DSIN分享

1. 引入：

- **现如今的工业化的RS主要包含两个stage：**
  - **candidate generation**
  - **candidate ranking。**

    而我今天要介绍的**DSIN**（Deep Session Interest Network）便是后者中的一个模型。

- **那什么是candidate ranking?**

  他做的事情就是对一堆东西根据某个规则进行**排序**，挑出top-K来进行推荐。

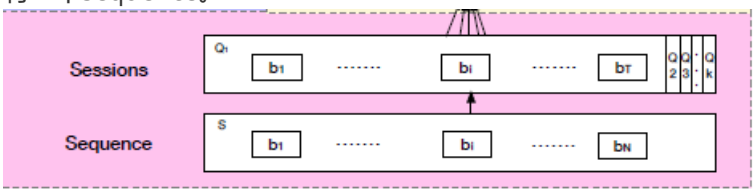
  而对于DSIN，他是**CTR（click-through rate） predication task**，也就是根据**点击率预测数据的高低**来进行candidate ranking，高概率会被点击的进行推荐。

- **CTR的想法很直觉，那么和别的CTR模型比，DSIN不一样在那里?**
  - 简单来说就是不同在他的**S**，也就是**DSIN利用了session来提高效果。**
  - 虽然可能session这个想法并不是复杂的。但事实上，session的想法在CTR prediction任务中很少被用到，他更多是用在sequential recommendation。（文章说法，待考证）

    2.2 Session-based Recommendation

    The concept of session is commonly mentioned in sequential recommendation but rare in the CTR prediction task.
  - 根据相关的介绍，虽然目前绝大多数的CTR都是从用户历史行为数据中建模他们动态、兴趣特征。
    但是大多数都停留到**behavior sequence**的阶段便不再分析下去了。（文章说法，待考证）

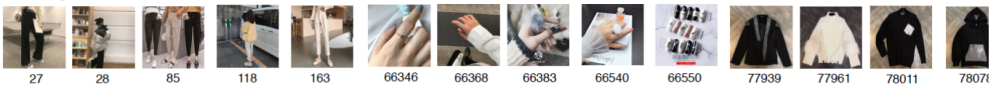
    However, these models overlook the intrinsic structure of the sequences: the sequences are composed of sessions. A
  - 虽然sessions和sequence性质上都是由多个behaviors组成。但是session是根据一定规则对sequence进行划分的结果。
    所以可以看到，图中上下两个方框其实是一样的内容，但上面是k个session组成，而下面只有一个sequence。

- **session的划分规则**
  根据用户的点击时间：**前一个session最后一个behavior到后一个session第一个behavior的时间差大于等于30min** [Grbovic and Cheng, 2018]。

- **采用session而非单纯sequence的优势?**
  - 首先要强调一下，不论是sessions之间还是sequences之间，在目前考虑的CTR问题中他们都是**有顺序的**，这样做的目的也很直接：方便对用户时序的兴趣变化进行建模。

- 从***sequence***的角度我们可能会看到：



- 但是如果从***session***的角度：

- 从sequence的角度我们虽然能看到interest的变化过程，但是却忽视了这样两个事实：
  1. **同一个session内的行为高度同构**
  2. **不同sessions间的behavior异构**

> 这样的说法很好理解，比如我们早上想买手机，我们可能在一个连续时间段中一直在看各类牌子的手机，然后偶尔看一下手机壳。但我们主要是在挑手机，大多数行为是同构的。
> 但是可能下午我想买书，晚上我想买衣服。那么这些session间的行为是异构的是

