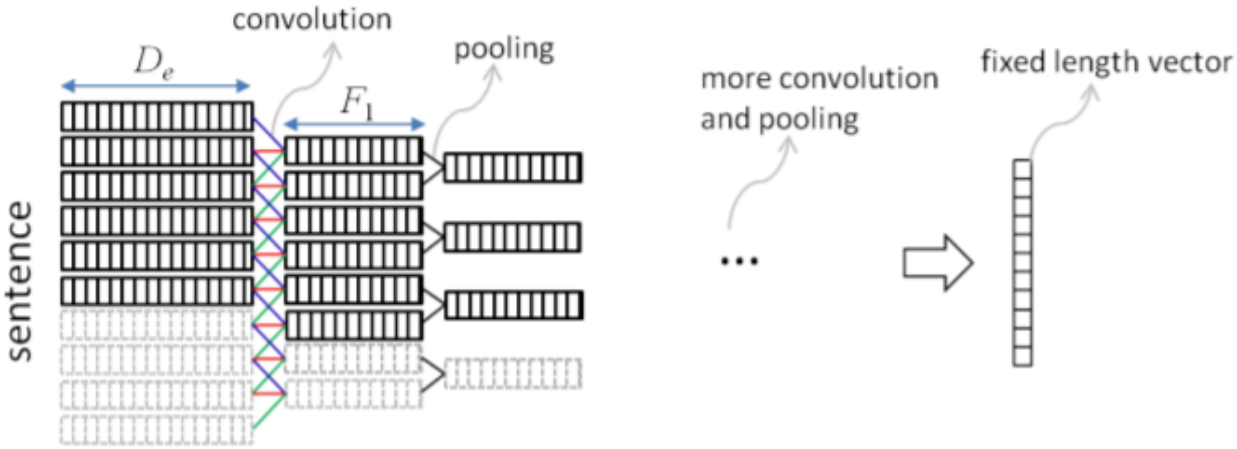
Representation-based IR Models(2017年之前)

基于表示的信息检索模型采用典型的双塔结构（这也是在召回中使用的结构）。query用query vector表示，document用document vector表示，最后计算相关度。



1. Convolutional Neural Network Architectures for Matching Natural Language Sentences. (2015) 中的ARC-I

这篇文章出自华为诺亚方舟实验室，采用 **CNN** 模型来解决语义匹配问题。首先，文中提出了一种基于CNN的句子建模网络，如下图：



图中灰色的部分表示对于长度较短的句子，其后面不足的部分填充的全是0值(Zero Padding)。图中的卷积计算和传统的CNN卷积计算无异，而池化则是使用Max-Pooling。

下面是基于之前的句子模型，建立的两种用于两个句子的匹配模型。

ARC-1：

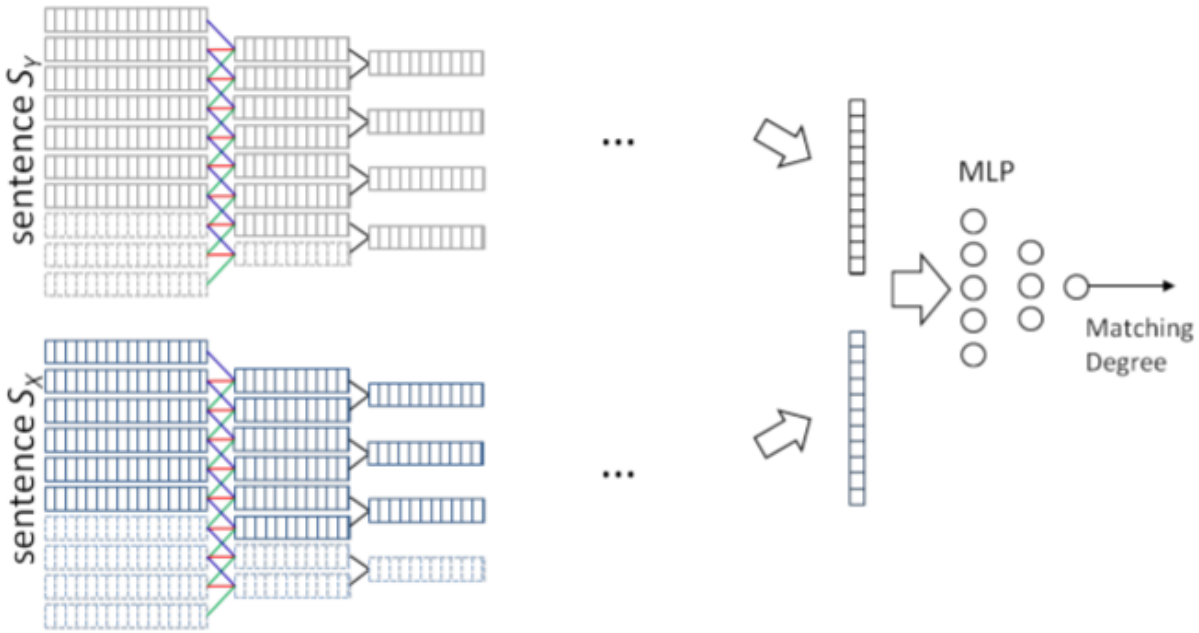


Figure 3: Architecture-I for matching two sentences.

