DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation

叶宇涛

2023.12

摘要

尽管许多基于矩阵分解的协同过滤方法取得了巨大成功, 但在推荐系统领域仍然有很大的改进空间。一个主要障碍是冷启动和数据稀疏问题, 需要更好的解决方案。近期的研究尝试将评论信息整合到评分预测中。然而, 存在两个主要问题: (1) 大多数现有工作使用静态和独立的方法提取用户和物品评论的潜在特征表示, 忽略了潜在特征之间的相关性, 可能无法全面捕捉用户的偏好。(2) 缺乏一个有效的框架将评分和评论统一起来。因此, 本文提出了一种新颖的面向物品推荐的评分和评论之间的双重注意力相互学习方法, 命名为 DAML。具体而言, 本文利用卷积神经网络的局部和相互注意力来共同学习评论的特征, 以增强所提出的 DAML 模型的可解释性。然后, 评分特征和评论特征被整合到一个统一的神经网络模型中, 并通过神经分解机实现特征的高阶非线性交互, 完成最终的评分预测。在五个真实世界数据集上的实验证明, 与最先进的方法相比,DAML 在评分预测准确性上取得了显著的改进。此外, 注意力机制可以突显评论中的相关信息, 增强评分预测的可解释性。本文增强了原论文的精度, 改善了部分功能。

关键词: 推荐系统; 神经网络; 注意力机制; 神经分解机; 评分预测

1 引言

本文是针对于文章:*DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation* [1] 的复现。

推荐系统已成为电子商务和在线服务的重要组成部分。它可以帮助用户从大量的物品中找到感兴趣的物品。在如今的推荐系统中, 推荐系统是一种通过分析用户的历史行为、兴趣和偏好, 向用户提供个性化建议的技术。这些建议通常是关于产品、服务、内容或信息的推荐。推荐系统在许多领域都有广泛的应用, 包括电子商务、社交媒体、在线视频和音乐流媒体等。

如今, 为了解决冷启动问题, 多模态推荐系统正在成为主流。多模态信息可以用来补充历史中用户-物品的交互记录, 并且可以补充不同模态之间的隐藏关系, 也可能捕获单个模态中缺乏的互补信息。像本文中的评分信息可能不足以完成高性能的推荐效果, 而加上评论这一模

态则可以有效提高推荐的性能。在这之中,DAML 是如今很多多模态推荐系统中文本类任务的对比模型。也正是因为该文足够经典, 因此本文选择复现该模型。

2019 年之前, 用户评分作为一种反映用户兴趣的用户反馈已被广泛用于预测用户的偏好。矩阵分解（MF）[2] 在学习用户偏好方面取得了巨大成功。依靠用户-物品交互记录,MF 方法将用户的偏好和物品的特征表示为隐空间向量, 用于评分预测。然而, 评分仅反映用户对物品的整体满意度, 并没有解释用户为什么给予物品这样的评分, 这导致了缺乏可解释性和冷启动问题 [3]。为了解决这些问题, 研究人员开始关注评论。伴随评分的评论通常包含与用户偏好和物品属性相关的各种信息。深度学习模型从评论中提取有效的潜在特征进行评分预测。

在这些工作中, 卷积神经网络（CNNs）被用来捕捉评论的更有效的上下文特征。尽管这些研究在推荐性能上取得了比基于主题的模型更好的效果, 但仍存在一些未深入研究的问题如下:

1. 大多数现有的研究以一种静态且独立的方式学习用户和物品的潜在特征 [4]。然而, 用户和物品的评论文本通常包含与用户和物品相关的语义信息, 而未考虑它们之间特征的相关性。这可能导致在预测用户偏好时产生较大偏差。

2. 有两种融合方法。一种是基于 MF 或因子分解机（FMs）的传统数据融合方法 [5]。然而, 这种方法无法捕捉不同模态之间特征的复杂性。另一种是一种直接的方法, 将从不同数据源提取的特征平等对待, 按顺序进行 concatenation, 成为用于推荐的特征向量。然而, 简单的向量连接并未考虑潜在特征之间的任何交互, 这在建模协同过滤模型中效果不佳 [6]。

为了解决上述问题, 该文提出了一种 DAML 模型。该模型利用 CNNs 的注意力机制和多层感知机的非线性特性, 实现对商品的预测评分。

- 该文所提出的 DAML 模型采用了双重注意力层: 一个局部注意力层和一个相互注意力层。前者在卷积层之前使用, 从局部窗口中选择有信息量的词, 有助于提取物品属性和用户偏好。后者在卷积层之后使用, 学习用户评论文本和物品评论文本之间的相关语义信息, 实现用户和物品的动态交互。

- 本文提出了一个统一的神经网络模型来融合评分和评论。然后使用神经分解机来实现潜在特征的高阶非线性交互。模型参数通过端到端的多任务学习方法进行训练。

- 实验在五个真实世界数据集上进行, 实验结果表明, 所提出的 DAML 模型在评分预测准确性方面优于现有 (2019 年) 的最先进方法。进一步的研究表明,DAML 在评论中突出显示的词汇非常有意义, 能够揭示用户对物品的具体偏好, 这有助于提高推荐系统的可解释性。

请在此部分对选题背景, 选题依据以及选题意义进行描述。

2 相关工作

2.1 推荐系统中的评论信息

传统的协同过滤推荐算法存在两个显著的缺点: 一是数据稀疏性, 二是冷启动问题。随着用户与系统平台之间互动的增加, 一些与用户和物品密切相关的辅助信息已经被利用来缓解

上述问题, 其中评论文本已成为提高推荐系统性能的研究热点。一些研究采用主题建模技术, 如 Latent Dirichlet Allocation（LDA）, 在评论上进行建模, 并将潜在主题与评分进行结合, 相较于仅利用评分或评论的 benchmark, 已经有显著的提升。然而, 这些研究忽略了评论中的词语顺序和局部上下文信息, 导致很多以短语和句子形式的具体信息丢失 [7]。与此同时, 这些研究采用了线性方法而非非线性方法 [8] 来结合评论和评分进行评分预测, 这并不足以捕捉特征交互的非线性关系和复杂结构。

2.2 推荐系统中的深度学习

深度学习技术在推荐系统领域取得了巨大成功。He 等人应用多层 MLP 来提取高层次的隐藏特征, 通过最大化用户-物品交互实现用户-物品特征的非线性交互 [9]。Wang 等人利用神经网络学习评论的深度特征表示, 使用概率矩阵分解（PMF）完成评分预测 [10]。然而, 这项工作仍然采用词袋 (bag-of-words) 表示法来学习潜在主题。为了提高深度特征表示的能力, 使用了词向量模型和卷积神经网络（CNNs）来学习用户行为和物品属性的表示 [7]。然而, 它们仍然以一种静态和独立的方式学习用户和物品的潜在特征, 未能学习潜在特征之间的相关性。

受关系预测和深度学习的研究 [11] 启发, 本文提出了一种用于物品推荐的 DAML 模型。提出的 DAML 模型在几个方面与上述模型显著不同。首先, 利用双重注意力机制来学习用户-物品交互。局部注意力层关注句子中不同单词的重要性, 而相互注意力层关注特征交互的学习。其次, 本文提出了一种新的方法, 不是应用注意力矩阵来推导特征之间的关系, 而是提出了定义相关性评分函数的新方法, 用于计算特征的相关性, 更能直观地捕捉用户和物品之间的关联。然后, 本文利用神经网络按照一些规定的规则将评分和评论统一起来进行评分预测。实验结果显示, 相较于最先进的方法,DAML 模型表现出更卓越的性能。

