论文｜从DSSM语义匹配到Google的双塔深度模型召回和广告场景中的双塔模型思考

原创 Thinkgamer 搜索与推荐Wiki 2020-08-04

收录于话题
#Thinkgamer 11 #论文笔记 19

▼ 相关推荐 ▼

1、基于DNN的推荐算法介绍
2、传统机器学习和前沿深度学习推荐模型演化关系
3、论文｜AGREE-基于注意力机制的群组推荐（附代码）
4、论文｜被“玩烂”了的协同过滤加上神经网络怎么搞?

本文包含（文章较长，建议先收藏再阅读，点击文末的阅读原文，查看更多推荐相关文章）：

- DSSM
- DSSM的变种
- MV-DNN
- Google Two Tower Model
- 广告场景中的DSSM双塔模型
- 总结

基于深度学习的召回近些年已经得到了长足的发展，其中双塔模型更是作为经典的深度学习召回模型被各大公司应用，回顾双塔模型的发展可以追溯到2013年微软发布的DSSM模型，本篇文章将会从DSSM开始介绍几篇论文，看一下DSSM模型是怎么发展成为双塔模型并应用在推荐系统中做召回的。

DSSM

DSSM模型是2013年微软发布的，其论文全称为：Learning Deep Structured Semantic Models for Web Search using Clickthrough Dara（https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/cikm2013_DSSM_fullversion.pdf ）。其发表的本意是用来语义相似

度计算。通过对用户的Query历史和Document进行embedding编码，使用余弦相似度计算用户query的embedding和document的相似度，继而达到语义相似度计算的目的。

论文首先介绍了已有的语义分析模型，比如

- LSA、PLSA、LDA，但是因为他们采用的都是无监督的学习方法，因此在实际的场景中表现的并不好。
- 通过用户查询和点击序列进行建模的BLTMs和DPMs，BLTMs不仅要求共享主题分布，而且还给每个主题分配相似的词组；DPMs则使用S2Net算法并结合LTR中的piarwise方法。在实验中他们的效果表现要比LSA、PLSA、LDA效果好。然后虽然BLTMs使用了点击数据进行训练，但其使用的是最大似然方法优化目标，该方法对于排序来讲并不是最优的。DPMs则会产生一个巨大的稀疏矩阵，虽然可以通过删除词汇减小维度，但相应的效果也会减弱
- 结合深度自编码的方法，虽然表现较传统的LSA效果好，但由于其采用的是无监督的学习方法，模型参数的优化是基于重建文档，而不是区分相关性，因此基于深度自编码的方法在效果上并没有显著优于关键词匹配。

因此作者们提出了深层结构化语义模型（Deep Structured Semantic Model，DSSM）。相比之前的提到几个模型DSSM的主要区别在于：

- 有监督，使用最大似然函数进行优化
- 使用word-hashing方法解决大规模且稀疏的词典问题
- 将用户的Query行为和Document映射到同一语义空间中，通过余弦相似度计算相似性

DSSM模型的结构如下图所示：

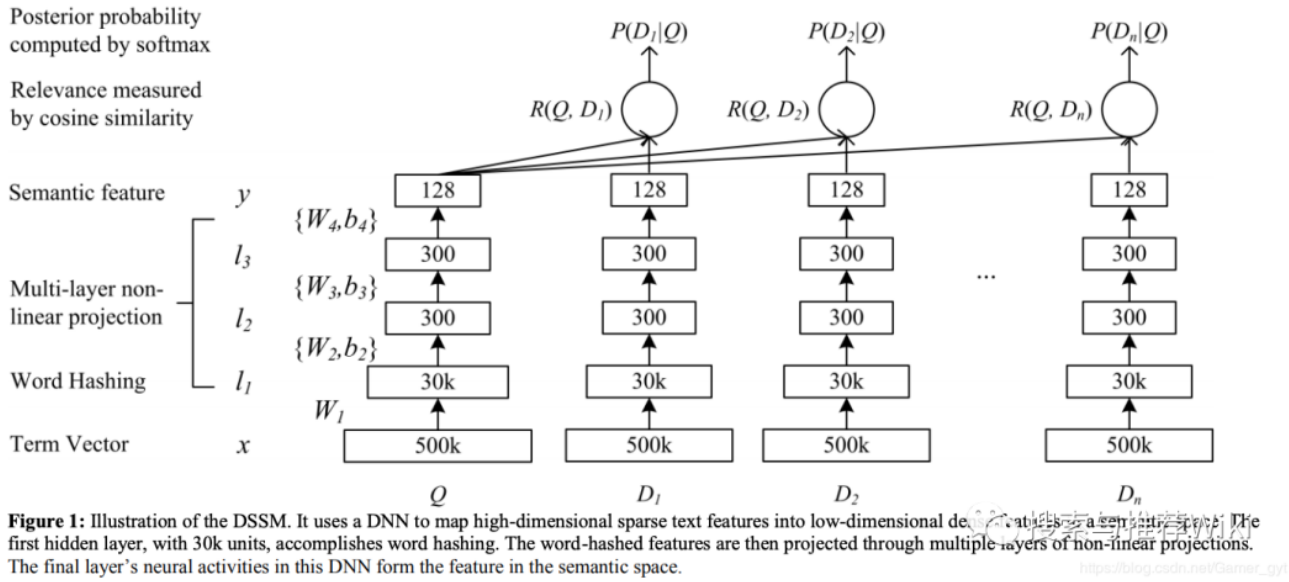


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear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

DSSM模型结构

从上图可以看出，输入DSSM的是一个高维的向量，经过若干层的神经网络，输出一个低维的向量，分别用来表示user的query意图和document，最后通过余弦相似度计算Q和D的相似度。

