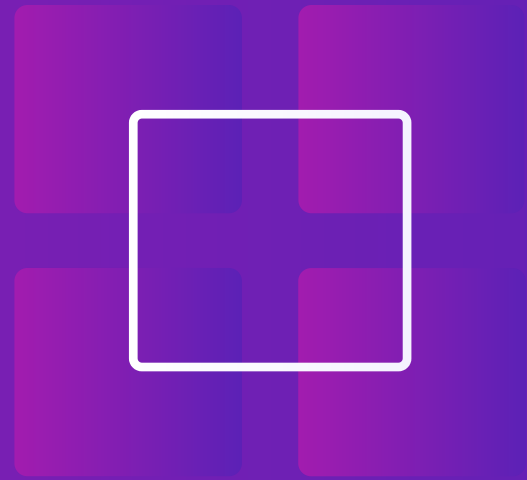
Behavior Sequence Transformer for E-commerce Recommendation in Alibaba

乔梁 2022.7.22

一、引言
二、架构
三、实验
四、总结
五、学习收获



一、引言

基于 DL 的 RSs

基于深度学习的方法已广泛应用于工业推荐系统（RSs）。以前的工作采用 Embedding+MLP 范式：将原始特征嵌入到低维向量中，然后将其馈送到 MLP 以进行最终推荐。

不足之处：这些工作大多只是串联不同的特征，忽略了用户行为的序列特征。在本文中，我们建议使用强大的 Transformer 模型来捕获用户行为序列背后的序列信号，以便在阿里巴巴中进行推荐。实验结果证明了该模型的优越性，并将其部署在淘宝网上，与两个基线模型相比，在线点击率（CTR）有显著提高。

背景

- 在过去十年中，推荐系统（RSs）一直是工业中最流行的应用。
- 在过去五年中，基于深度学习的方法被广泛应用于工业RSs，例如 Google 和 Airbnb。
- 在中国最大的电子商务平台阿里巴巴，RSs一直是其商品总量和收入的关键引擎，各种基于深度学习的推荐方法已部署在丰富的电子商务场景中。阿里巴巴的RSs分为两个阶段：匹配和排名。在匹配中，根据用户交互的项目选择一组相似的项目，然后学习微调预测模型来预测用户点击给定候选项目集的概率。

研究问题

在本文中，我们重点关注阿里巴巴旗下中国最大的消费者对消费者（C2C）平台——淘宝的排序阶段，我们在淘宝有数百万个候选项目，我们需要根据用户的历史行为预测用户点击候选项目的概率。在深度学习时代，Embedding 和 MLP 已成为工业RSs的标准范例：大量原始特征作为向量嵌入到低维空间中，然后馈送到全连接层，即多层感知器（MLP），以预测用户是否会单击某个项目。代表作品是谷歌的 Wide&Deep 和阿里巴巴的 DIN。

在淘宝，我们在 Wide&Deep 之上建立了排序模型，在 Embedding 和 MLP 范式中使用了各种特征，例如，项目的类别和品牌、项目的统计数字或用户配置文件特征。尽管该框架取得了成功，但它本身远远不能令人满意，因为它在实践中忽略了一种非常重要的信号，即用户行为序列背后的序列信号，即用户按顺序单击的项目。实际上，顺序对于预测用户未来的点击量至关重要。

例如，用户在淘宝上买了一部iphone后，往往会点击手机壳，或者在买了一条裤子后，试图找到合适的鞋子。从这个意义上讲，在淘宝排名阶段部署预测模型时，如果不考虑这一因素，就存在问题。在 Wide&Deep 中，它们只是连接所有特征，而不捕获用户行为序列中的顺序信息。在 DIN 中，他们提出使用注意力机制来捕捉候选项目和用户先前点击的项目之间的相似性，而没有考虑用户行为序列背后的序列性质。