- 基于这样的观察，作者构造了DSIN来更好地利用session处理CTR prediction task

2. 模型剖析：

基本的CTR模型（Base Model）

- 在开始看DSIN模型前，我们先来了解一下基本的CTR模型：
  - 一般的CTR模型的**input**有三个：
    - User Profile： 性别、城市、用户ID
    - User Behavior：用户最近点击的物品ID序列
    - Item Profile： 商家ID、品牌ID
  - 很显然上面三类input是高维稀疏特征，所以分别做**embedding**将他们从变成低维密集的特征。

$$\mathbb{R}^{M \times d_{model}}$$

M是这类input特征的数目，**d_model**是embedding后每个vector的size

  - 然后将dense features进行**concatenate和flatten**，再送入**MLP**（一般使用**RELU**作为激活函数，**softmax**来给出输出）
  - 他的损失函数是常见的**负对数似然**（最大似然估计取反）

$$L = -\frac{1}{N}\sum_{(x,y)\in\mathbb{D}} (y\ \log p(x) + (1-y)\ \log(1-p(x))) \quad (1)$$

D是training set；**x**是三类input在embedding的数据[Xu，Xi，S]；（即User Profile，Item Profile，User Behavior）

y是{0,1}的数据，反应实际用户是否真实点击了；**p()**是我们计算出的用户点击的概率

- 说了这么多，我们应该对CTR有一定认识了，那么在看DSIN结构前，让我们再回顾一下**DSIN的精神**：
  找出sessions的interest并推导出这些interests的sequential关系

> tially related to each other. DSIN is proposed to extract users' session interest in each session and capture the sequential relationship of session interests.

好啦，接下来我们终于要开始看DSIN的框架了☺

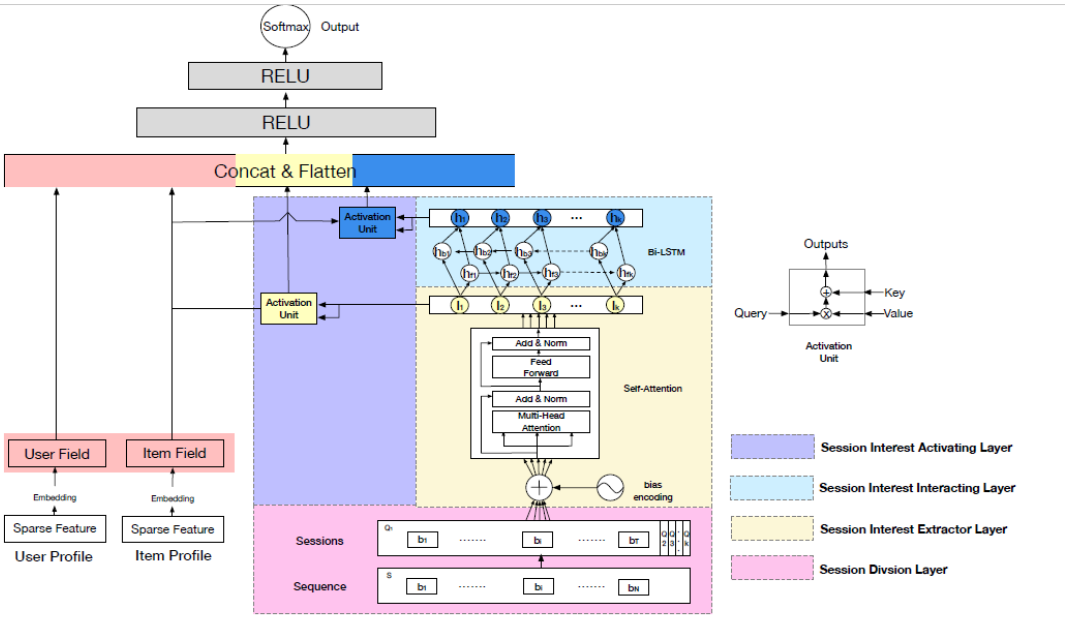
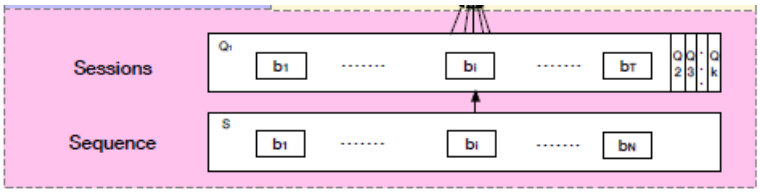


Figure 2: The overview of our proposed model DSIN. Overall, before MLP layers, DSIN has two main components. One is sparse features and the other processes the user behavior sequence. From the bottom up, the user behavior sequence **S** is first divided into sessions **Q**, which are then added with bias encoding and extracted into session interests **I** with self-attention. With Bi-LSTM, we mix session interests **I** with contextual information as hidden states **H**. Both Vectors of session interests **I** and hidden states **H** activated by the target item and embedding vectors of *User Profile* and *Item Profile* are concatenated, flattened and then fed into MLP layers for the final prediction.