每一层的输出可以表示为：

$$l_i = f(W_i l_{i-1} + b_i), i = 1, 2, \ldots, N-1$$

其中：

- W_i 表示第i层的权重矩阵
- b_i 表示第i层的偏置，特别的第一层的偏置为0

最终输出的结果为：

$$y = f(W_N l_{N-1} + b_N)$$

这里采用的激活函数为tanh，其表达形式为：

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

余弦相似度的计算公式为：

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

其中：

- y_Q、y_D分别表示用户查询低维向量和文档低维向量

论文的另一个出色的点在于抛弃了传统的Bag-of-word模型，因为其会带来高维度的向量特征，这里使用word hashing技术来代替词袋模型，word hashing基于n-gram，比如一个单词*good*，使用word hashing技术进行拆分，首先在其两端补充标记符 “#”，假设n=3，则“#good#” 可以表示为：#go、goo、ood、od#。

但采用word hashing技术会带来一个问题就是：词汇冲突，即两个表达含义不同的词拥有相同的n-gram向量表示。但是论文作者也提到了这种概率很小，冲突的概率在0.0044%，如下图所示：

	Letter-Bigram		Letter-Trigram	
Word Size	**Token Size**	**Collision**	**Token Size**	**Collision**
40k	1107	18	10306	2
500k	1607	1192	30621	22

Table 1: Word hashing token size and collision numbers as a function of the vocabulary size and the type of letter ngrams.

词汇冲突统计

在用户的一次查询中，我们假设用户点击的doc是和query相关的，这样就可以使用有监督的方法进行模型的训练，这里采用的是最大似然函数（maximize the conditional likelihood），其中

要学习的参数是权重矩阵W_i和偏置b_i。

这里通过softmax函数计算用户query doc之后被点击的后验概率，表达式如下：

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

其中：

- γ 表示的是平滑因子
- D 表示的整个候选文档库（其中用户在一次查询中点击的结果用D^+表示，未点击的结果用D^-表示，这里对未点击的进行负采样，比例为1:4）

> 论文中特意提到采用不同的采样比例对结果影响不大

模型训练采用最大似然函数，优化目标是最小化下列损失函数：

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

其中

- Λ 表示神经网络的参数W_i, b_i

关于一些实验的细节

- 将准备好的数据集分为两部分（训练集和验证集），且没有交叉
- DNN网络使用三个隐藏层结构，第一层是word hashing 层，大约30K个节点，第二、第三层有300个节点，输出层128个节点
- 网络权重初始化使其符合分布$-\sqrt{6/(fanin + fanout)}, \sqrt{6/(fanin}$，其中$fanin$ 、 $fanout$分别表示输入和输出单元数
- 实验对比，分层预训练并没有比直接训练效果好
- 优化算法采用的是随机梯度下降（SGD）
- 训练的批大小 batch-size=1024
- 前向和反向传播过程中单个批样本迭代次数 epochs=20
- 实验评估指标为NDCG，同时结合t-test和p-value，进行分析，使结果更严谨

实验结果实验结果如下表所示，9-12 表示的是使用不同配置的DSSM

- 9：DSSM但没有使用word hashing + 有监督训练
- 10：DSSM + word hashing + 有监督训练 + 线性激活函数
- 11：DSSM + word hashing + 有监督训练 + 非线性激活函数
- 12：DSSM + word hashing + 有监督训练 + 非线性激活函数 + 细节上的一些优化

> 11 和 12 有一些模糊，看了好几遍说明，也没有看懂具体的区别在哪

#	Models	NDCG@1	NDCG@3	NDCG@10
1	TF-IDF	0.319	0.382	0.462
2	BM25	0.308	0.373	0.455
3	WTM	0.332	0.400	0.478
4	LSA	0.298	0.372	0.455
5	PLSA	0.295	0.371	0.456
6	DAE	0.310	0.377	0.459
7	BLTM-PR	0.337	0.403	0.480
8	DPM	0.329	0.401	0.479
9	DNN	0.342	0.410	0.486
10	L-WH linear	0.357	0.422	0.495
11	L-WH non-linear	0.357	0.421	0.494
12	**L-WH DNN**	**0.362**	**0.425**	**0.498**

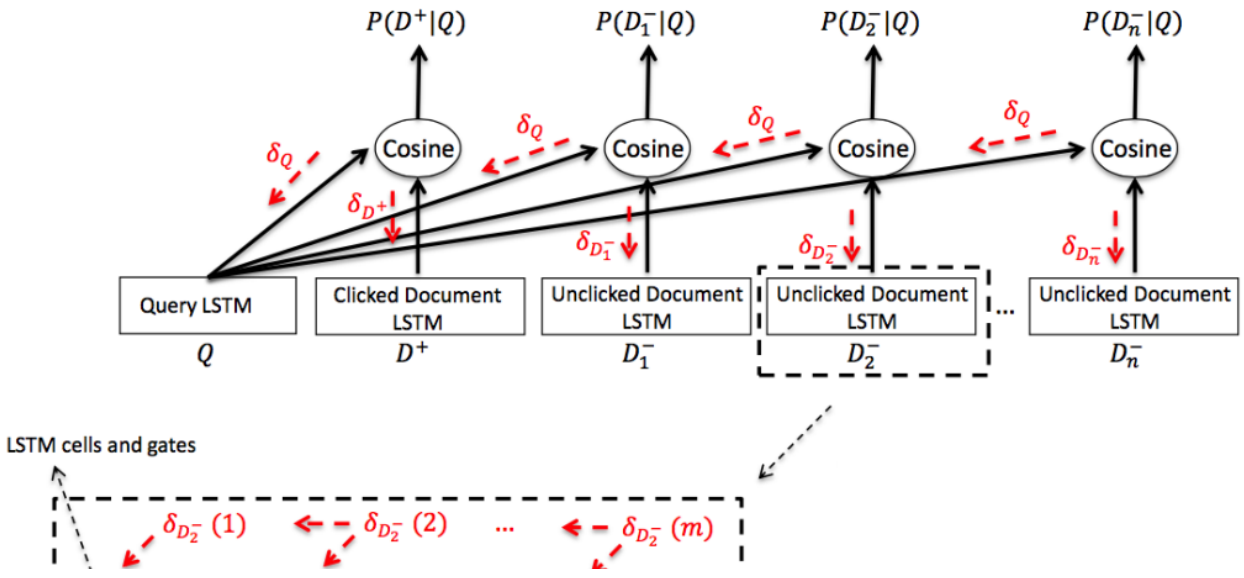
Table 2: Comparative results with the previous state of the art approaches and various settings of DSSM.