这个模型比较简单，但是有一个较大的缺点：两个句子在建模过程中是完全独立的，**没有任何交互行为**，一直到最后生成抽象的向量表示后才有交互，这样做使得句子在抽象建模的过程中会丧失很多语义细节，同时过早地失去了句子间语义交互计算的机会。（这也是双塔模型被诟病的原因）

2. Deep Semantic Similarity Model(DSSM, 2013)

这篇文章出自微软2013年的文章，算是双塔最早的文章之一。

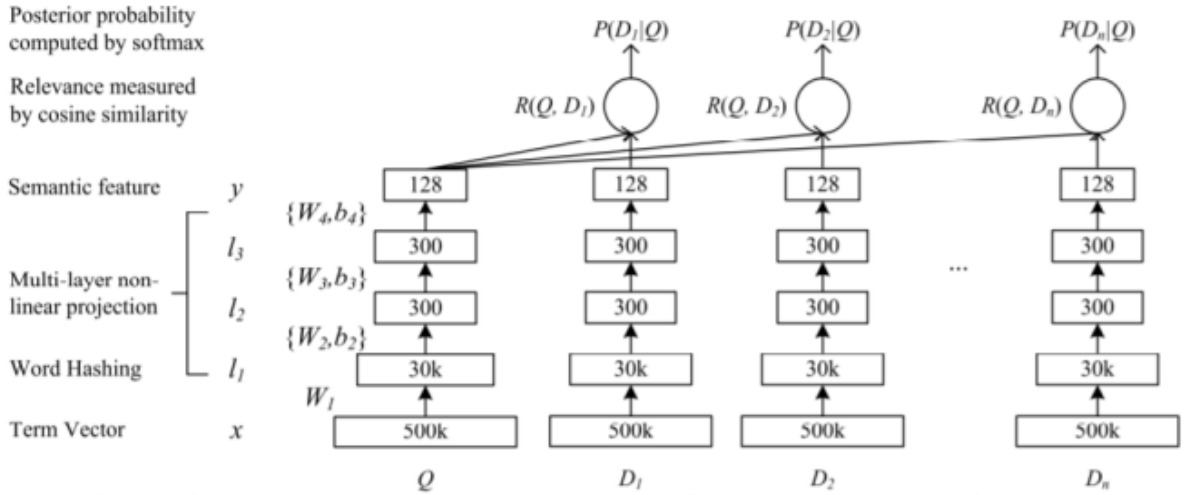
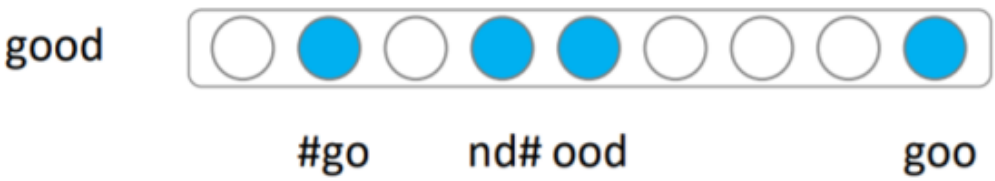


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear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

输入层为经过word-hashing之后的30k维结果，输出为计算的query和文档之间的cosine similarity。

文中提到的word hashing方法是为了解决token过多的问题（对于one-hot编码），同时解决OOV，但是会在一定程度上带来一些哈希冲突。文中提到的character-trigram在今天还经常使用，作为BPE、word embedding、n-gram word embedding的必要补充。

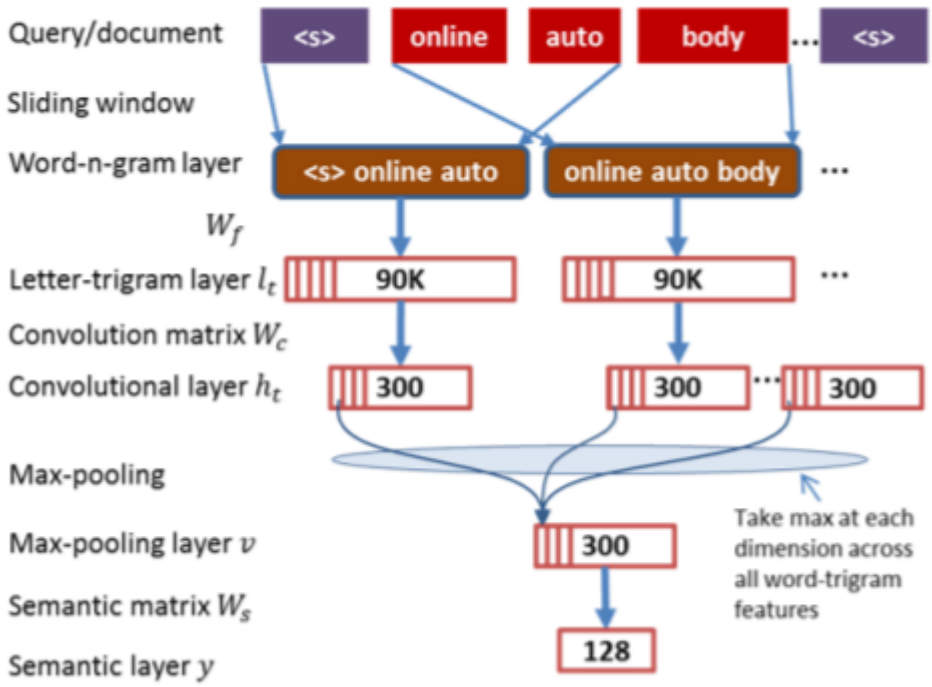
- Given a word
  - **good**
- Add a mark (#) to the start and end of the word
  - **#good#**
- Break the word into letter n-grams
  - **trigrams: #go, goo, ood, od#**
- Represent the word using a vector of letter n-grams



- Why word hashing:
  - Vocabulary is often too large (out of vocabulary)
- Collision in word hashing vectors:
  - Different words may have a same word hashing vector
    - For example, '#bananna#' and '#bannana#'
  - The collision probability is very low:

Vocabulary	Type	Unique Key	Collision
40K	Bigram	1107	18
	Trigram	10306	2
500K	Bigram	1607	1192
	Trigram	30621	22

3. [Convolutional Latent Semantic Model (CLSM,2014)]



CLSM(convolutional latent semantic model) 主要的思想是使用CNN模型来提取语料的语义信息，卷积层的使用保留了词语的上下文信息，池化层的使用提取了对各个隐含语义贡献最大的词汇。