3 本文方法

3.1 问题定义

在本文中, 问题定义如下: u 表示一个用户,$U = \{u_1, u_2, ..., u_i, ..., u_M\}$ 表示整个用户集。同样,i 表示一个评论,$I = \{i_1, i_2, ..., i_j, ..., i_N\}$ 表示整个评论集。用户物品的评分矩阵 $R \in \mathbb{R}^{\boldsymbol{M}\times\boldsymbol{N}}$ 表示用户和物品之间的交互, 可以为评分或者是 0/1 样的交互反馈。在本文中,$r_{u,i} \in R$ 为真实的评分。

x 为一个用户的评论,$X = \{x_1, x_2, ..., x_i, ..., x_M\}$ 为整个用户集的评论集, 同样,s 为一个物品的评论,$S = \{s_1, s_2, ..., s_i, ..., s_N\}$ 整个物品集的评论集。

输入: 输入的交互信息是用户或者物品的身份标识, 也就是 ID。本文用独热稀疏编码 v_u^U 和 v_i^I 分别表示用户 $u \in U$ 和物品 $i \in I$。输入的评论信息为 $x_u \in X$, 为用户 u 的评论集,$s_i \in S$ 为物品 i 的评论集。**输出:** 整个训练过程可以被解释为以下函数 $f_{\boldsymbol{u}} : U, X, I, S \to \hat{R}$。模型的输出为最后的评分 $\hat{R}$, 也就是说, 对于任何的用户 u, 可以得出预测评分 $\hat{r_{u,i}}$ 基于函数 $f_u : \nu_u^U, \nu_i^I, \mathbf{x}_u, s_i \to \hat{r}_{u,i}$.

3.2 本文方法概述

本文中 DAML 模型的整体框架如图 1所示。

模型分为两个部分：一个是特征学习模块, 另一个是特征交互模块。

特征学习模块包含两个组件：基于评论的特征学习组件和基于交互的特征学习组件。基于评论的特征学习组件利用卷积神经网络（CNNs）的注意力机制学习用户偏好与物品特征之间的相关性。基于交互的特征学习组件用于将用户和物品映射成低维稠密向量, 以捕捉特征之间的非线性交互。

特征交互模块将评分和评论特征整合到一个统一的神经网络模型中。然后使用神经分解机 NFM 对潜在特征向量之间的高阶非线性交互进行建模。

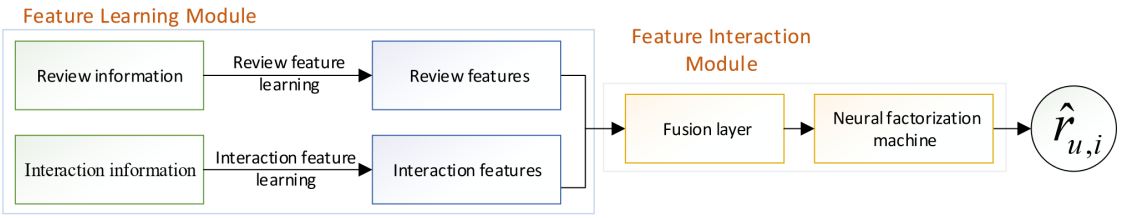

图 1. DAML 整体框架图

3.3 特征学习模块

基于评论的特征学习。基于评论的特征学习利用了卷积操作和卷积神经网络（CNNs）的注意力机制, 以学习用户和物品特征之间的相关语义信息。图2展示了基于评论的特征学习的框架。

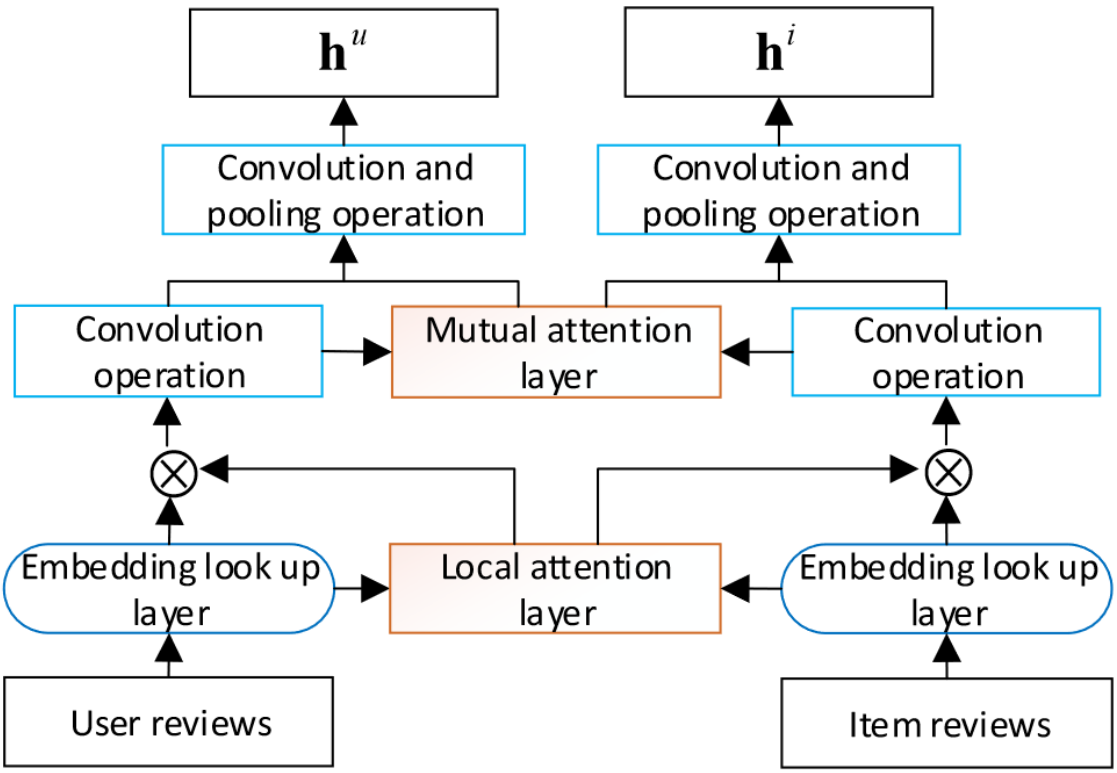

图 2. 双重注意力学习的机制目的在于学习用户和物品特征之间的相关性。符号 $\otimes$ 表示逐元素相乘,h_u 和 h_i 分别表示用户特征和物品特征。

嵌入查找层：给定包含 l 个单词并能表达完整含义的评论文本 $x_u \in X$。在嵌入层中, 使用词向量模型 Glove[15] 将 x_u 中的每个单词映射到词向量 w_i, 然后按照它们在评论文本中出现的顺序连接这些单词, 形成一个评论文本矩阵 $\mathbf{D} \in \mathbb{R}^{d \times l}$, 并且保留了单词的顺序。

$$\mathbf{D} = [\cdots, \mathbf{w}_{i-1}, \mathbf{w}_i, \mathbf{w}_{i+1}, \cdots] \tag{1}$$