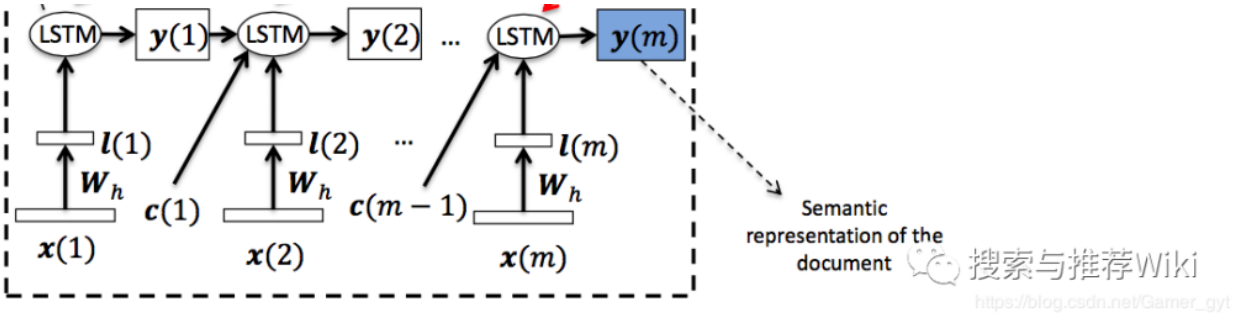
实验结果

DSSM的变种

关于DSSM有很多演变模型，这里主要还是针对语义匹配模型。其主要的变种有：

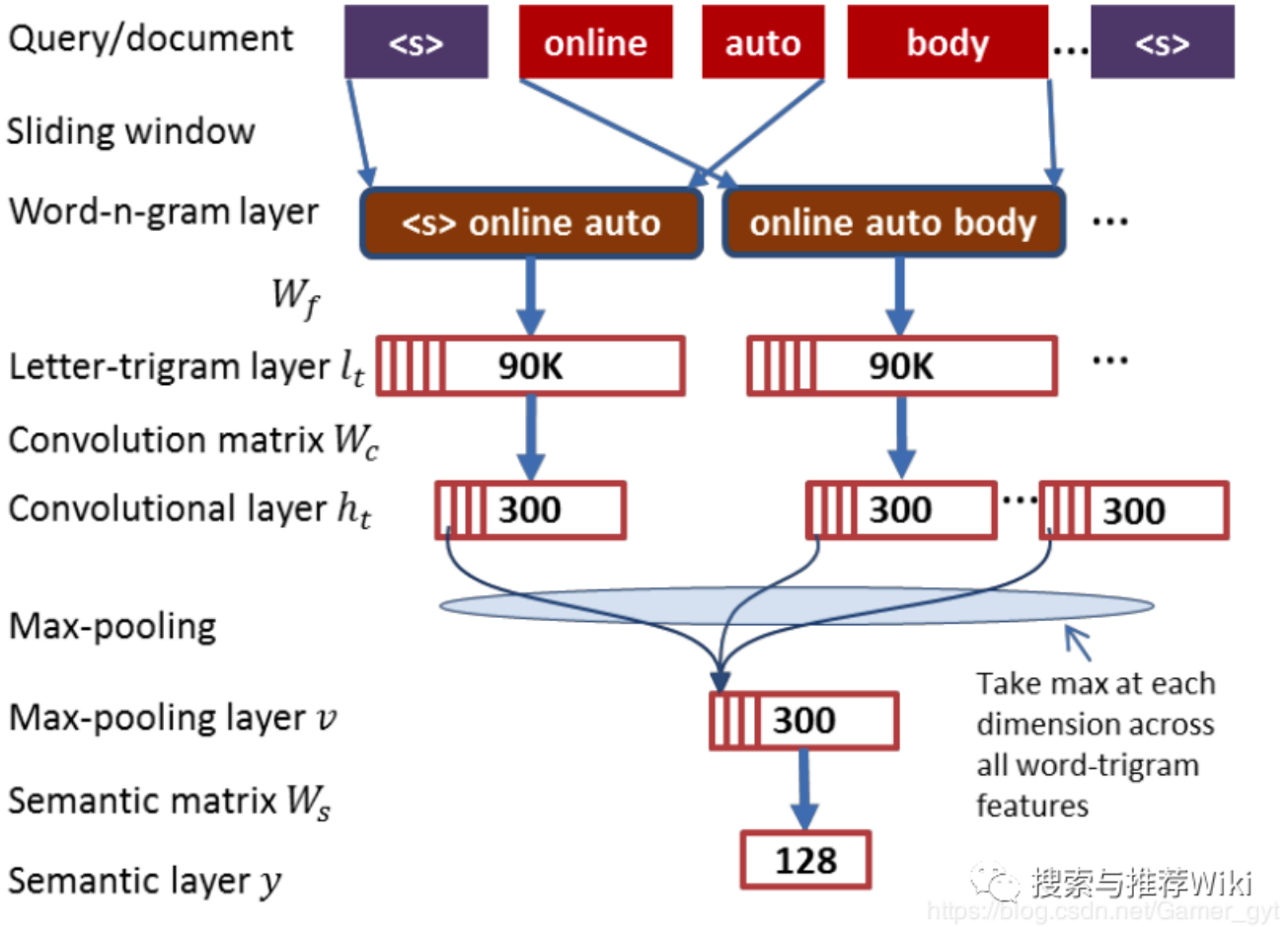
- LSTM-DSSM：https://arxiv.org/pdf/1412.6629.pdf ，2014年提，主要思路是将DSSM换成了LSTM