2.1 Session Division Layer

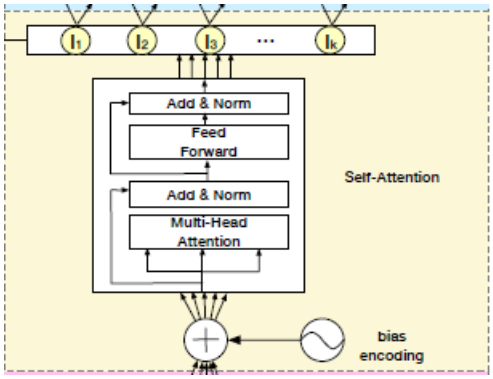


首先是最下面的division层，他做的事情之前已经解释过了，就是根据规则将sequence S划分成sessions Qk

$\mathbf{Q}_k = [\mathbf{b}_1; ...; \mathbf{b}_i; ...; \mathbf{b}_T] \in \mathbb{R}^{T \times d_{model}}$,

T 是 每个session中behaviors 数目；***d_model***是embedding后每个behavior的size；***bi*** 是session中第i个behavior.

2.2 Session Interest Extractor Layer



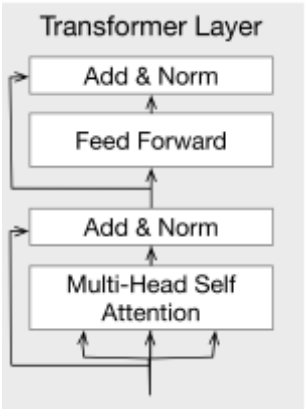
然后我们来看division层上面的interest extractor层。 这一层的目的是**寻找session内部的行为之间关系，来进一步提取session interest**

- 就像之前提到的那样，即使同个session中的行为是高度同构的，但是还是会有一些随意的一些行为会使得session的兴趣表示变得不准确：比如我想买衣服，看着看着衣服，但不小心误触到其他推送内容，但那和我的真实想去点击的东西无关。
- 所以为了**对同一session中多个行为的关系进行建模**和**减轻那些不相关行为的影响。** DSIN对每个session都使用**Transformer**中的**multi-head self-attention**模块来抽取用户session的兴趣特征。 （TODO：Transformer里面的东西还是一知半解）
- 但是我们可以看到，之前division层的输出在进入self-attention之前被做了***bias encoding***，这是什么意思呢？🤨
    - 对于一般的self-attention，输入的数据要先做position encoding来利用sequence之间的位置关系的。在这里，对于这类position encoding（**PE**）做出了优化，命名为bias encoding（**BE**）。
    - 过去的**PE** 对sequence中每个behavior，以及每个behavior对应的embedding vector的每个unit，进行了处理 。
      但在**BE**，我们输入的是多个sessions，所以除了对**PE**中两个内容进行处理外，还需要对每个session的位置进行处理
      所以对于第k个session中，第t个behavior的embedding vector的第c个unit的偏置项**BE**如下：

      $$\mathbf{BE}_{(k,t,c)} = \mathbf{w}_k^K + \mathbf{w}_t^T + \mathbf{w}_c^C$$

      其中第一项为session的bias vector，第二项为behavior的bias vector，第三项为unit的bias vector

      $$\mathbf{Q} = \mathbf{Q} + \mathbf{BE}$$
- 然后我们将经过**Bias Encoding**处理的输入传入**transformer**结构里 来提取session内部interest
- 这里面**Multi-head Self Attention**其实就是多个Self-Attention结构的结合，每个head学习到在不同表示空间中的特征，如下图所示，两个head学习到的Attention侧重点可能略有不同，这样给了模型更大的容量。