在这里,d 是词向量的维度,w_i 表示在 x_u 中的第 i 个单词的词向量表示。

局部注意力层：受到工作 [18] 的启发, 本文利用一个注意力窗口来学习评论中每个单词的权重。将在词向量矩阵中 $\mathbf{D} \in \mathbb{R}^{d \times l}$ 的第 i 个单词作为中心词,ω 表示注意力窗口的大小。第 i 个单词的注意力权重 $s(i)$ 可以通过参数矩阵 W_{L_a} 和偏置向量 b_{L_a} 计算得到。

$$\begin{aligned} \mathbf{w}_{L-a,i} &= (\mathbf{w}_{i+\frac{-\omega+1}{2}}, \mathbf{w}_{i+\frac{-\omega+3}{2}}, \ldots, \mathbf{w}_i, \ldots, \mathbf{w}_{i+\frac{\omega-3}{2}}, \mathbf{w}_{i+\frac{\omega-1}{2}}) \\ s(i) &= \delta(\mathbf{w}_{L-a,i}\mathbf{W}_{L\_a} + \mathbf{b}_{L\_a}) \end{aligned} \tag{2}$$

在这里,δ 表示非线性激活函数, 本文使用 Sigmoid 函数作为激活函数。$s_(i)$ 是第 i 个词嵌入的注意力权重, 用以判断该词在句子中的重要性。根据注意力权重, 第 i 个词向量 $\widehat{\mathbf{w}}_i$ 可以计算如下：

$$\widehat{\mathbf{w}}_i = s(i)\mathbf{w}_i \tag{3}$$

在这里,$s_(i)$ 和 w_i 分别为注意力权重和第 i 个词的词向量表示。带有注意力权重的词向量矩阵可以表示为下式：

$$\hat{\mathbf{D}} = [\cdots, \hat{\mathbf{w}}_{i-1}, \hat{\mathbf{w}}_i, \hat{\mathbf{w}}_{i+1}, \cdots] \tag{4}$$

卷积操作：卷积操作可以有效地提取输入数据中的局部特征。通过卷积核在输入数据上滑动, 可以检测到不同位置的特定模式, 从而帮助网络学习数据的局部表示。在这里, 给出了词向量矩阵 $\hat{D}$, 卷积操作用来提取矩阵中的语义信息。第 $j-th$ 个卷积核且滑动窗口大小为 ω 可以提取出矩阵 $\hat{D}$ 中的局部特征 c_i^j。可以表示为：

$$c_i^j = \mathbf{W}_c^j * \hat{D}_{(;,i:(i+\omega-1))} \tag{5}$$

在这里,* 表示卷积操作,$\mathbf{W}_c^j$ 表示第 $j-th$ 个卷积核,$\hat{D}_{(;,i:(i+\omega-1))}$ 表示矩阵 $\hat{D}$ 中第 i 到第 $i+\omega-1$ 列的子矩阵。为了产生 l 个单词, 在卷积操作前, 将 $\omega-1$ 维零向量填充到矩阵 $\hat{D}$ 的尾部。由于在卷积核中共享权重的关系, 一种卷积只能捕捉一种类型的上下文关系。在这里, 本文用了多种不同权重的卷积核对每个单词捕获上下文信息, 在卷积操作后, 第 $i-th$ 位置的单词可以被表示为 c_i:

$$\mathbf{c}_i = [c_i^1, c_i^2, \cdots, c_i^i, \cdots, c_i^f] \tag{6}$$

在这里,c_i^f 是由第 f 个卷积核产生的上下文特征。因此, 用户评论和物品评论可以表示为下式：

$$\begin{aligned} \mathbf{U} &= [\mathbf{c}_1^u, \mathbf{c}_2^u, \mathbf{c}_3^u, \cdots, \mathbf{c}_{l_u}^u] \\ \mathbf{V} &= [\mathbf{c}_1^i, \mathbf{c}_2^i, \mathbf{c}_3^i, \cdots, \mathbf{c}_{l_i}^i] \end{aligned} \tag{7}$$

在这里,c_k^u 和 c_j^i 分别为用户中第 $k-th$ 个单词的和物品中第 $j-th$ 个单词的上下文特征向量,l_u 和 l_i 分别为用户和物品评论的长度。

相互注意力层：类似于文献 [23, 26], 我们利用一个相互注意力层来研究用户评论和物品评论之间的关联。具体而言, 我们定义了一个关联评分函数 $f_{relation-score}$, 它利用欧几里得距离计算用户上下文特征 U 与物品上下文特征 V 之间的相关性,$f_{relation-score}$ 可以计算如下：

$$f_{relation-score} = 1/(1 + |\mathbf{c}_k^u - \mathbf{c}_j^i|) \tag{8}$$

依据公式 (8) 可以计算 $\mathbf{A} \in \mathbb{R}^{\boldsymbol{M}\times\boldsymbol{N}}$：

$$\mathbf{A} = f_{relation-score}(\mathbf{U}, \mathbf{V}) \tag{9}$$

在 A 矩阵中的每一个元素表示用户-物品之间的相关性。在该矩阵中的每一行 $A_{k,*}$ 表示 U 中每个上下文特征 c_k^u 与 V 中上下文特征的相关性。同样对于 $A_{*,j}$ 也是一样的。上下文特征 c_k^u 和 c_j^i 的相关性权重 g_k^u 和 g_j^i 可以表示为下式：

$$\begin{aligned} g_k^u &= \sum \mathrm{A}[k,:] \\ g_j^i &= \sum \mathrm{A}[:,j] \end{aligned} \tag{10}$$

局部池化层：受到工作 [28] 的启发, 本文引入了注意力权重作为一种替代方案, 但作为 baseline 还是采用了平均池化法, 具体如下。对于相互关注层的输出特征图, 我们对连续行进行逐行平均, 以生成更高粒度、更抽象的特征。因此可以得到带有相互注意力权重的上下文特征：

$$\begin{aligned} \mathbf{t}_k^u &= \sum_{k=k:k+\omega} g_k^u \cdot \mathbf{c}_k^u \\ \mathbf{t}_j^i &= \sum_{j=j:j+\omega} g_j^i \cdot \mathbf{c}_j^i \end{aligned} \tag{11}$$

在这里,$\mathbf{t}_k^u \in \mathbf{U}^u = [\mathbf{t}_1^u, \mathbf{t}_2^u, \mathbf{t}_3^u, \cdots, \mathbf{t}_{l_u}^u]$ 和 $\mathbf{t}_j^i \in \mathbf{V}^i = [\mathbf{t}_1^i, \mathbf{t}_2^i, \mathbf{t}_3^i, \cdots, \mathbf{t}_{l_i}^i]$ 表示用户评论和物品评论的第 k 个位置和第 j 个位置的上下文特征。为了提取更抽象的特征并减少无关方面的噪音, 我们在局部池化层上叠加了一个卷积层和一个平均池化层。抽象特征的计算如下：

$$\begin{aligned} h_h^j &= \delta(\mathbf{W}_a^j \mathbf{U}_{h:h+\omega-1}^u + b_a^j) \\ \boldsymbol{h_h} &= mean(h_1^j, \cdots, h_{l_u-\omega+1}^j) \\ \mathbf{h}^u &= [h_1, \cdots, h_f] \end{aligned} \tag{12}$$