LSTM-DSSM

- CNN-DSSM：http://www.iro.umontreal.ca/~lisa/pointeurs/ir0895-he-2.pdf ，2015年提出，主要思路是将DSSM换成了CNN



CNN-DSSM

- MV-DNN【2015年提出】：https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/frp1159-songA.pdf

---

另外附上两篇关于语义匹配的实践文章：

- 百度丨语义匹配中的技术实践：https://www.jiqizhixin.com/articles/2017-06-15-5
- 阿里丨深度语义模型以及在淘宝搜索中的应用：https://developer.aliyun.com/article/422338

这里主要说一下MV-DSSM，因为MV-DSSM将DSSM应用在推荐上，并取得了不错的效果。

MV-DNN

Multi-View-DNN联合了多个域的丰富特征，使用multi-view DNN模型构建推荐，包括app、新闻、电影和TV，相比于最好的算法，老用户提升49%，新用户提升110%。并且可以轻松的涵盖大量用户，很好的解决了冷启动问题。但其使用前提是统一个主体公司下有很多APP，不同APP之间的用户数据是可以互相使用的，从其设计思路上讲就很容易的将该算法的使用限定在了为数不多的公司里。

MV-DSSM的五点贡献：

- 使用丰富的用户特征，建立了一个多用途的用户推荐系统
- 针对基于内容的推荐，提出了一种深度学习方法。并学习不同的技术扩展推荐系统
- 结合不同领域的数据，提出了Multi-View DNN模型建立推荐系统
- Multi-View DNN模型可以解决用户冷启动问题
- 在四个真实的大规模数据集，通过严格的实验证明所提出的推荐系统的有效性

训练模型使用的数据集：

- User Features：用户在Bing搜索引擎中搜索的关键词和点击的链接数据（这里使用TF-IDF剔除掉不重要的词汇特征数据）
- News Features：用户在Bing News中的新闻点击数据，每篇新闻选择了三部分特征（新闻的标题且使用letter tri-gram技术进行编码、新闻的类别数据编码为二进制的特征、新闻的命名实体词提取然后使用letter tri-gram技术进行编码）
- App Features：用户在Windows AppStore中的下载历史数据，每个APP的的标题（通过letter tri-gram技术进行编码）和APP的类别数据（编码为二进制的特征）
- Movie/TV Features：用户在XBOX上的浏览数据，主要利用的是标题、描述、流派

在MV-DNN中，userfeatures 被用来作为user view，其余的特征映射到不同的view中，训练时一个user features中的样本和一个其他的样本形成样本对，为了达到这个训练目的，使用微软公司的唯一用户ID进行关联，News Features、App Features、Movie/TV Features分别和User Features中的进行关联，能够关联上的数据条数如下表所示：

Type	DataSet	UserCnt	Feature Size	Joint Users
User View	Search	20M	3.5M	/
Item View	News	5M	100K	1.5M
	Apps	1M	50K	210K
	Movie/TV	60K	50K	16K

Table 1: Statistics of the four data sets used in this paper.

The *Joint Users* column indicates the number of common users between each item view and the user view.

训练数据关联

MV-DNN 里面的 MV 为 Multi-View，一般可以理解为多视角的 DSSM，在原始的DSSM中需要训练的有 Query 和 Doc这两类的embedding，同时里面DNN的所有权重都是共享的，而MV-DSSM他可以训练不止两类的训练数据，同时里面的深度模型的参数是相互独立：