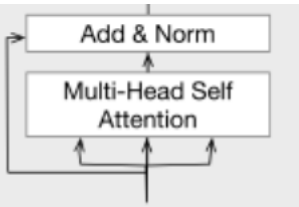
> 这张图是阿里BST里的截图，可以看到multi-head self-attention 要输出到add&norm
> 然而DSIN的模型上面画错了，他漏掉了这个箭头
> 其实这篇文章还是有蛮多typo的。。。

- 再来看看模型中这些**Add & Norm**模块的作用。
  其中Add代表了Residual Connection，是为了解决多层神经网络训练困难的问题，通过将前一层的信息无差的传递到下一层，可以有效的仅关注差异部分，这一方法之前在图像处理结构如ResNet等中常常用到。 （TODO：残差网络，还没看完）
  而Norm则代表了Layer Normalization，通过对层的激活值的归一化，可以加速模型的训练过程，使其更快的收敛 。
- 那么Multi-head self attention在具体做的时候是怎样的呢?
  - 因为用户的点击行为会受各种因素影响，比如颜色、款式和价格，那么这些因素就是作为所谓的head。所以第一步就是将session Q根据H个head划分为H份。Multi-head self attention模块可以在不同的表示子空间层面上建模这种关系
    subspaces. Mathematically, let $\mathbf{Q}_k = [\mathbf{Q}_{k1}; ...; \mathbf{Q}_{kh}; ...; \mathbf{Q}_{kH}]$ where $\mathbf{Q}_{kh} \in \mathbb{R}^{T \times d_h}$ is the h-th head of $\mathbf{Q}_k$, H is the number of heads and $d_h = \frac{1}{h} d_{model}$. The output of $\mathbf{head}_h$ is

    这里d_h是d_model的1/H；就可以理解成embedding后的vector被H个head平均分配了。（存疑）
  - 在对Q划分完后就要对每个head去计算，下面是他的计算方式：

$$\begin{aligned} \mathbf{head}_h &= \text{Attention}(\mathbf{Q}_{kh}\mathbf{W}^Q, \mathbf{Q}_{kh}\mathbf{W}^K, \mathbf{Q}_{kh}\mathbf{W}^V) \\ &= \text{softmax}(\frac{\mathbf{Q}_{kh}\mathbf{W}^Q {\mathbf{W}^K}^T \mathbf{Q}_{kh}^T}{\sqrt{d_{model}}})\mathbf{Q}_{kh}\mathbf{W}^V \end{aligned} \quad (4$$

    论文在这面是一笔带过的，但是如果我们不熟悉transformer或者说是self-attention，其实这个是有些迷惑的
    我们再看一下这个结构：



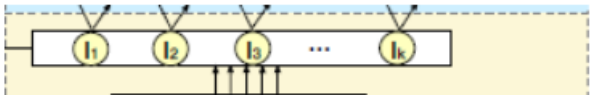
    其实在multi-head下面有三个箭头其实是比较细节的。
    因为对于self-attention来讲，这三个输入量分别为：Q(Query), K(Key), V(Value)三个矩阵。他们均来自同一输入，首先我们要计算Q与K之间的点乘，然后为了防止其结果过大，会除以一个尺度标度 $\sqrt{d_k}$ ，其中 d_k 为一个query和key向量的维度。再利用Softmax操作将其结果归一化为概率分布，然后再乘以矩阵V就得到权重求和的表示。
    该操作可以表示为 $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$
    现在再看DSIN的公式是不是会简单些？ 稍有不同的 W^Q, W^K, W^V 是我们模型训练过程学习到的合适的参数
  - 再将这些结果和之前的输入放入add&norm后放入FFN，然后再做add&norm，得到**小interest**。这里的add&norm作用应该也和前一个一样。W^o应该和前面的类似也是学出来的？（存疑）

$$\mathbf{I}_k^Q = \text{FFN}(\text{Concat}(\mathbf{head}_1, ..., \mathbf{head}_H)\mathbf{W}^O)$$