其中,$\mathbf{W}_{\boldsymbol{a}}^{\boldsymbol{j}}$ 表示第 j 个卷积滤波器的卷积权重, 滑动窗口的大小为 ω,b_a^j 是偏置项, 第 j 个卷积滤波器在位置 h 的特征值为 h_h^j。最终, 得到了具有用户-物品相关性的上下文特征 h^u 类似地, 我们可以得到物品上下文特征 h^i。最后, 我们使用非线性函数将上下文特征值映射到一个维度空间, 以完成推荐任务。

$$\begin{aligned} \mathbf{h^u} &= \delta(\mathbf{W^u h^u} + b^u) \\ \mathbf{h^i} &= \delta(\mathbf{W^i h^i} + b^i) \end{aligned} \tag{13}$$

在这里,W^u 和 W^i 分别表示用户和物品的上下文特征的权重矩阵。b^u 和 b^i 为偏置项。δ 表示非线性激活函数。

基于评分的特征学习：将用独热编码表示的物品和用户的身份标识表示为 v_i^I 和 v_u^U。通过 embbeding 层中的潜在因子矩阵 $\mathbf{P}_u \in \mathbb{R}^{M\times K}$ 和 $\mathbf{Q}_i \in \mathbb{R}^{N\times K}$ 将 v_i^I 和 v_u^U 转化为低维稠密向量。可以表示为下式：

$$\begin{aligned}\mathbf{p}_u &= \mathbf{P}_u^T \nu_u^U \\ \mathbf{q}_i &= \mathbf{Q}_i^T \nu_i^I\end{aligned} \tag{14}$$

3.4 特征交互模块

在这里,p_u 和 q_i 分别表示用户和物品的交互特征。为了全面捕捉用户偏好和物品特征, 首先在特征交互模块中融合这两种模态信息。融合后的用户和物品特征可以表示为：

$$\begin{aligned}\mathbf{u}^u &= \mathbf{h}^u + \mathbf{q}_u \\ \mathbf{v}^i &= \mathbf{h}^i + \mathbf{p}_u\end{aligned} \tag{15}$$

其中 u^u 和 v^i 分别表示最终的用户和物品特征。通过 concat 来连接用户和物品特征。

$$\mathbf{z} = \delta\left[\mathbf{u}, \mathbf{v}\right] \tag{16}$$

在这里,z 表示用户-物品特征,δ 为激活函数,[,] 表示在隐层进行 cancat 操作。受到工作 [7] 的启发, 使用 NFM 来捕捉高阶非线性相互作用特征,。目标函数可以表示为：

$$\hat{r}_{u,i}(\mathrm{z}) = m_0 + \sum_{j=1}^{|\mathrm{z}|} m_j z_j + f(\mathrm{z}) \tag{17}$$

在这里,$\mathbf{z} \in \mathbb{R}^{|\mathbf{z}|}$ 表示用户-物品特征向量,$z_j \in \mathbf{z}$ 为特征 j 的值,m_0 和 m_j 分别为全局偏置项和权重项。$f(\mathbf{z})$ 表示高阶非线性交互特征, 可以表示为：

$$f(\mathbf{z}) = \frac{1}{2}[(\sum_{j=1}^{|\mathbf{z}|} z_j \mathbf{v}_j)^2 - (\sum_{k}^{|\mathbf{z}|} z_k \mathbf{v}_k)^2] \tag{18}$$

其中,$z_j, z_k \in \mathbf{z}$ 表示第 j 个位置和第 k 个位置的用户-物品特征,$\mathbf{V}_j, \mathbf{V}_k \in \mathbb{R}^s$ 表示特征 j 和特征 k 的嵌入向量, 与 [27, 29] 类似, 通过堆叠多层全连接层来捕获特征之间的高阶交互, 最终的预测目标函数可以表示为：

$$\hat{r}_{u,i}(\mathrm{z}) = m_0 + \sum_{j=1}^{|\mathrm{z}|} m_j z_j + \mathrm{h}^{\mathrm{T}} \delta_{\mathrm{L}}(\mathrm{W}_{\mathrm{L}}(\cdots \delta_1(\mathrm{W}_1 f(\mathrm{z}) + \mathrm{b}_1)\cdots) + \mathrm{b}_{\mathrm{L}}) \tag{19}$$

其中 $\hat{r}_{u,i}(\mathrm{z})$ 为最后的预测评分。模型的参数 $\Theta = \{m_0, \{m_j, \mathbf{v}_j\}, \mathbf{h}, \{\mathbf{W_L}, \mathbf{b_L}\}\}$,$\delta_{\mathrm{L}}$ 为激活函数。与 FM 相比, 增加的参数项 $\{\mathrm{W}_{\mathrm{L}}, \mathrm{b}_{\mathrm{L}}\}$ 是为了捕获高阶非线性交互特征。

3.5 训练细节

为了学习 DAML 模型的参数, 用均方误差作为目标函数:

$$J = \sum_{(u,i)\in R} (\hat{r}_{u,i} - r_{u,i})^2 + \lambda_\Theta ||\Theta||^2 \tag{20}$$

其中,R 表示用户-物品的评分矩阵,$r_{u,i}$ 为用户 u 对物品 i 的真实评分,$\hat{u_{u,i}}$ 为预测的评分,Θ 为所有参数,$\lambda_\Theta ||\Theta||^2$ 为 L2 正则化参数, 防止过拟合。DAML 的参数基于公式 (20) 通过随机梯度下降 (SGD) 和反向传播进行优化。也就是说, 两个学习模块的参数是共同学习的。对于参数更新, 在小批量上使用自适应矩估计（Adam）。此外, 为了防止过度拟合, 对神经分解机采用了 dropout 策略。

4 复现细节

4.1 与已有开源代码对比

原论文并未给出源代码, 在查找 github 上面的资料后, 发现存在其他作者复现的代码: https://github.com/TianHongTao/ID-DAML。但该代码存在一些缺陷, 如没有给出数据集的下载地址, 没有给出模型的测试代码, 没有给出模型的评价指标,NLP 中也没有应用 Glove 进行转换等。因此, 本文在该代码的基础上进行了改进, 将 word2vec 转换成 glove, 给出了数据集的下载地址, 给出了模型的训练和测试代码, 给出了模型的评价指标等。并且, 在 predict 模块中, 作者并未使用 NFM, 而是利用了 FM 进行代替, 这一定程度上影响了模型的性能。同时, 作者在优化器中选择的是 Adam, 而非本论文使用的 SGD。参考了该作者的代码之后, 本文的代码做出的改进如下:

1. 整合了 DAML 的模型, 形成了框架, 该框架包括数据预处理、模型训练、模型测试等模块, 方便使用, 提高了代码的可读性, 优化了代码的结构, 更加解耦化。
2. 数据预处理提供一键处理, 已经整理好数据集的 config 文件。
3. 将整个模型训练流程进行了分块, 分为模型初始化、评论融合、模型预测三个模块。
4. 在评论融合中, 提供了多种融合方式, 比如 concat、add、multiply、self-Attention 等。
5. 在模型预测中, 提供了多种预测方法, 比如 FM、NFM、LFM、MLP 等。
6. 在模型初始化中, 提供了多种优化器供选择, 比如说 ADAM、SGD 等。
7. 所有参数都初始化为可调参数, 方便使用者进行调参。