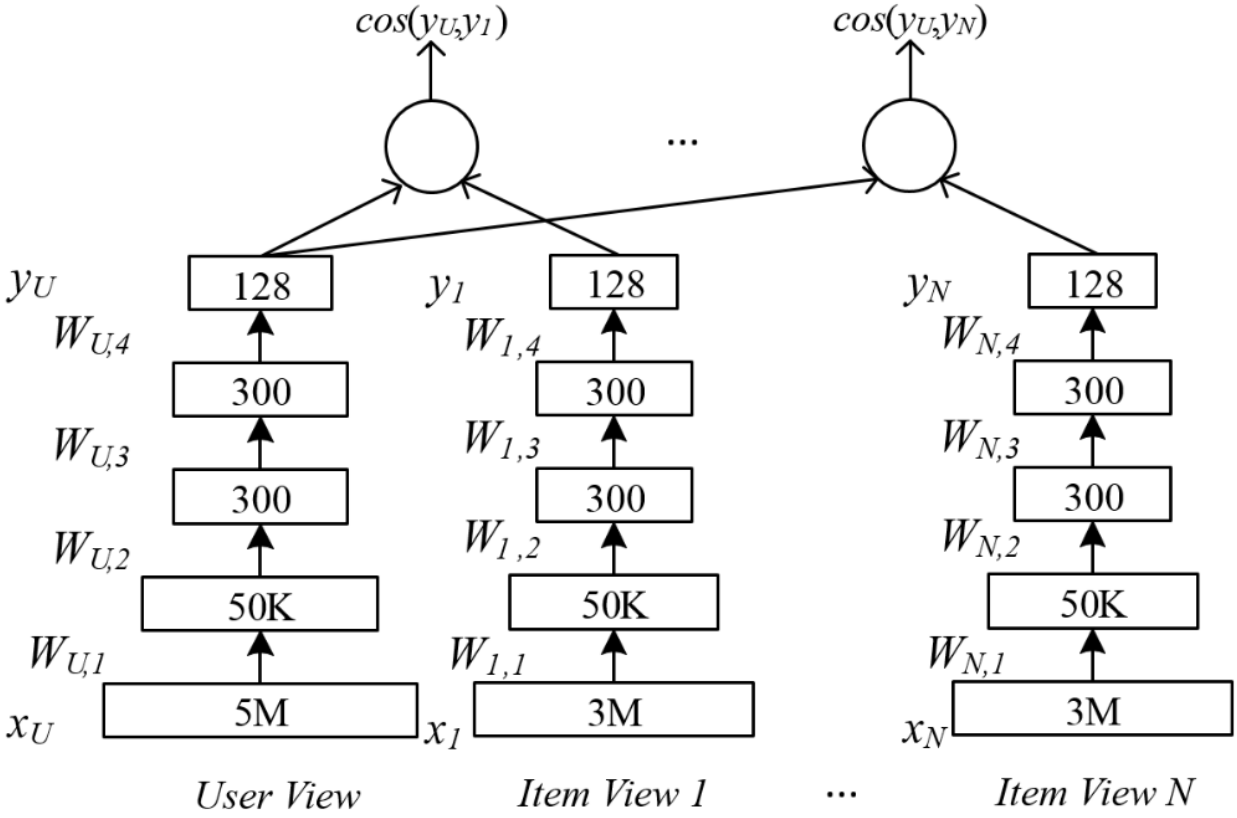


Figure 2: Multi-view DNN for multiple domain recommendation. It uses a DNN to map high-dimensional sparse features (e.g., raw features of users, News, App) into low-dimensional dense features in a joint semantic space.. The first hidden layer, with 50k units, accomplishes word hashing. The word-hashed 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. Note that the input feature dimension x (5M and 3M) in this figure is hypothetical as in practice each view can have arbitrary number of features. See text for details.

MV-DNN

基于Multi-View的DSSM的参数变多了，由于是多视角的训练，输入的语料也变得不同，自由度更大，但是训练时也会变得更加困难。

MV-DNN的训练过程为：

Algorithm 1 Training Multi-View DNN

Input: N = # of view pairs, M = # of training iterations,
U_A = user view architecture,
$I_A = \{I_{A1}, ...I_{AN}\}$ item view architecture,
$U_D = \{U_{D1}, ...UDN\}$ user input files,
$I_D = \{I_{D1}, ...I_{DN}\}$ item input files,
$\mathbf{W_U}$ = user view weight matrix,
$\mathbf{W_I} = \{W_{I1}, ...W_{IN}\}$ item view weight matrices
Initialization
Initialize $\mathbf{W_U}$ and $\mathbf{W_I}$ using U_A and I_A
for $m = 1$ to M
  for $v = 1$ to N
    $T_U \leftarrow U_{Dv}$
    $T_I \leftarrow I_{Dv}$
    train $\mathbf{W_U}$ and $\mathbf{W_I}$ using T_U and T_I
  end for
end for
Output: $\mathbf{W_U}$ = final user weight matrix,
$\mathbf{W_I}$ = final set of item view weight matrices

MV-DNN的训练过程

训练目标为：

$$p = arg\max_{W_u, W_1 ... W_v} \sum_{j=1}^{N} \frac{exp(a_\alpha \cos(Y_u, Y_{a,j}))}{\sum_{X' \in R^d a} exp(a \cos(Y_u, f_a(X', W_a)))}$$

论文中提到的几种降低特征维度的方法：

- 通过TF-IDF选取top K features
- 使用KMeans将相似度的特征划分到一个cluster中，并结合最终的clusters生成新的特征
- 局部敏感哈希：通过一个随机的矩阵将数据映射到低维向量空间中，并且保持原始空间上的pairwise cos距离再新的空间上仍然获得保留

> 假设原始维度为d，降维为k，那么映射矩阵$A \in R^{d*k}$，即A包含了k个映射，每个映射A_i都将X映射为Y_i，输出为$Y \in R^k$，计算Y_i的公式为：

$$Y_i = \begin{cases} 1 & if A_i X >= 0 \\ 0 & else \end{cases}$$

> 计算X_1 、 X_2的cos相似度近似表示为：$cos(\frac{H(Y_1,Y_2)}{k}\pi)$，其中$H(Y_1,Y_2)$表示汉明距离

- 减小训练样本数：每个用户在每个域都有大量的日志数据，每个用户在每个域只选择一个<user, item>对，具体为用户特征-用户在此域喜欢的所有item的平均分数

Google Two Tower Model

双塔模型是应用非常普遍的深度学习召回模型，但是其本身也存在一些问题，比如使用softmax函数计算后验概率会带来采样偏差，popular item在负采样时出现的频率也会比较高等。