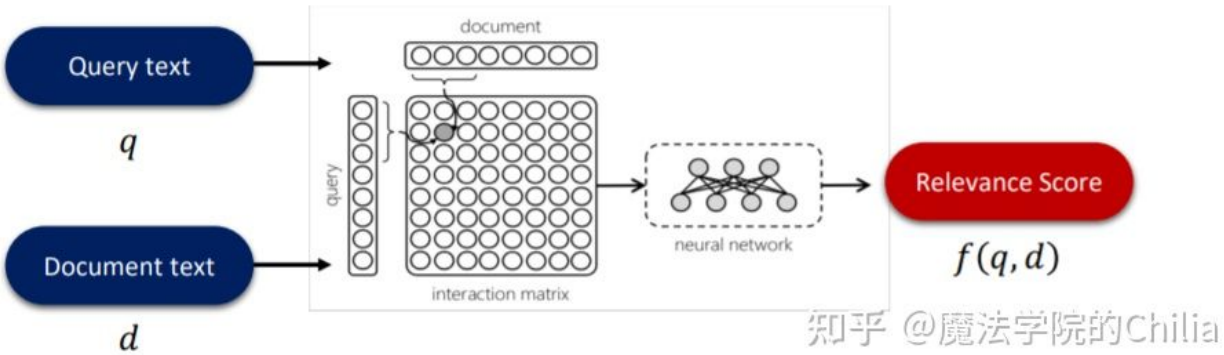
首先用一个滑动窗口得到word-n-gram, 然后通过word-hashing得到每个窗口的向量表示。max-pooling层把不同的滑窗的最大信息提取出来，构成一个fix-length vector。本文和DSSM最大的区别就是不是词袋模型了，而是考虑了滑窗的位置信息。

- The cosine similarities between learned word-n-gram feature vectors of ***office*** and ***body*** in different contexts

microsoft *office* software		car *body* shop	
Free *office* 2000	0.550	car *body* kits	0.698
download *office* excel	0.541	auto *body* repair	0.578
word *office* online	0.502	auto *body* parts	0.555
apartment *office* hours	0.331	wave *body* language	0.301
massachusetts *office* location	0.293	calculate *body* fat	0.220
international *office* berkeley	0.274	forcefield *body* armour	0.165

基于交互的模型

- Establish an interaction matrix M
  - M_{ij} is obtained by comparing the i^{th} word in query and the j^{th} word in doc
    - For example, $M_{ij} = \cos(\vec{v}_{t_i}, \vec{v}_{t_j})$
- Employ neural networks to extract features and get the ranking score



representation-based IR model（双塔模型）和Interaction-based IR model（**金字塔模型**）的区别可以形象地表示为下图：

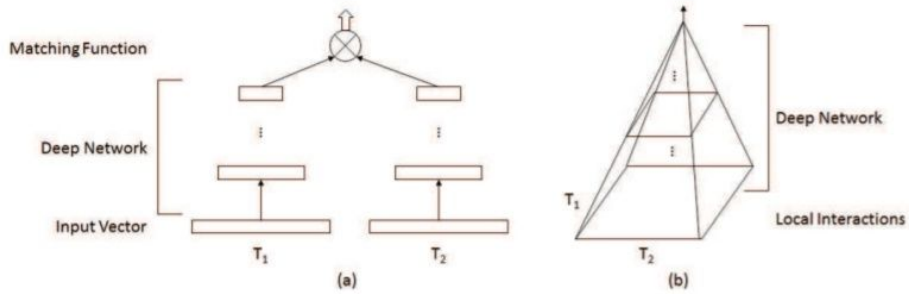


Figure 1: Two types of deep matching models: (a) Representation-focused models employ a Siamese (symmetric) architecture over the text inputs; (b) Interaction-focused models employ a hierarchical deep architecture over the local interaction matrix.

代表论文：

1）Convolutional Neural Network Architectures for Matching Natural Language Sentences. (2015) 中的ARC-II：

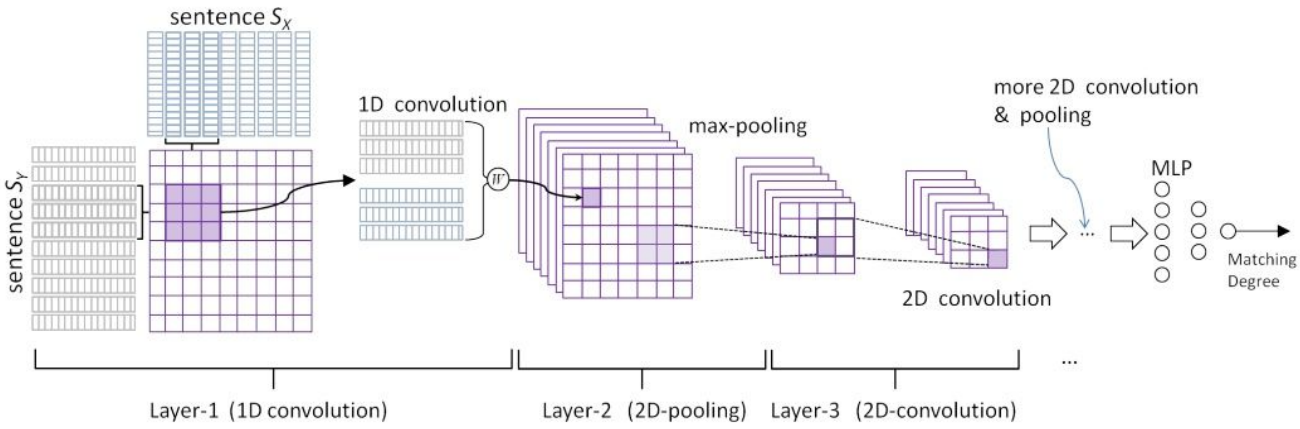


Figure 4: Architecture-II (ARC-II) of convolutional matching model

上图所示的 ARC-II 在第 1 层卷积后就把文本 X 和 Y 做了融合，具体的融合方式是，首先从Sentence x中任取一个向量 $\boldsymbol{x_a}$ ，再从Sentence y中将每一个向量和$\boldsymbol{x_a}$进行一维卷积操作，这样就构造出一个 2D 的 feature map，然后对其做 2D MAX POOLING，多次 2D 卷积和池化操作后，输出固定维度的向量，接着输入 MLP 层，最终得到文本相似度分数。