本文的代码总框架如图3所示。

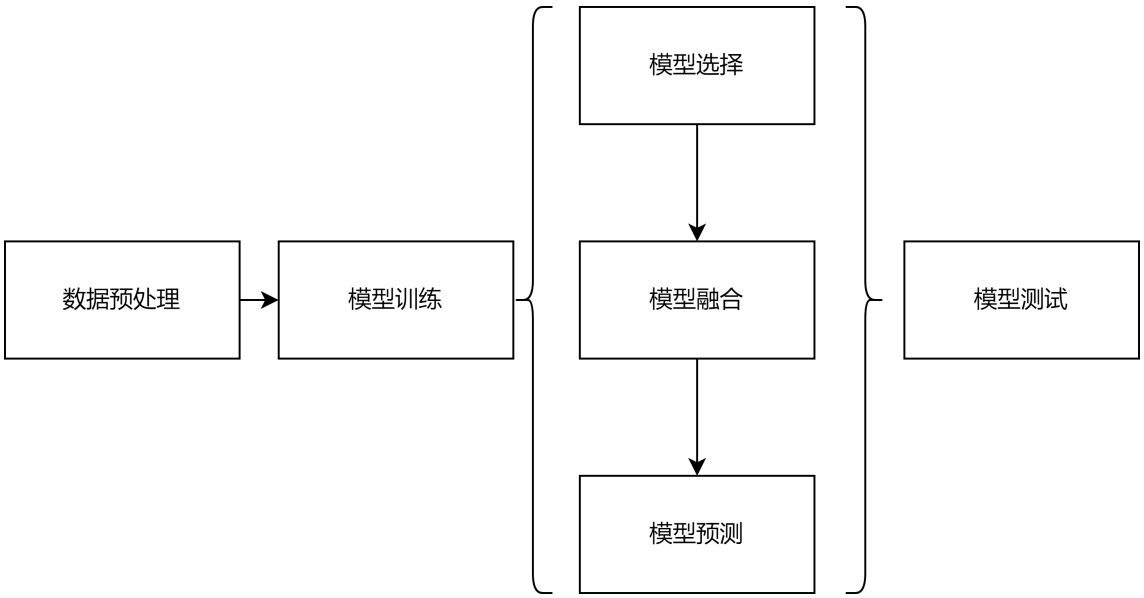

图 3. 代码总框架图

4.1.1 引用代码说明

reappearance/user_latent 文件,引用了 LocalAttention、MutualAttention、forward、DAML 类，本文模型写法基本和其一致。

Utils/AWS.py 文件，引用了 numerize 函数，将其用在数据预处理处。

Utils/gen_vec.py 文件, 引用了 main、clean_str 函数, 主要用在数据预处理处, 清洗文本。https://github.com/vanzytay/KDD2018_MPCN/blob/master/tylib/lib/compose_op.py#L13中，引用了其中的 build_fm(), 用于预测评分模块中的 FM 功能。

4.1.2 数据预处理

在数据预处理中, 本文提供一键处理的功能, 并且将处理完后的数据集信息输入到 config 文件中, 方便使用者进行查看。数据预处理的流程如下：

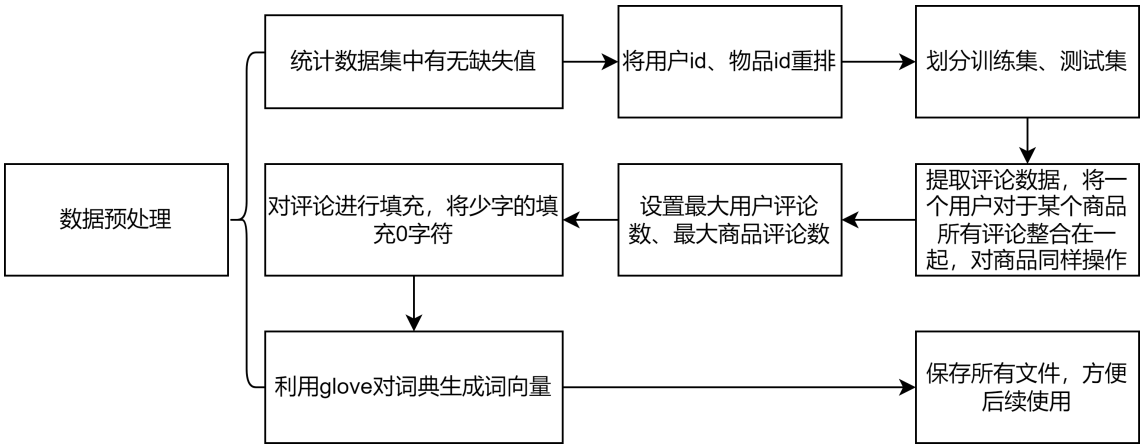

图 4. 数据预处理流程图

这里用一个 doc 表示商品的全部评论, 是为了方便以后模型的操作, 减少数据的维度。这也是因为多个用户对同一商品的评论是没有顺序的, 将评论进行拼接不影响模型性能。同时,

利用 Glove 生成词向量而非 word2vec, 因为 Glove 可以更好的处理 OOV 问题。这里选择的时候 Glove.27B.100d, 是因为该词向量是在互联网上训练得到的, 可以更好的处理互联网上的评论数据。同时, 该词向量的维度为 100, 可以更好的表达评论的语义信息。

4.1.3 模型选择

本文中的模型选择模块, 主要是对模型的初始化, 这里主要是 DAML 模型的初始化, 包括了 user_id 的 embedding,user_review 的 embedding 局部注意力层、相互注意力层、融合层、池化层。同时还提供了多种优化器进行选择, 比方说 ADAM、SGD 等优化器。同时, 还提供了多种损失函数进行选择, 比方说 MSE、MAE 等损失函数。同时, 还提供了多种评价指标进行选择, 比方说 RMSE、MAE 等评价指标。

4.1.4 模型融合

本文中的模型融合模块, 提供了模型的多种融合方法。本文的特征融合, 主要是 concat 操作, 也就是将两个特征向量直接进行拼接。本文还增加了 self-Attention 机制, 在后文中会进行详细介绍, 主要是利用 transformer 的 encoder 模块进行注意力方面上的融合。同时, 还增加了 add 和 multiply 两种融合方式,add 是将两个特征向量进行相加,multiply 是将两个特征向量进行相乘。

4.1.5 模型预测

模型预测模块中, 源代码是利用 FM 进行预测, 不符合该论文原文的要求。因此这里增加了 NFM 预测模块, 同时还增加了 LFM、MLP 等预测模块。

4.2 实验环境搭建

本文实验条件如表1所示。

软件	版本	硬件	型号
pytorch	1.10.2	显卡	RTX 4090 24G
cuda	11.8	内存	512G
python	3.6.13	CPU	Intel(R) Xeon(R) Platinum 8358P CPU @ 2.60GHz
pandas	1.1.5	系统版本	Ubuntu 9.4.0