而Google2019年提出的双塔模型则主要是为了解决softmax带来的采样偏差问题和popular item的修正问题，主要是通过两种方法进行修正：

- in-batch softmax
- streaming frequency estimation

该论文的主要贡献点：

- 提出流数据频率估计方法（streaming frequency estimation）：针对流数据来估计item出现的频率，并利用实验分析估计结果的偏差与方差，模拟实验证明该方法在数据动态变化时的功效
- 提出模型架构：提供了一个针对大规模的检索推荐系统，包括了in-batch softmax损失函数与流数据频率估计方法，减少了负采样在每个batch中可能会出现的采样偏差问题
- YouTube推荐：在YouTube上应用了该模型框架，实现了端到端的推荐
- 线下和真实实验：利用两个数据集实验，检验模型效果

算法原理

利用双塔模型构架推荐系统，Queries特征向量$\{x_i\}_{i=1}^N$ item特征向量$\{y_i\}_{i=1}^M$ ，目标是给定一个query，检索到一系列item子集用于后续排序推荐任务。模型结构如图所示：

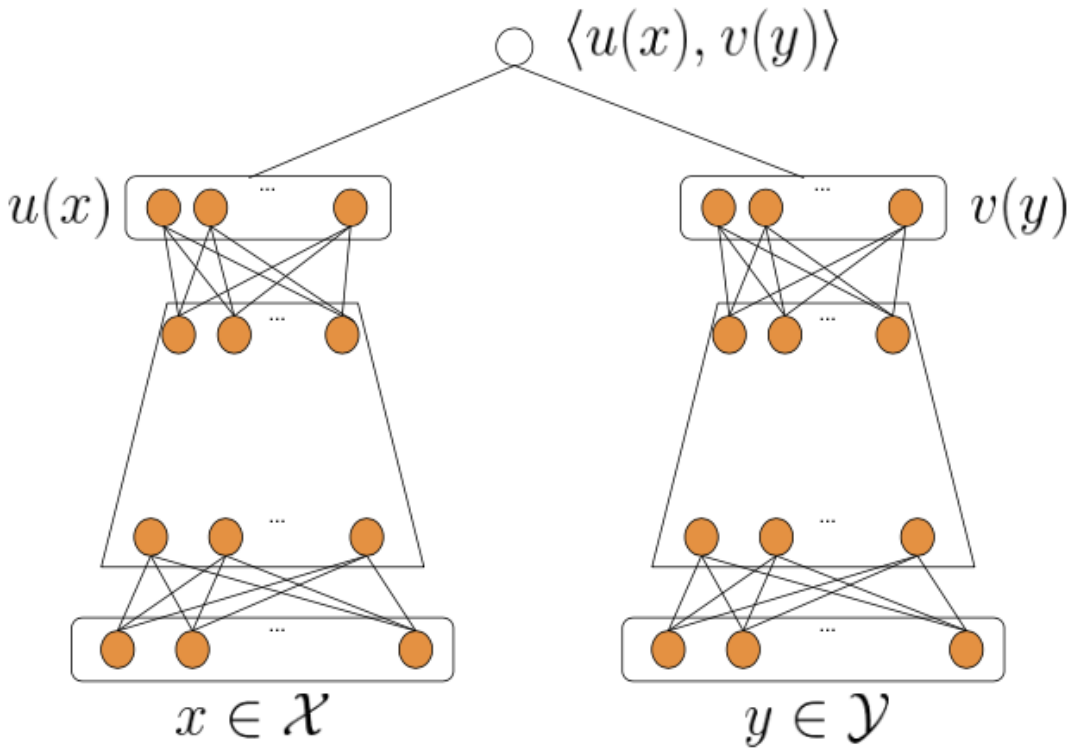


Figure 1: A two-tower DNN model for learning query and candidate representations.

首先建立两个参数embedding函数， $u : X \times R^d \to R^k, v : y \times R^d \to R^k$ ，把query和候选item映射到k维向量空间，模型的输出为二者的embedding内积，即：$s(x,y) =
，我们的目标是根据训练集T:= \left \{ (x_i, y_i, r_i) \right \}_{i=1}^{T}来学习参数\theta(其中r_i$ 为用户反馈，比如说用户花在一个视频上的时间等)

in-batch loss function

检索问题可以看作是一个多分类问题，给定query x，从结果M个结果中$\{y_i\}_{j=1}^{M}$得到y，概率通过softmax函数得到：

$$p(y|x;\theta) = \frac{e^{s(x,y)}}{\sum_{j \in [M]} e^{s(x,y)}}$$

结合用户的反馈r_i，对数似然损失函数加权得到：

$$L_T(\theta) := -\frac{1}{T} \sum_{i \in [T]} r_i \cdot log(p(y_i|x_i;\theta))$$

当M非常大时，我们通常可以利用负采样算法进行计算。然而对于流数据，我们考虑在同一个batch中采样负样本，batch-softmax函数为：

$$p(y_i|x_i;\theta) = \frac{e^{s(x_i,y_i)}}{\sum_{j \in [B]} e^{s(x_i,y_i)}}$$

在每个batch中，由于存在幂律分布现象，即如果在每个batch中随机采样负样本，会使热门商品更容易被采样到，在损失函数中就“过度”惩罚了这些热门商品，因此考虑用频率对采样进行修正，即：

$$s^c(x_i, y_i) = s(x_i, y_i) - log(p_j)$$

其中p_j是在每个batch中随机采样到item j的概率(下文会有说明)，因此修正后的条件概率函数为：

$$P_B^c(y_i|x_i;\theta) = \frac{e^{s^c(x_i,y_i)}}{e^{s^c(x_i,y_i)} + \sum_{j\in[B],j\neq i} e^{s^c(x_i,y_i)}}$$