2）MatchPyramid

https://arxiv.org/pdf/1606.04648.pdfarxiv.org/pdf/1606.04648.pdf

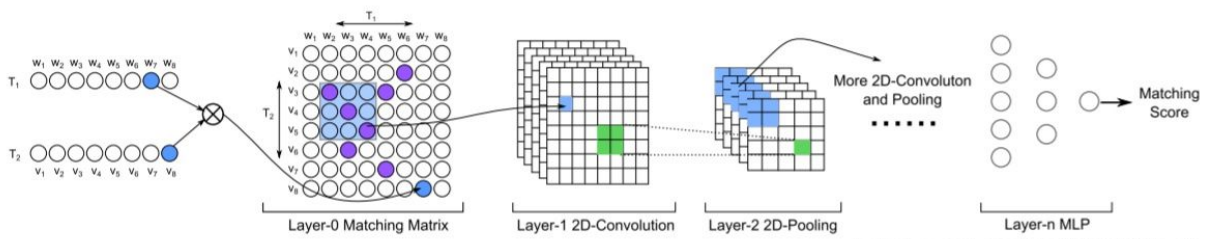


Figure 1: Model structure of MatchPyramid.

其中，

$\otimes$

是计算相似度的符号。文中提出四种计算相似度的方法：

Indicator Function produces either 1 or 0 to indicate whether two words are identical.

$$\mathbf{M}_{ij} = \mathbb{I}_{\{w_i = v_j\}} = \begin{cases} 1, & \text{if } w_i = v_j \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Cosine views angles between word vectors as the similarity, and it acts as a soft indicator function.

$$\mathbf{M}_{ij} = \frac{\vec{\alpha_i}^\top \vec{\beta_j}}{\|\vec{\alpha_i}\| \cdot \|\vec{\beta_j}\|}, \quad (3)$$

where $\|\cdot\|$ stands for the ℓ_2 norm of a vector.

Dot Product further considers the norm of word vectors, as compared to cosine.

$$\mathbf{M}_{ij} = \vec{\alpha_i}^\top \vec{\beta_j}. \quad (4)$$

Gaussian Kernel is a well-known similarity function. This similarity function is introduced in this work based on our studies.

$$\mathbf{M}_{ij} = e^{-\|\vec{\alpha_i} - \vec{\beta_j}\|^2}. \quad (5)$$

3) Deep Relevance Matching Model (DRMM)

https://arxiv.org/pdf/1711.08611.pdfarxiv.org/pdf/1711.08611.pdf

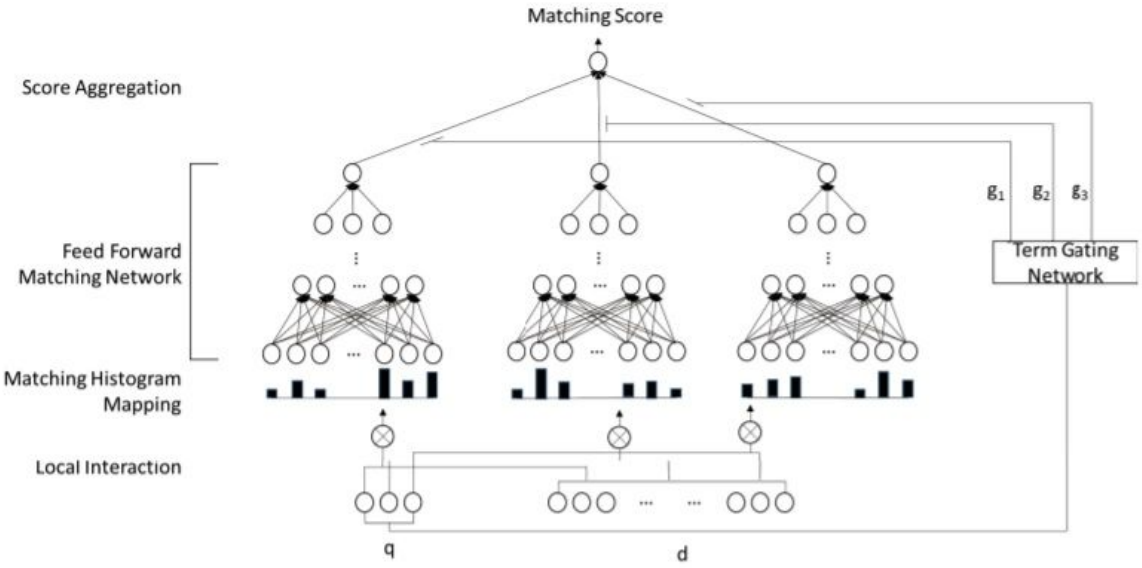


Figure 2: Architecture of the Deep Relevance Matching Model.