表 1. 实验环境配置

4.3 界面分析与使用说明

由于本实验在服务器上进行, 因此没有界面, 只有命令行界面。使用说明如下。

main.py 提供该项目的主入口，kwargs 填写模型的各项需要进行设置的参数。主要可以设置如下参数:

参数	说明	参数	说明
filters_num	卷积核数量	dataset	数据集
output	评分预测方式	model	DAML
num_epochs	训练 epoch	num_fea	隐层特征维度
self_att	是否使用自注意力机制	batch_size	batch 大小
pth_path	存储路径	gpu_id	使用哪个 GPU
optimizer	优化器	lr	学习率

表 2. kwargs 填写参数设置

设置完毕后，使用 train(kwargs) 函数，即可训练模型。模型将会自动保存到 dataset 文件夹中。使用 test(kwargs) 函数，可以进行模型的测试，模型测试的结果也将会自动保存到 dataset 文件夹中。如果需要添加模型，需要在 model 文件夹中编写模型。编写的模型需要在 forward() 中返回经过模型处理后的特征向量。

4.4 创新点

4.4.1 在融合阶段加入 transformer 块

多模态任务的困难点在于模态之间的对齐和交互。而 transformer [12] 块利用 self-Attention 机制, 在模态转换为特征向量之后, 通过 concat 特征向量, 可以自然地捕捉模态之间的关系, 进行融合。在最后的融合阶段, 根据公式 (16), 用户和物品特征是用 concat 进行连接的。而在公式 (15) 中, 为了加入评分信息, 直接相加融合用户特征和用户评分。我们猜测, 这样直接对特征向量相加再拼接, 可能会导致用户和物品之间的信息丢失, 因此, 在公式 (16) 中, 不再直接简单的拼接, 而是利用 transformer Encoder 块, 增加特征向量之间的交互, 使得用户和物品之间的信息更加充分, 从而提高模型的性能。因此, 公式 (16) 可以修改为:

$$\begin{aligned} \mathbf{u}_{out}, i_{out} &= transformer\left[\mathbf{u}, \mathbf{v}\right] \\ z &= \left[\mathbf{u}_{out}, \mathbf{i}_{out}\right] \end{aligned} \tag{21}$$

公式 (21) 中,transformer 块的输入为用户和物品的特征向量, 输出为用户和物品特征向量的隐层向量, 然后再进行拼接, 得到最终的用户-物品特征向量。transformer 块的结构如图5所示。

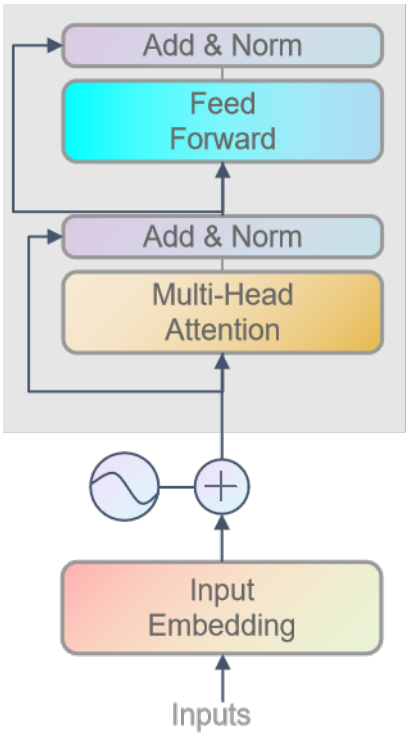

图 5. transformer Encoder 块结构图

而 transformer 中存在多头注意力机制, 类似于多个卷积核可以提取不同的特征, 多头注意力机制可以提取不同的特征, 因此, 这里采用了多头注意力来增强特征之间的交互。

4.4.2 卷积核调整

数量过大的卷积核虽然能提取足够的特征, 但对于性能可能存在比较大的影响, 在之后的实验中会证明这个结论。在 NLP 领域中, 常用的一维卷积形式如所示：在本文的代码实现中, 采用了 100 个卷积核, 将 100 维的 embedding 层进行特征提取, 最后得出来的特征向量拼接后还是 100 维。而这种做法十分耗费时间, 并且,100 个相同大小的卷积核即使权重不同, 对于提取特征来说, 可能会存在一定的冗余。因此, 本文将卷积核的数量减少, 并且提高一个 batch 中的样本数量, 这样可以减少训练时间, 同时, 也可以提高模型的性能。在之后的实验中, 会对这一点进行详细的实验说明。

4.4.3 优化器

在原论文中使用的优化器是 SGD 优化器, 众所周知, 不同的优化算法对模型训练的结果有很大的影响。但是, 原论文中没有给出 SGD 的具体参数, 这就导致我们如果想对模型进行进一步提升, 需要对 SGD 中的参数进行精细的调整。在本论文中的五个数据集中, 工作量无疑是巨大的。而 Adam 在 SGD 的基础上, 进行了进一步的优化：

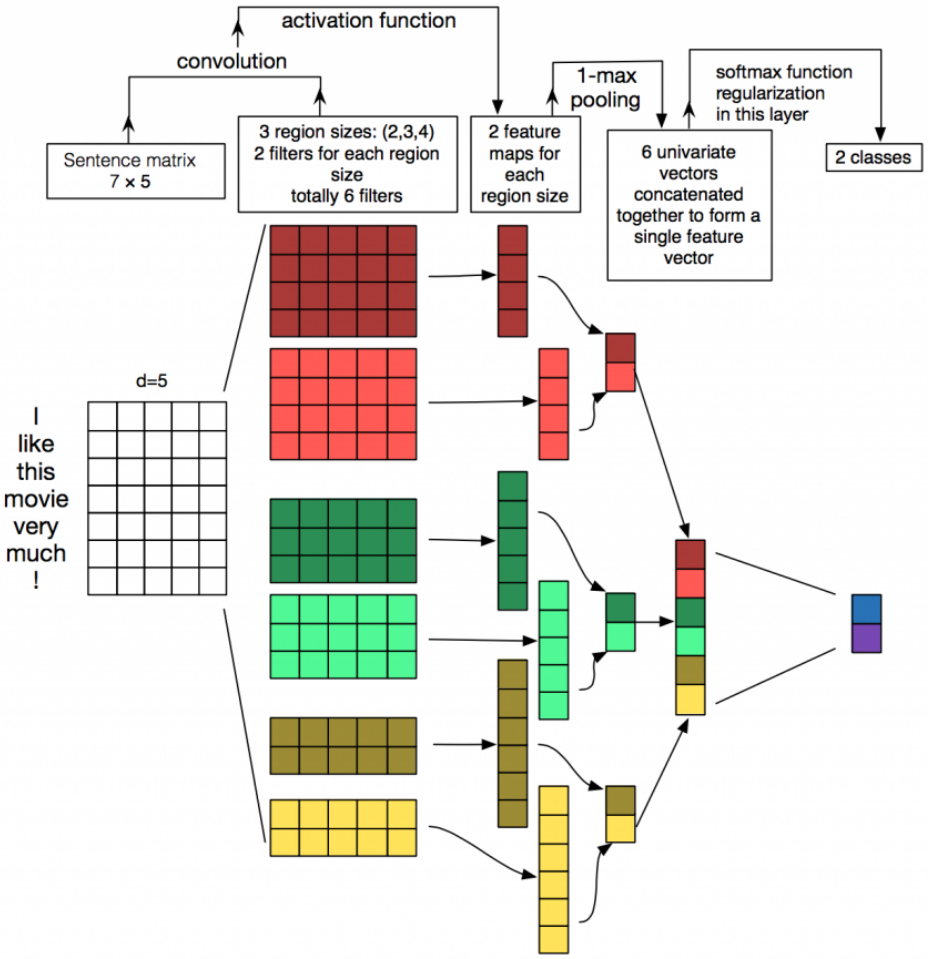

图 6. 卷积核形式

$$
\begin{aligned}
\mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\nabla_\theta J(\theta_{t-1}) \\
\mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\left(\nabla_\theta J(\theta_{t-1})\right)^2 \\
\hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1-\beta_1^t} \\
\hat{\mathbf{v}}_t &= \frac{\mathbf{v}_t}{1-\beta_2^t} \\
\theta_t &= \theta_{t-1} - \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}+\epsilon}
\end{aligned}
\tag{22}
$$