本文思路

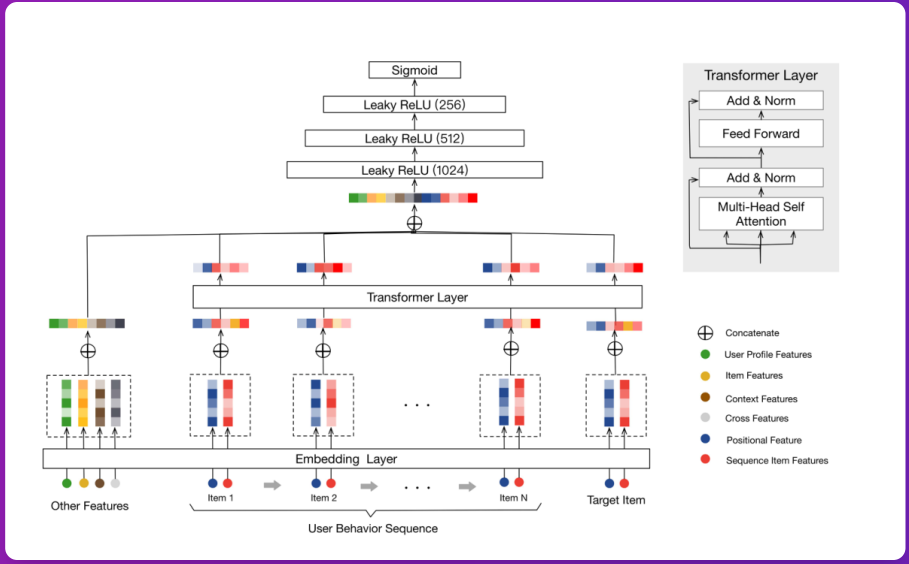
因此，在这项工作中，为了解决 Wide&Deep 和 DIN 面临的上述问题，我们尝试将用户行为序列的序列信号纳入淘宝的RS中。受自然语言处理中机器翻译任务转换器的巨大成功的启发，我们应用自注意力机制，通过考虑嵌入阶段的序列信息，为用户行为序列中的每个项目学习更好的表示，然后将其输入 MLP，以预测用户对候选项目的响应。Transformer 的主要优点是，它可以通过自注意力机制更好地捕捉句子中单词之间的依赖关系，直观地说，Transformer 还可以提取用户行为序列中项目之间的“依赖关系”。因此，我们提出了用于淘宝电子商务推荐的用户行为序列转换器（BST）。离线实验和在线A/B测试表明，与现有方法相比，BST具有优越性。BST已部署在排序阶段，用于淘宝推荐，每天为数亿消费者提供推荐服务。

二、架构

定义

在排序阶段，我们将推荐任务建模为点击率（CTR）预测问题，该问题可以定义为：给定用户 u 点击的用户行为序列 $S(u) = \{v_1, v_2, \ldots, v_n\}$，我们需要学习函数 $\mathcal{F}$ 来预测 u 点击目标项 v_t（即候选项）的概率。其他特征包括用户配置文件、上下文、项目和交叉特征。我们在 Wide&Deep 的基础上构建了BST，其总体架构如图1所示。 从图1中可以看出，它遵循了流行的Embedding+MLP范式，其中先前单击的项目和相关特征首先嵌入到低维向量中，然后再馈送到MLP。BST 和 Wide&Deep 之间的关键区别在于，我们添加了 Transformer 层，通过捕获底层序列信号来更好地表示用户单击的项目。在以下部分中，我们以自下而上的方式介绍了BST的关键组件：嵌入层、转换层和MLP。

图1：BST的总体架构。BST将用户的行为序列（包括目标项）和其他特征作为输入。它首先将这些输入特征嵌入为低维向量。为了更好地捕捉行为序列中项目之间的关系，Transformer 层用于学习序列中每个项目的更深层表示。然后，通过连接其他特征的嵌入和 Transformer 层的输出，使用三层 MLP 学习隐藏特征的交互，并使用 sigmoid 函数生成最终输出。注意，“位置特征”包含在“序列项特征”中。



嵌入层

第一个组件是嵌入层，它将所有输入特征嵌入到固定大小的低维向量中。在我们的场景中，有各种特征，如用户配置文件特征、项目特征、上下文特征以及不同特征的组合，即交叉特征。由于这项工作的重点是使用 Transformer 对行为序列进行建模，为了简单起见，我们将所有这些特征表示为“其他特征”，并在表1中给出了一些示例。如图1所示，我们将左侧的“其他特征”连接起来，并将其嵌入到低维向量中。对于这些特征，我们创建了嵌入矩阵 $\mathbf{W}_o \in \mathbb{R}^{|D| \times d_o}$ ，其中 d_o是维度大小。