对于query的每个token，都去计算它和document的每个token的相似度，然后分桶，作为若干层MLP的输入

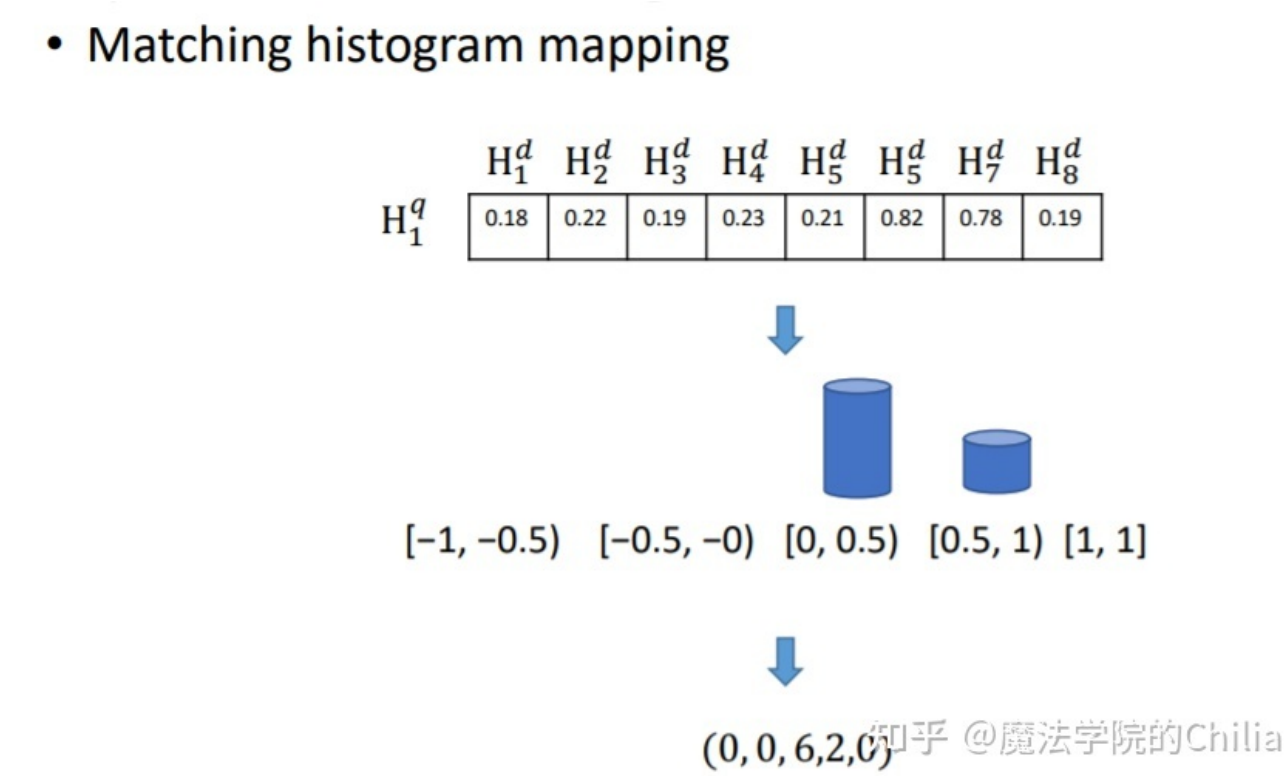
$$\boldsymbol{z}_i^{(0)} = h(w_i^{(q)} \otimes d), \qquad i = 1, \ldots, M$$

$$\boldsymbol{z}_i^{(l)} = tanh(\boldsymbol{W}^{(l)} \boldsymbol{z}_i^{(l-1)} + \boldsymbol{b}^{(l)}), \qquad i = 1, \ldots, M, l = 1, \ldots, L$$

$$s = \sum_{i=1}^{M} g_i z_i^{(L)}$$

$\otimes$

表示计算相似度，h表示分桶函数，g是门控。



4) Kernel-based Neural Ranking Model (K-NRM, 2017)

https://arxiv.org/pdf/1706.06613.pdfarxiv.org/pdf/1706.06613.pdf

本文和 MatchPyramid 的核心的不同之处在于 RBF Kernel：

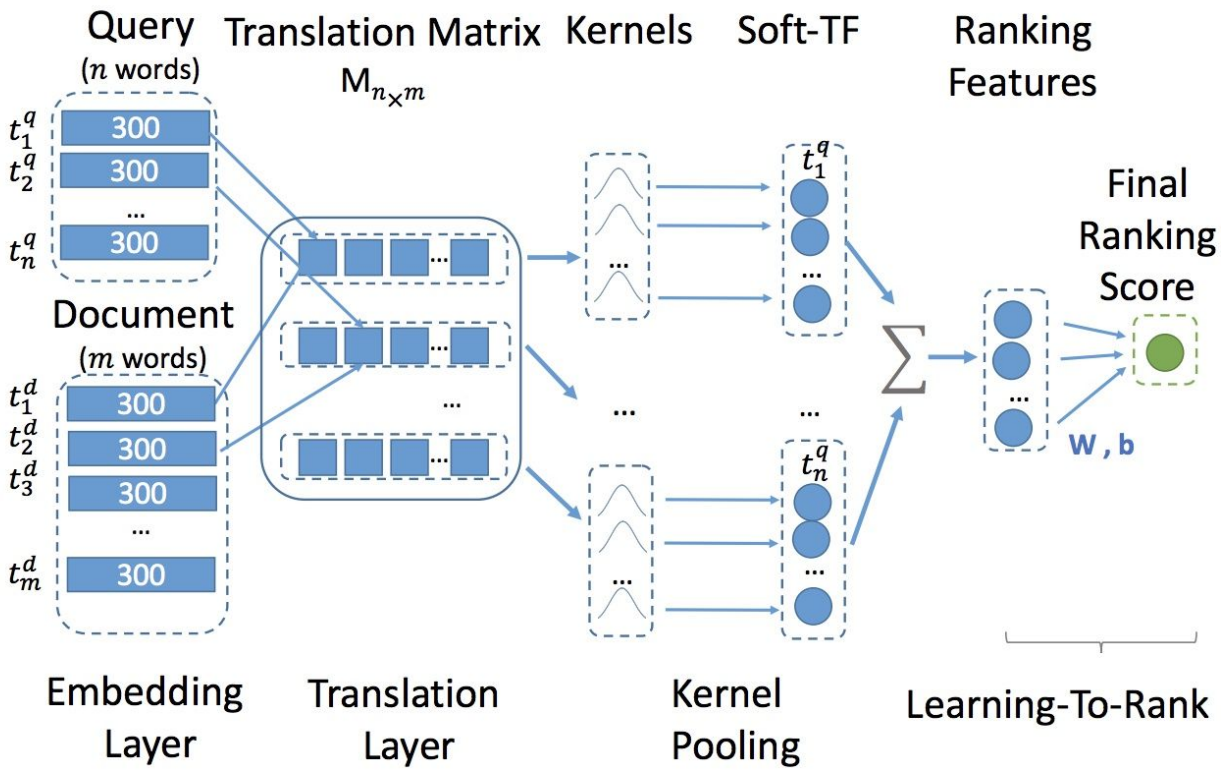
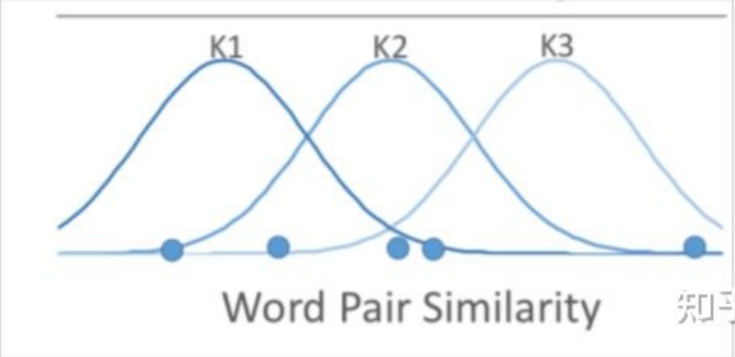