其中,β_1 和 β_2 分别为一阶矩估计和二阶矩估计的指数衰减率,ϵ 为防止分母为 0 的项。Adam 算法的优点在于, 可以自动调节学习率, 而不需要人为的进行调节。因此, 本文在实验中, 将优化器从 SGD 改为 Adam, 以期望提高模型的性能。在后续实验中, 可以证明这一点。

4.4.4 MLP

在前面的一系列操作中, 结合了局部注意力、全局注意力、transformer 层三个注意力层, 都是为了捕捉特征之间的关系。最后的 NFM 操作, 是为了获得特征之间的高阶关系。这里猜想, 如果注意力操作过多, 会导致特征过于平滑, 失去了特征之间的差异性, 可能导致一些特征失去原本的作用。因此, 本文在模型预测模块中, 如果取消了 NFM 操作, 而是采用了 MLP 操

作, 可能可以提高模型的性能。MLP 操作如下:

$$\begin{aligned} \mathbf{h}_1 &= \delta\left(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1\right) \\ \mathbf{h}_2 &= \delta\left(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2\right) \\ \hat{r}_{u,i}(\mathbf{z}) &= \mathbf{h}_2 \end{aligned} \tag{23}$$

其中,$\mathbf{W}_1$ 和 $\mathbf{W}_2$ 分别为第一层和第二层的权重矩阵,$\mathbf{b}_1$ 和 $\mathbf{b}_2$ 分别为第一层和第二层的偏置项,δ 为激活函数。

5 实验结果分析

实验主要采用的是推荐系统中的经典数据集:Amazon 5-core, 利用五个公开的数据集, 这五个数据集来自 Amazon 5-core [13], 其中包括有关乐器、办公产品、杂货和美食、视频游戏、运动和户外的评论。由于原始数据非常大且稀疏, 我们对其进行了预处理, 以确保所有用户和项目至少有一个评分。为了减轻评论的长尾效应, 我们按照 [4] 中使用的预处理步骤来调整评论的长度。这些数据集的特征如表3所示。

Dataset	#Users	#Items	#Ratings	Word per user	Word per item	density
Musical Instruments	1,429	900	10,261	170	163	0.798%
Grocery and Gourmet Food	4,905	2,420	53,228	203	172	0.449%
Office Products	14,681	8,713	151,254	177	155	0.118%
Video Games	24,303	10,672	231,577	148	135	0.089%
Sports and Outdoors	35,598	18,357	296,337	159	154	0.045%

表 3. 数据集信息，包含了五个大小不一的数据集，最大的是 Sports and Outdoors ，最小的是 Musical Instruments 数据集。

本文做了 17 个实验, 主要围绕以下几个方面:

1. 不同的融合方式对模型的影响;
2. 不同的超参数比较;
3. 不同的优化器对模型的影响;
4. 不同的卷积核对模型的影响;
5. 不同的评价指标对模型的影响;
6. 不同的模型预测方式对模型的影响;
7. transformer 块对模型的影响。

表4展示的是原论文中的结果与本文复现结果的对比：

Method	Musical Instruments	Office Products	Grocery and Gourmet Food	Video Games	Sports and Outdoors
PMF	1.137	1.265	1.397	1.395	1.203
NeuMF	0.7198	0.7301	0.9434	0.8693	0.7516
CDL	0.8336	1.062	0.9669	0.9018	0.8522
ConvMF	0.7860	0.7279	0.8634	0.8993	0.8235
DeepCoNN	0.7590	0.7109	0.8016	0.8752	0.7192
D-attn	0.7420	0.7161	0.8241	0.8422	0.7840
NARRE	0.6949	0.6807	0.7467	0.7991	0.6897
CARL	0.6766	0.6469	0.7534	0.7979	0.6864
DAML	**0.6510**	**0.6124**	**0.7354**	**0.7881**	**0.6676**
MY-DAML	0.66	0.6684	0.7405	0.792	0.7399
Δ%	1.382%	9.144%	0.694%	0.495%	10.830%

表 4. 不同模型在五个数据集上的 MAE 对比, 其中 DAML 为原论文中得出的结果,MY-DAML 为本文复现的实验结果, 参数基本一致。**加粗字体部分**为最好的 MAE 结果,下划线部分为次优结果。Δ 为复现的结果与原论文结果相差百分比。

将原论文中的结果与本文复现的结果进行对比, 可以发现, 在五个数据集上, 本文复现的结果与原论文的结果相差不大, 这一点可以从表4中的 Δ 百分比中看出。这一点说明, 本文复现的结果是可信的, 后续各项参数的调整得出的实验结果, 也应当具备一定可信度。但是, 距离原论文差别比较大的两个数据集为 Office Products 和 Sports and Outdoors, 这两个数据集的 MAE 结果分别比原论文的结果低了 9.144% 和 10.830%, 这里差异的原因, 可能在于大数据集的训练时间不足够, 在原论文参数的基础上进行训练, 会导致收敛不足。

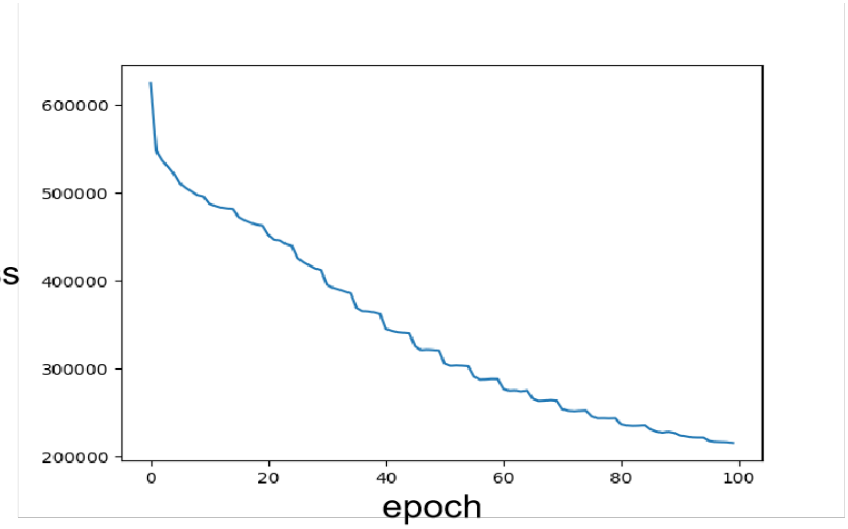

图 7. Sports and Outdoors 训练曲线

如图7, 从 loss 曲线可以看出, 在学习率较大时, 曲线收敛速度够快, 在后续学习率衰减时, 收敛速度减少, 在最后几个 epoch 时,loss 并不是很稳定, 这也可能是导致模型性能下降的原因