此外，我们还获得了行为序列中每个项的嵌入，包括目标项。如图1所示，我们使用两种类型的特征来表示一个项目，“序列项目特征”（红色）和“位置特征”（深蓝色），其中“序列项目特征”包括项目id和类别id。请注意，一个项目往往有数百个特征，而选择全部来表示行为序列中的项目成本太高。

正如我们在之前的工作[14]中介绍的那样，项目id和类别id对于性能来说足够好，我们选择这两个作为稀疏特征来表示嵌入用户行为序列中的每个项。“位置特征”对应于以下“位置嵌入”。然后，对于每个项目，我们连接序列项目特征和位置特征，并创建嵌入矩阵 $\mathbf{W}_V \in \mathbb{R}^{|V| \times d_V}$ ，其中 d_V 是嵌入的维度大小，$|V|$ 是项数。我们使用 $\mathbf{e}_i \in \mathbb{R}^{d_V}$ 表示给定行为序列中第 i 项的嵌入。

位置嵌入：顺序存在于用户的行为序列中。因此，我们在将其投影为低维向量之前，将“位置”添加为底层每个项目的输入特征。注意，项目 v_i 的位置值计算为 $pos(v_i) = t(v_t) - t(v_i)$，其中 $t(v_t)$ 表示推荐时间，$t(v_i)$ 表示用户单击项目 v_i 时的时间戳。

嵌入层

表1：图1的左侧显示的“其他特征”。我们在实践中使用了更多特征，并为简单起见显示了许多有效的特征。

User	Item	Context	Cross
gender	category_id	match_type	age*item_id
age	shop_id	display position	os_item_id
city	tag	page No.	gender*category_id
…	…	…	…

Transformer 层

在这一部分中，我们介绍了Transformer层，它通过捕获行为序列中与其他项的关系来学习每个项的更深层表示。

自注意力层：缩放点积注意力定义如下：

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

其中 $\mathbf{Q}$ 表示查询，$\mathbf{K}$ 表示键，V表示值。在我们的场景中，自注意力操作将项目的嵌入作为输入，通过线性投影将其转换为三个矩阵，并将其反馈到注意力层。我们使用了多头注意力：

$$\mathbf{S} = \text{MH}(\mathbf{E}) = \text{Concat}\left(\text{head}_1, \text{head}_2, \cdots, \text{head}_h\right)\mathbf{W}^H$$

$$head_i = \text{Attention}\left(\mathbf{E}\mathbf{W}^Q, \mathbf{E}\mathbf{W}^K, \mathbf{E}\mathbf{W}^V\right)$$

其中投影矩阵 $\mathbf{W}^Q$，$\mathbf{W}^K$，$\mathbf{W}^V \in \mathbb{R}^{D\times D}$，而 $\mathbf{E}$ 是所有项目的嵌入矩阵，H 是头的数量。

Transformer 层

逐点前馈网络：在[12]之后，我们添加了点前馈网络（FFN），以进一步增强非线性模型，其定义如下：

$$\mathbf{F} = FFN(\mathbf{S})$$

为了避免过度拟合和分层学习有意义的特征，我们在自注意力和FFN中都使用了dropout和LeakyReLU。

然后，自注意力和FFN层的总体输出如下：

$$\begin{aligned}
\mathbf{S}' &= \mathrm{LayerNorm}(\mathbf{S} + \mathrm{Dropout}(MH(\mathbf{S})), \\
\mathbf{F} &= \mathrm{LayerNorm}\left(\mathbf{S}' + \mathrm{Dropout}\left(\mathrm{LeakyReLU}\left(\mathbf{S}'\mathbf{W}^{(1)} + b^{(1)}\right)\mathbf{W}^{(2)} + b^{(2)}\right)\right),
\end{aligned}$$

其中 $\mathbf{W}^{(1)}$，$b^{(1)}$，$\mathbf{W}^{(2)}$，$b^{(2)}$ 是可学习的参数，而层标准是标准归一化层。

Transformer 层

堆叠自注意力模块：在第一个自注意力块之后，它汇总了所有先前项目的嵌入，并进一步对项目序列的复杂关系进行建模，我们将自建块和第 b 个块定义为：

$$\mathbf{S}^b = SA\left(F^{(b-1)}\right)$$
$$\mathbf{F}^b = FFN\left(\mathbf{S}^b\right), \forall i \in 1, 2, \cdots, n.$$