    最后得出第k个session的interest是对之前的各个**小interest**做average pooling为：

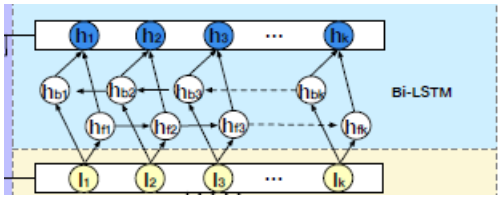
$$\mathbf{I}_k = \text{Avg}(\mathbf{I}_k^Q)$$

- 最后得出的每个session的interest：



2.3 Session Interest Interacting Layer

- 在提取出每个session的interest后，很自然就会想要去捕获不同session之间的顺序关系，那么这里DSIN使用了Bi-LSTM来做这件事。



- 那么对于Bi-LSTM，每个时刻的hidden state Ht计算如下：

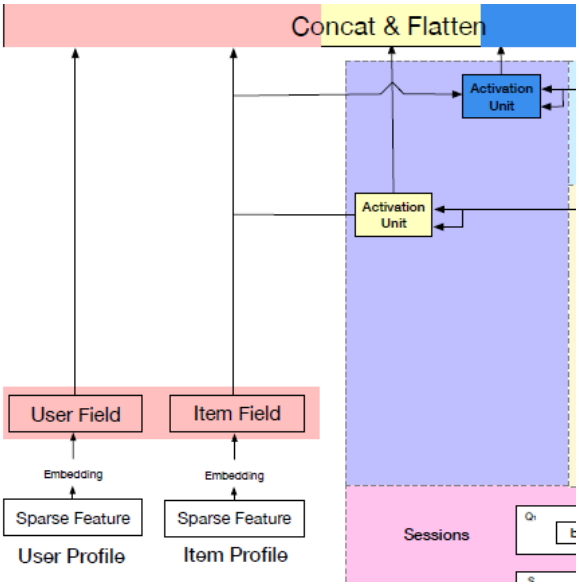
$$\mathbf{H}_t = \overrightarrow{\mathbf{h}_{ft}} \oplus \overleftarrow{\mathbf{h}_{bt}}$$

等式右边两项分别是前向传播和反向传播对应的t时刻的hidden state。
一共有k个H_t，代表了参考前后memory的某个session 的interest。（对应图上的ht）

> 这里的符号是异或还是加？（存疑）
>
> 这里文章也说的很简单，我能理解这层干了什么事情，但不知道有没有遗漏些什么😮

那么，到这为止，模型已经同时捕获到了session内部和session之间的顺序关系。

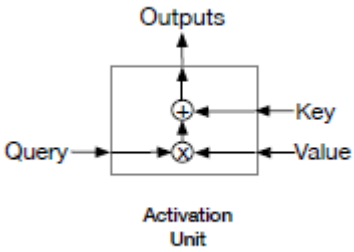
2.4 Session Interest Activating Layer



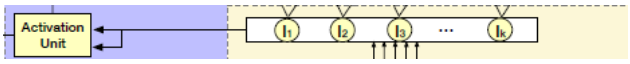
（图中紫色部分）

> 这个layer给我的感觉有点像是建立用户画像的感觉，也不知道对不对😶

- 这一层大家一看，诶，只有两个activation unit。好像很简单的样子，那什么是activation unit呢？



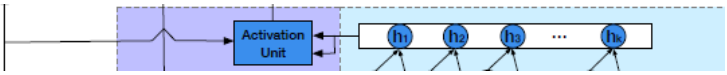
  - 首先就单纯看他的长相，左边输入的是被embedding的商品信息，右边两个都是用户的interest信息，然后给出一个输出。
  - 就我目前的观察而言，activation unit是个蛮常见的东西，至少他在阿里关于CTR的模型中是个常客。
  - 实际上，它的作用类似一个开关，控制了这类特征是否放入特征向量以及放入时权重的大小。
  - 那这个开关由谁控制呢？它是由被预测item跟session interest的关系决定的。也就是说，在预测用户u是否喜欢商品i这件事上，我们只把跟商品i有关的特征考虑进来，其他特征会完全不考虑或者权重很小。
  - 这是比较好理解的，用户的session interest与target item关系越密切，对用户会去点击target item的影响越大，因此这样的session interest应该赋予更大的权重。
- 这样做了之后，对于session内 interest做一下改进：