则损失函数为：

$$L_B(\theta) := -\frac{1}{B}\sum_{i\in[T]} r_i \cdot log(P_B^c(y_i|x_i;\theta))$$

即为batch loss function，然后可以利用SGD来更新参数 θ。

两个技巧

- 最近邻搜索：当embedding映射函数u和v学习好后，预测包含两步：

  考虑到耗时问题，此处利用hash技术采用近邻搜索等方法进行处理

  - 1、计算query的向量$u(x,\theta)$
  - 2、从事先训练好的函数v中找到最邻近的item。

- 归一化：经验表明，对函数归一化效果更好

  即：$u(x,\theta) = \frac{u(x,\theta)}{||u(x,\theta)||_2}, v(x,\theta) = \frac{v(x,\theta)}{||v(x,\theta)||_2}$，对每个logit函数，利用超参数 τ 进行处理：$s(x,y) = \frac{s(x,y)}{\tau}$

Streaming Frequency Estimaton

此方法用来估计流数据中，batch 数据下 item 出现的概率。如果一个item每隔50步出现一次，则对应的概率为1/50=0.02，按照这样的想法，针对流数据，利用哈希序列来记录采样id(暂时不考虑hash collision的问题)。

定义两个大小为H的数组A，B，哈希函数h可以把每个item映射为[H]内的整数。

- A[h(y)]表示item y上次被采样到的时刻
- B[h(y)]表示每多少步item y可以被采样一次

则，当第t步y被采样到时，利用迭代可更新A，B：

$$B[h(y)] = (1-\alpha)B[h(y)] + \alpha(t - A[h(y)])$$

α可看作学习率。通过上式更新后，则在每个batch中item y出现的概率为 $1/B[h(y)]$ 。

直观上，上式可以看作利用SGD算法和固定的学习率 α 来学习“可以多久被采样到一次”这个随机变量的均值。

> 具体证明和推断过程参考论文。

模型结构

Google在YouTube上进行了模型的测试，模型结构如下图所示，其中：

- 训练标签：点击并进行了完整观看的为1，只点击或者观看很少的认为是0
- Video特征：视频ID、频道ID，转化为Embedding，对于一些多值特征（比如视频的主题）采用Embedding加权平均
- User特征：历史观看视频的平均Embedding（这里选取的是过去的一段时间）
- 对于同一个ID的embedding在双塔模型的左侧和右侧是共享的

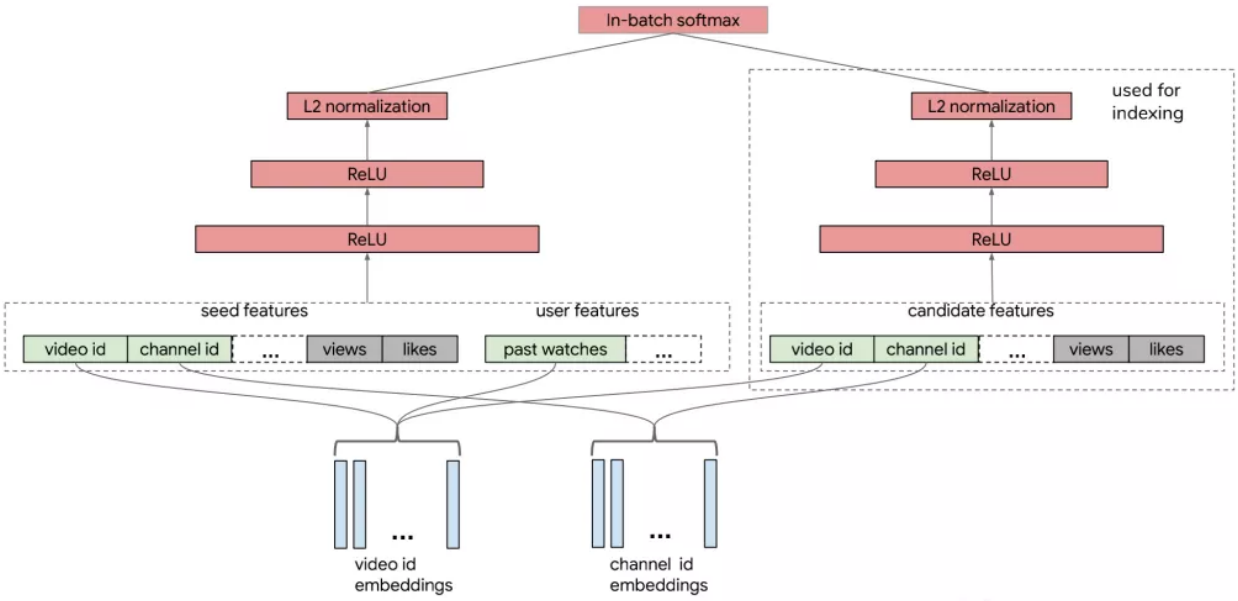
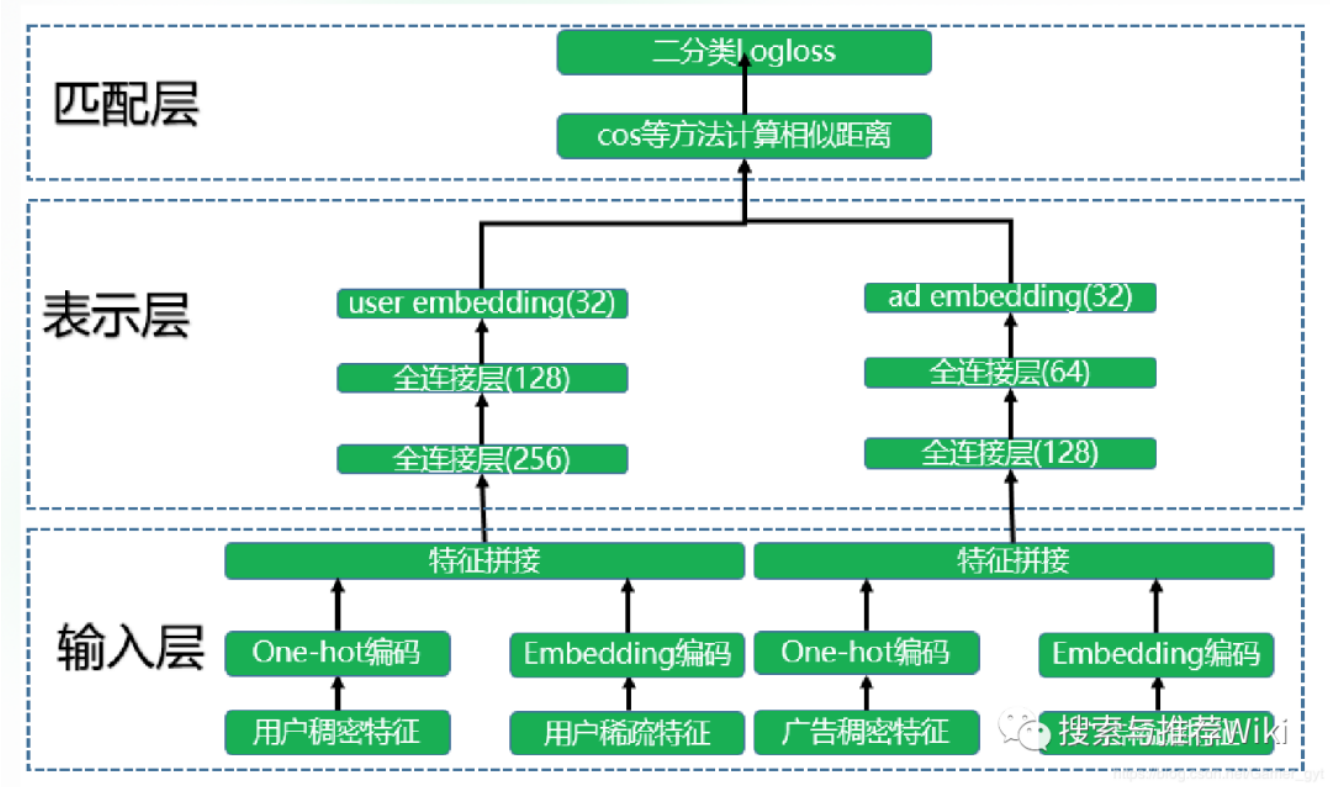