在实践中，我们在实验中观察到 $b = 1$ 相比 $b = 2, 3$ 获得了更好的性能（请参见表4）。为了效率，我们没有尝试更大的 b，而是将其留给将来的工作。

MLP 层和损失函数

通过将其他特征的嵌入和适用于目标项目的 Transformer 层的输出连接，然后我们使用三个全连接层来进一步学习稠密特征之间的相互作用，这是工业 RSs 中的标准实践。

为了预测用户是否会单击目标项目 v_t，我们将其建模为二分类问题，因此我们将 Sigmoid 函数用作输出单元。为了训练模型，我们使用交叉熵损失函数：

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

其中 $\mathcal{D}$ 表示所有样本，而 $y \in \{0,1\}$ 是表示用户是否单击项目的标签，$p(x)$ 是 sigmoid 单元之后网络的输出，代表样本 x 被点击的预测概率。

三、实验

实验设置

数据集：该数据集是根据TAOBAO应用程序的日志构建的。我们根据用户的行为在八天内构建一个离线数据集。我们将前七天用作训练数据，最后一天用作测试数据。数据集的统计数据显示在表2中。我们可以看到数据集非常大且稀疏。

基线：为了显示 BST 的有效性，我们将其与两个模型进行了比较：Wide&Deep 和 DIN。此外，我们通过将顺序信息合并到 Wide&Deep 来创建基线方法，该信息平均汇总了先前单击项目的嵌入。

评估指标：对于离线结果，我们使用曲线下的区域（AUC）得分来评估不同模型的性能。对于在线A/B测试，我们使用CTR和平均RT来评估所有模型。 RT是响应时间（RT）的缩写，这是为给定查询生成推荐结果的时间成本，即TAAOBAO的用户的一个请求。我们使用平均RT作为指标来评估在线产品环境中不同的效率。

设置：我们的模型使用Python 2.7和TensorFlow 1.4实现，并且选择“ Adagrad”作为优化器。此外，我们给出表3中模型参数的详细信息。

实验分析

结果显示在表4中，从中，我们可以看到BST与基准相比的优越性。具体而言，AUC的离线实验从0.7734（Wide&Deep）和0.7866（DIN）提高到0.7894（BST）。在比较 Wide&Deep 和 WDL（+SEQ）时，我们可以看到以简单的平均方式合并顺序信息的有效性。这意味着借助自注意力，BST 提供了捕获用户行为序列的顺序信号的强大能力。请注意，从我们的实践经验中，即使是离线AUC的少量收益也可能导致在线CTR中的巨大收益。 Google的研究人员在WDL中报告了类似的现象[2]。

此外，就效率而言，BST 的平均 RT 接近 Wide&Deep 和 DIN 的RT，这可以保证在现实世界中部署像 Transformer 这样的复杂模型的可行性。

最后，我们还显示了在第2.2节中堆叠自注意力层的影响。从表4中，我们可以看到 $b = 1$ 获得了最佳的离线AUC。这可能是由于：用户行为序列中的顺序依赖性不如机器翻译任务中的句子那么复杂，因此较少数量的块足以获得良好的性能。因此，我们选择 $b = 1$ 将 BST 部署在生产环境中，并且仅报告表4中 $b = 1$的在线CTR增益。

实验分析

表4：离线AUC和在线CTR收益不同的方法。在线CTR增益相对于对照组。

Methods	Offline AUC	Online CTR Gain	Average RT(ms)
WDL	0.7734	-	13
WDL(+Seq)	0.7846	+3.03%	14
DIN	0.7866	+4.55%	16
BST(b=1)	**0.7894**	+7.57%	20
BST(b=2)	0.7885	-	-
BST(b=3)	0.7823	-	-

四、总结

总结

在本文中，我们介绍了如何将 Transformer 应用于 TAOBAO 推荐的技术细节。通过使用捕获顺序关系的强大能力，我们在建模用户行为序列中通过实验证明了 Transformer 的优越性。此外，我们还介绍了在淘宝的产品环境中部署模型的细节，该模型为中国数亿用户提供了推荐服务。

五、学习收获

学习收获

1. 把 Transformer 模型作为特征提取工具；
2. 关注序列特征。

Thank you