$$a_k^I = \frac{\exp(\mathbf{I}_k \mathbf{W}^I \mathbf{X}^I))}{\sum_k^K \exp(\mathbf{I}_k \mathbf{W}^I \mathbf{X}^I)}$$
$$\mathbf{U}^I = \sum_k^K a_k^I \mathbf{I}_k \qquad (9)$$

得出U^I

- 对跨session的interest做一下改进：



$$a_k^H = \frac{\exp(\mathbf{H}_k \mathbf{W}^H \mathbf{X}^I))}{\sum_k^K \exp(\mathbf{H}_k \mathbf{W}^H \mathbf{X}^I)}$$
$$\mathbf{U}^H = \sum_k^K a_k^H \mathbf{H}_k$$

得出U^H

> 我记得好像在别的文章里看到过说activation的key，value在self -attention里面默认是一样的就是Ht或者I,不知道是不是这样。

query就是公式中的X，就是目标item的embedding

- 之后就是把四部分的向量：用户特征向量、待推荐物品向量、会话兴趣加权向量UI、带上下文信息的会话兴趣加权向量UH进行横向拼接，输入到全连接层中，得到输出

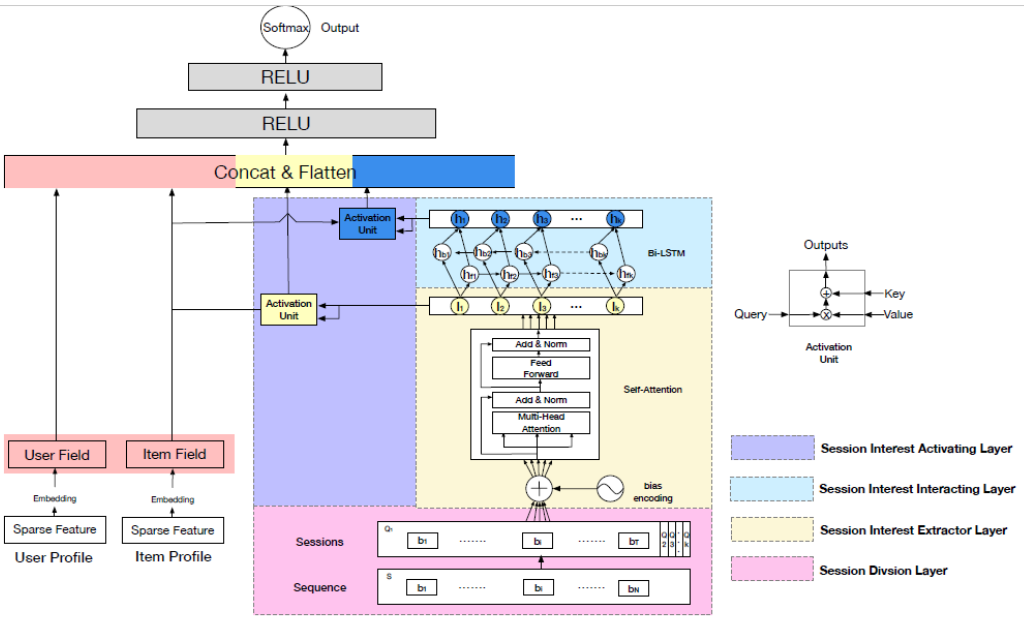


Figure 2: The overview of our proposed model DSIN. Overall, before MLP layers, DSIN has two main components. One is sparse features and the other processes the user behavior sequence. From the bottom up, the user behavior sequence **S** is first divided into sessions **Q**, which are then added with bias encoding and extracted into session interests **I** with self-attention. With Bi-LSTM, we mix session interests **I** with contextual information as hidden states **H**. Both Vectors of session interests **I** and hidden states **H** activated by the target item and embedding vectors of *User Profile* and *Item Profile* are concatenated, flattened and then fed into MLP layers for the final prediction.