Figure 1: The Architecture of K-NRM. Given input query words and document words, the embedding layer maps them into distributed representations, the translation layer calculates the word-word similarities and forms the translation matrix, the kernel pooling layer generate soft-TF counts as ranking features,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

先把 term 映射为 Word Embedding,再计算两两相似度矩阵M，然后通过 RBF Kernel：

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

- Where K_k is the k-th kernel, μ_k is the mean of kernel k, σ defines the kernel width, and M is the interaction matrix



translation矩阵的每一行经过kernel之后都变成一个数，总共有K个kernel。

最后把所有的soft-TF输出取log相加，再过若干层MLP输出结果。采用 pairwise learning to rank loss 进行训练：

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

5）Conv-KNRM (2018)

http://www.cs.cmu.edu/zhuyund/papers/WSDM_2018_Dai.pdfwww.cs.cmu.edu/zhuyund/papers/WSDM_2018_Dai.pdf

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。

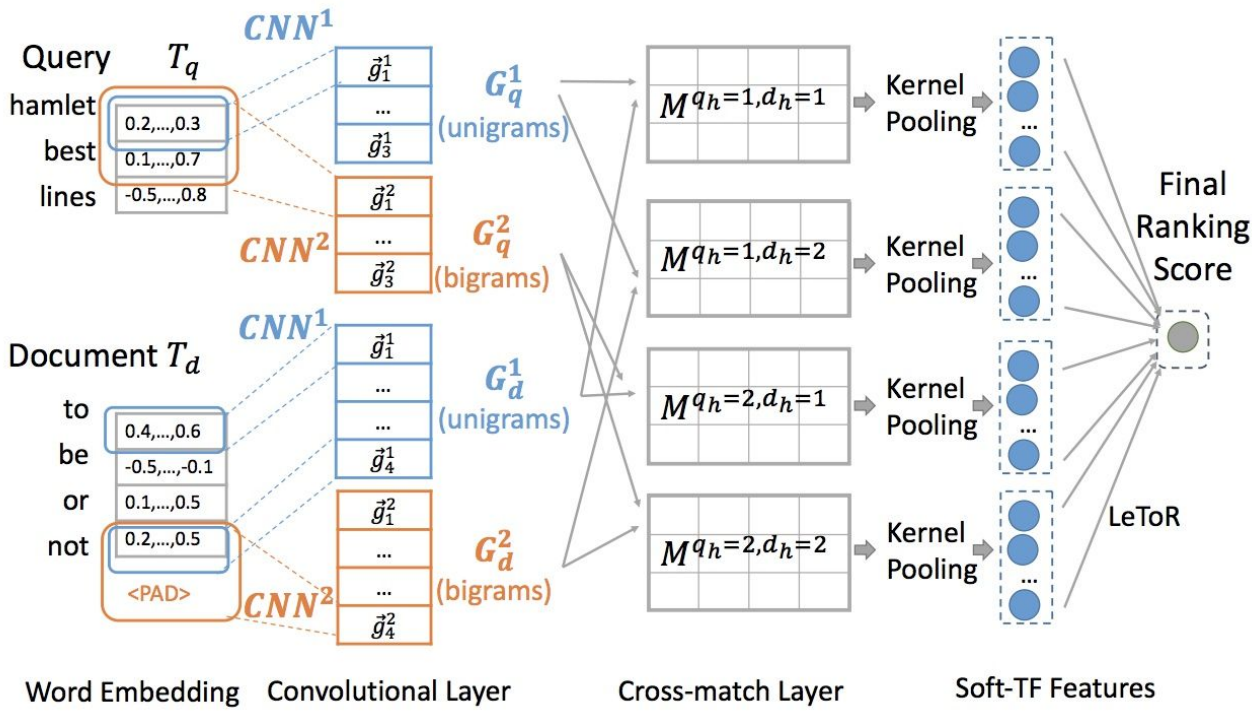


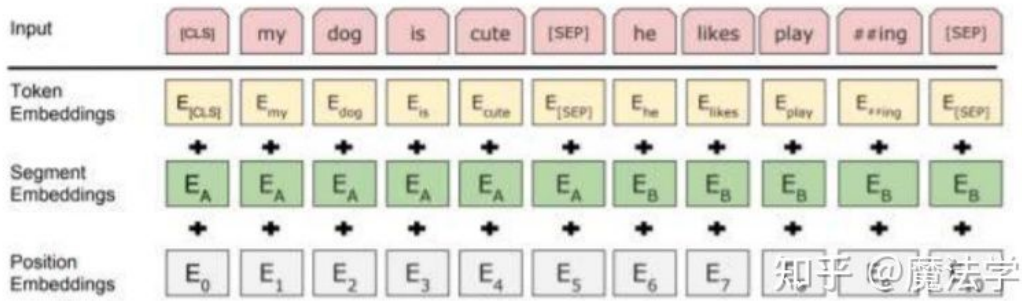
Figure 1: The Conv-KNRM Architecture. Given input query and document, the embedding layer maps their words into distributed representations, the convolutional layer generates n-gram embeddings ; the cross-match layer matches the query n-grams and document n-grams of different lengths, and forms the translation 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.

- Examples of matched n-grams between query and snippets