Figure 2: Illustration of the Neural Retrieval Model for YouTube.

模型结构

广告场景中的DSSM双塔模型

模型整体结构如下图所示，也分成三层：输入层、表示层和匹配层。



输入层

模型训练分成两座不同的“塔”分别进行，其实也就是两个不同的神经网络。其中一座塔是用于生成user embedding。输入用户特征训练数据，用户特征包括用户稠密特征和用户稀疏特征，其中用户稠密特征进行one-hot编码操作，用户稀疏特征进行embedding降维到低维空间(64或者32维)，然后进行特征拼接操作。广告侧和用户侧类似。关于里面的特征，不在于你要什么，而在于你有什么。整个工程超级复杂的就是这块的特征工作。这里不再赘述。

表示层

得到拼接好的特征之后会提供给各自的深度学习网络模型。用户特征和广告特征经过各自的两个全连接层后转化成了固定长度的向量，这里得到了维度相同的user embedding和ad embedding。各塔内部的网络层数和维度可以不同，但是输出的维度必须是一样的，这样才能在匹配层进行运算。项目中user embedding和ad embedding 维度都是32。

匹配层

模型训练好了之后会分别得到user embedding和ad embedding，将它们存储到redis这一类内存数据库中。如果要为某个特定的广告推荐人群，则将该广告的ad embedding分别和所有人群的user embedding计算cos相似度。选择距离最近的N个人群子集作为广告投放人群，这样就完成了广告推荐任务。模型训练过程中将cos函数得到的结果进入sigmoid函数和真实标签计算logloss，查看网络是否收敛。模型评估主要使用auc指标。小结下，本节讲了下我们使用DSSM双塔模型完成广告推荐任务。模型整体结构分成输入层、表示层和匹配层。首先在输入层处理数据获取特征；然后在表示层通过深度学习网络得到user embedding和ad embedding；最后在匹配层进行广告推荐。

一些思考

DSSM双塔模型有很多变种，比如CNN-DSSM、LSTM-DSSM等等。项目中表示层使用了两层全连接网络来作为特征抽取器。现在深度学习领域公认最强的特征抽取器是Transformer，后续是否可以加入Transformer。

总结

至此，从DSSM语义匹配模型到双塔模型召回的整体演变过程已经介绍完毕，在整理该篇内容的过程中看了好几篇论文，虽然追求认真和无错，但还是会存在错误的地方，欢迎在评论区指正一起交流。

同时也参考了很多优秀的文章和思路，如下：

- 深度召回模型在QQ看点推荐中的应用实践
- 深度语义模型以及在淘宝搜索中的应用
- 百度NLP | 神经网络语义匹配技术
- 【推荐系统经典论文(九)】谷歌双塔模型
- 广告行业中那些趣事系列10：推荐系统中不得不说的DSSM双塔模型

The end