实验分析：

Model	Advertising	Recommender
YoutubeNet-NO-UB[a]	0.6239	0.6419
YoutubeNet	0.6313	0.6425
DIN-RNN	0.6319	0.6435
Wide&Deep	0.6326	0.6432
DIN	0.6330	0.6459
DIEN	0.6343	0.6473
DSIN-PE[b]	0.6357	0.6494
DSIN-BE-NO-SIIL[c]	0.6365	0.6499
DSIN-BE[d]	**0.6375**	**0.6515**

[a] YoutubeNet without *User Behavior*.
[b] DSIN with positional encoding.
[c] DSIN with bias encoding and without session interest interacting layer and the corresponding activation unit.
[d] DSIN with bias encoding.

DSIN在广告和推荐两个数据集上的表现都比其他先进的CTR模型效果好

todo：讲一下这些DSIN的不同

评价标准为AUC:也就是正样本排在负样本前面的概率

$$\text{AUC} = \frac{1}{m^+ m^-} \sum_{x^+ \in D+} \sum_{x^- \in D-} (I(f(x^+) > f(x^-))) \tag{11}$$

3. 思考与总结：

Q1：在interest extractor层中的mult-head self attention，是根据头将vector平均分为H份吗？如果是，为什么可以均分呢？

Q2：在interest extractor层中的mult-head self attention，如果能做average pooling，I_k^Q是一个vector吗？还是说有很多个I_k^Q？但是I_k^Q好像已经包含了H个头的信息了。。。

Q3：Session Interest Interacting Layer的隐藏层公式里的负号代表相加还是异或?

Q4：文章里的公式有好多的***W***，文章只是说他们是linear matrix，我的理解是待训练的权重，这样想对吗🤔?

为什么找这篇文章呢?

首先是看了abstract后感觉和我们的研究问题（session-based recommendation）蛮相近的，又是发表在IJCAI 2019的文章，还有代码提供，所以就选择了它。读完后感觉还是一篇蛮有代表性的，方法蛮新的文章。读的过程中发现是DSIN从阿里的DIN，DIEN之类的演化而来的，所以感觉是不是读一篇顶三篇呀🤭？不过这三篇的模型长得就真的是很像。。。

对我帮助?

- 虽然这篇文章没有用什么很复杂的数学或者很复杂的模型，是一篇很工业化的文章，但是他作为我第一篇精读的推荐系统文章，我感觉还是学到了蛮多的东西。比如CTR、session-based recommendation、transformer等等😉。
- 如果说有什么想法的话，好像目前没什么感觉。记得沈老师之前有说过主要思考的问题有冷处理、利用相似的人物画像来群组推荐、考虑用户多久之前的memory...（有些记不清了，如果说错了，麻烦沈老师再纠正一下🤭）但是这样的文章真的好多呀，其实当时选论文读时就感觉有些迷失了，
- 我记得看知乎时有这样一个说法：做推荐系统就是"**揣摩人心**"，在机器学习的相关技术已经驾轻就熟了的时候，你反而应该从技术中跳出来，**站在用户的角度，去深度体验他们的想法，去发现他们想法中的偏好和习惯，再用你的大数据工具去验证它，用你的模型去模拟它。**

4.参考文章：

1. https://zhuanlan.zhihu.com/p/71695849
2. https://www.jianshu.com/p/82ccb10f9ede
3. https://blog.csdn.net/sxf1061926959/article/details/90680488
4. https://www.jianshu.com/p/6742d10b89a8
5. https://zhuanlan.zhihu.com/p/47282410
6. https://www.zhihu.com/question/46301335/answer/528177439
7. https://mp.weixin.qq.com/s/RLxWevVWHXgX-UcoxDS70w
8. https://blog.csdn.net/liweibin1994/article/details/79462554
9. https://zhuanlan.zhihu.com/p/48601882

   Thanks for these notes! Maybe I had read something else to writemy note，I'm also grateful to them.....😊

5.代码地址：

https://github.com/shenweichen/DSIN