  - Black phrases contribute more to the relevance score than gray ones

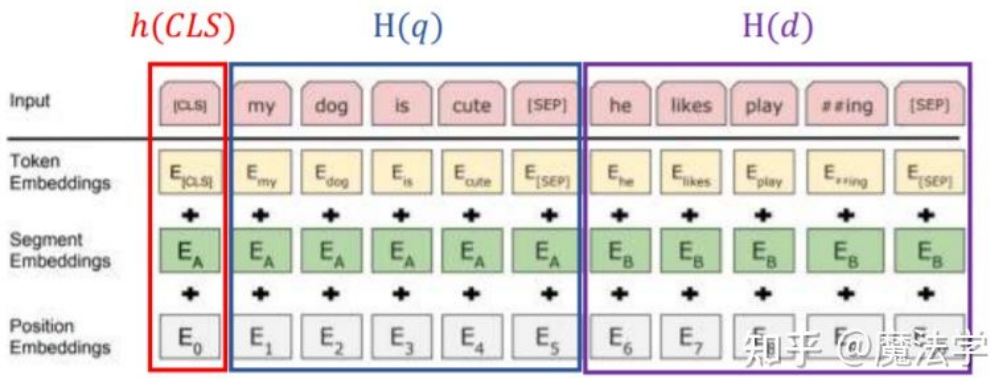
Query	Snippet
sewing instructions	...home free resources! newsletter sewing ideas...**quilting 101** what is a quilt...
atypical squamous cells	...treatment decision tools **cervical cancer** : prevention and early detection...
moths	... grouping of moth families commonly known as the 'smaller moths ' (**micro , lepidoptera**)...
fickle creek farm	.. bed & breakfast inns extended stay lodging rv parks where to **eat & drink** nightlife ...
university phoenix	campus locations programs : bachelor **degree masters degrees** account degrees business degree...
wedding **budget calculator**	...planning tips photographs bridal board my **perfect planner** tools my check lists...

6) ★ BERT (2019)

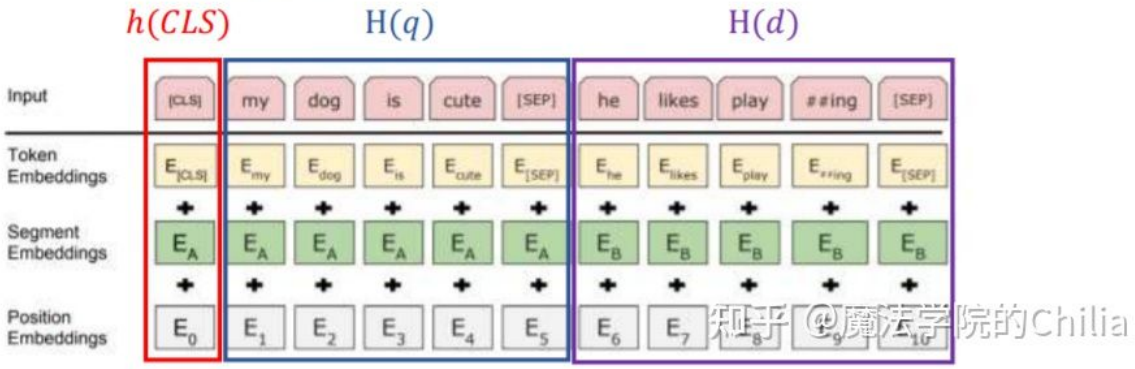
- Input: WordPiece embedding + position embedding + segment embedding
  - WordPiece: Convert words to subwords
    - Keeping the Secret of Genetic Testing [Keeping, the, Secret, of, Gene, ##tic, Testing]
    - He like play [He, like, play, ##ing]



- Given a query q and a document d.
  - Three kinds of representations are calculated
    - [CLS] representation $h(CLS)$
    - Query representation $\mathrm{H}(q)$
    - Document representation $\mathrm{H}(d)$



- $f(q,d) = MLP(h(CLS))$ with [CLS] representation
- Or $f(q,d) = MLP(\phi(H(q),H(d)))$ with query and document representations. ϕ can be representation-based and interaction-based architectures



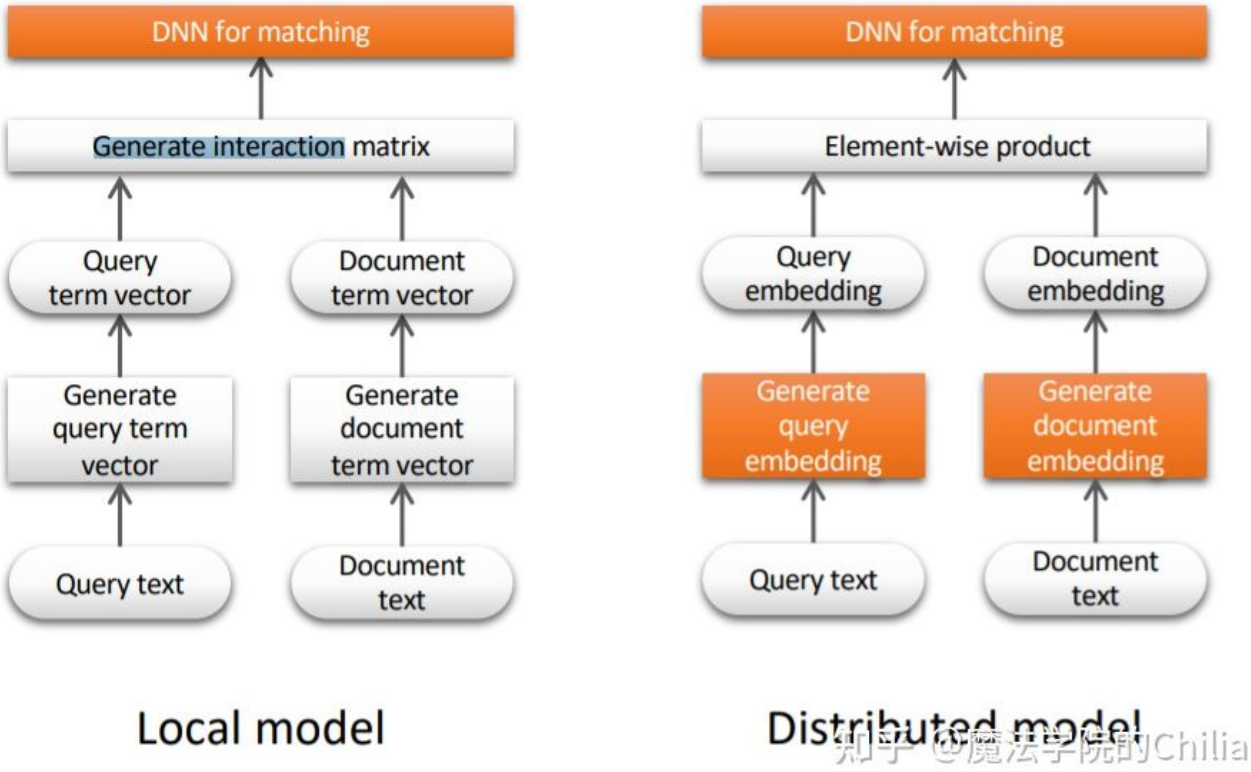
4. Further Combination

Duet model (2017)

https://arxiv.org/pdf/1610.08136.pdfarxiv.org/pdf/1610.08136.pdf

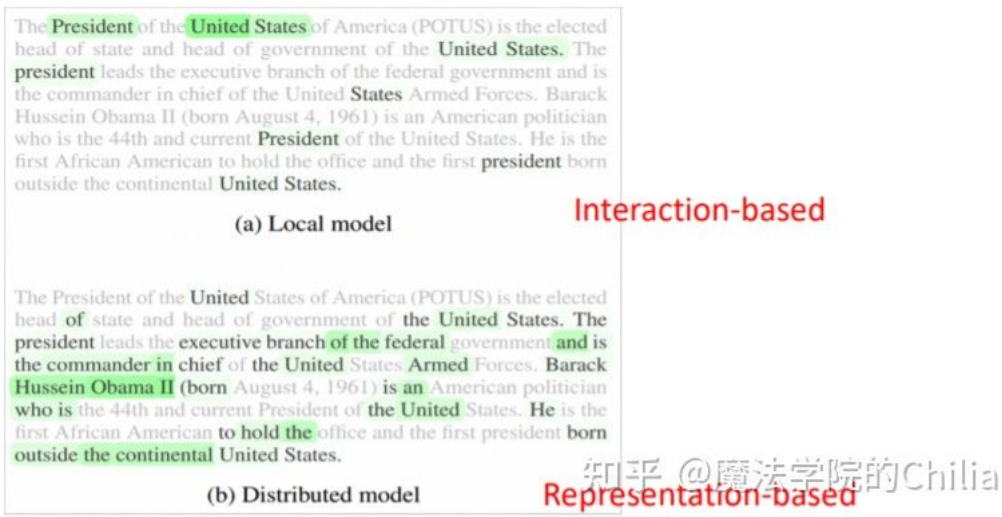
- local model: if each term is represented by a unique identifiers (local representation) then the query-document relevance is a function of the pattern of occurrences of the **exact** query terms in the document.
- distributed model: if the query and the document text is first projected into a continuous latent space, then it is their distributed representations that are compared.

Local model 与 distributed model 各有所长。Local model具有记忆能力，distributed model具有泛化能力。例如，distributed model会非常了解"Barack Obama" (因为训练语料很多)，而不了解"Bhaskar Mitra",所以在后者上会表现较差(甚至接近于随机初始化！)。Local model对"Barack Obama"和"Bhaskar Mitra"都没有了解，在local model看来，它们不过是一些token。但是在"Bhaskar Mitra"上，local model会表现得比distributed model出色。所以，为什么不把二者结合起来呢?



Local Model全部靠精确匹配

- Individually removing each term for the query “united states president”
  - Darker green parts drop significantly in retrieval score



local model是exact-match

我的理解是，exact-match和semantic similarity都很重要。现在工业界的搜索召回一般都是多路召回，其中用关键词去elasticsearch等搜索框架中去搜依然是很重要的一路召回。但是还要加上语义向量检索等多路召回以辅助，这样才能把用户可能感兴趣的item找出来。如果光用exact match，会导致很多和query相关的item搜不出来，比如搜"sneaker"不会出来"running shoes"；如果只用semantic search,会导致一些不相关的结果出现，比如搜"adidas shoes"会出现"nike sneakers".