之一。因此, 在后续的实验参数调整中, 会针对 epoch 和学习率进行调整, 保证模型不会欠拟合。

在这里，给出所有实验的比较结果，如表6所示。该表展示了本文中所有尝试的实验的实验结果。本文只采用了最小的数据集 Musical Instruments 进行实验，这是因为该模型存在一定规模，进行大数据集的训练时间消耗可能过长，不利于实验的进行。从实验结果对比可以看出，实验 3 获得了最佳的实验结果，实验 13 获得了次佳的实验结果，这里给出各个试验的参数的不同设置，如表5所示。在下节中，将详细解释各个实验结果，以及实验参数设置的原因。

实验	实验参数设置
EXP1	EMBEDDING_SIZE=32
EXP2	损失改 MAE，ID_EMD_SIZE=8
EXP3	**MAE,SELF_ATT=TRUE，2 头注意力**
EXP4	MSE 加上注意力
EXP5	SGD+MSE
EXP6	LR=1E-4
EXP7	注意力 +MAE+LR=1E-4
EXP8	NFM+25epoch
EXP9	batch_size=128,epoch=30
EXP10	batch_size=256,epoch=30
EXP11	batch_size=128,epoch=30,self_att
EXP12	batch_size=128,epoch=31,self_att,lr=1e-4
EXP14	batch_size=128,epoch=100,self_att,lr=1e-4, 交换 user,item
EXP15	batch_size=128,epoch=50,lr=1e-4, 交换 user,item，filter_num=20
EXP16	跟原文几乎一样,filter_num=10,epoch=30,lr=1e-3
EXP17	原文数据,batch_size=128.eppch=70,lr=1e-5,SGD

表 5. 实验参数设置

实验	MAE	DAML %	MY-DAML %
EXP1	0.6533	-0.353%	1.015%
EXP2	0.6546	-0.553%	0.818%
EXP3	**0.5852**	**10.108%**	**11.333%**
EXP4	0.6505	0.077%	1.439%
EXP5	1.075	-65.131%	-62.879%
EXP6	0.615	5.530%	6.818%
EXP7	None	None	None
EXP8	0.6478	0.492%	1.848%
EXP9	0.6232	4.270%	5.576%
EXP10	0.6207	4.654%	5.955%
EXP11	0.631	3.072%	4.394%
EXP12	0.6159	5.392%	6.682%
EXP12-second	0.6164	5.315%	6.606%
EXP13	0.6018	7.558%	8.818%
EXP14	0.6169	5.238%	6.530%
EXP15	0.6857	-5.330%	-3.894%
EXP16	0.611	6.144%	7.424%
EXP17	0.7837	-20.384%	-18.742%

表 6. 实验结果对比, 其中 DAML 为原论文中得出的结果对比,MY-DAML 为本文复现的实验结果对比，其中，实验 7 由于训练时服务器终止，导致结果丢失。实验 12 进行了两次实验，确保结果的正确。黑体表示该实验为最佳结果，下划线表现该实验为次佳结果。

5.1 不同的融合方式对模型的影响

实验 11 是为了验证 transformer 块对模型的影响，因此，实验参数设置基本和原文一致，只是在特征融合阶段，添加了 transformer 块进行特征之间的融合。相比于原文的结果，本文取得了 3.072% 的结果的改进，相比于复现的结果，取得了 4.394% 的改进。该结果的改进是显而易见的。原文中添加物品的 id_embedding 和用户的 id_embedding 到 user_embedding 和 item_embedding 是直接进行添加。这么操作的结果很容易将用户和物品的特征进行平滑化，缺失差异性。

添加的 transformer 模块主要是利用其中的 self-Attention 机制,将原本添加的 id_embedding 添加权重，防止过分干扰原本的特征向量，从而提高模型的性能。同时，transformer 模块中的多头注意力机制，允许模型在不同的子空间中学习不同的特征。每个注意力头都可以关注输入序列中的不同部分，从而捕捉更丰富的语义信息。通过组合多个头的信息，模型能够更好地理解输入数据的复杂关系。并且可以提高模型的鲁棒性，即使某些头在某些情况下学到

了噪声或错误的信息，其他头仍然可以提供有价值的信息。这有助于减少模型对特定注意力权重的依赖，使其更加鲁棒。

5.2 卷积核数量

根据公式 6 所示，原文将特征向量进行了卷积操作，形成了 f 维的特征向量。按照原文的参数设置，应该为 100 维的特征向量经过卷积层后，还是为 100 维的向量，这里进行了简单的实验，确认了卷积核的数量大小对实验运行时间的影响，如图8所示。

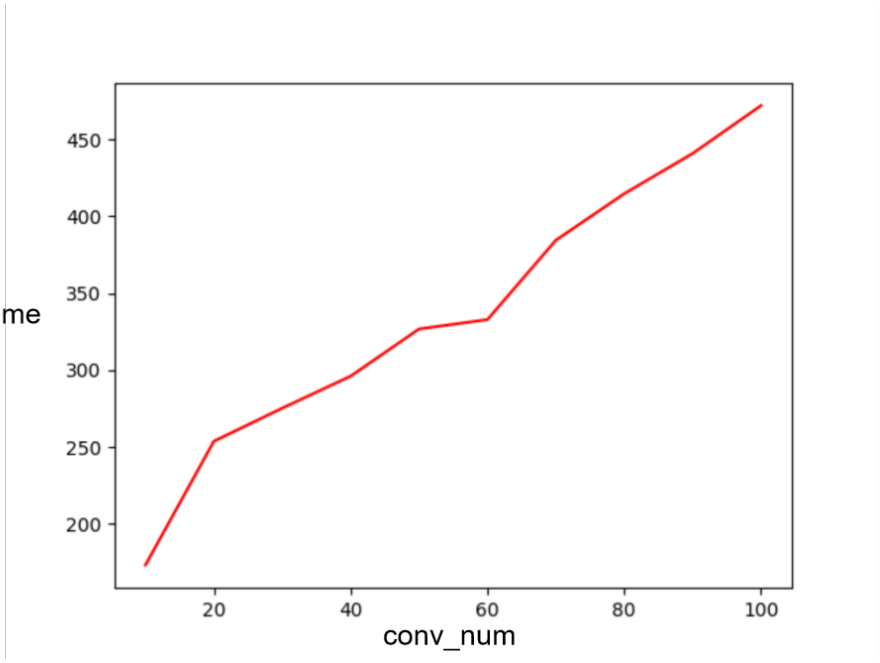

图 8. 卷积核数量对实验运行时间的影响

实验 9 的实验是针对卷积核的大小进行的实验，将卷积核的大小缩小之后，再调整 batch_size 的大小，基本可以保持原来的结果不变，而将训练时间减小一半以上。并且，针对于原来的参数，具有一定的精度提升。这是因为卷积核数量的减小一定程度上提取出了更加抽象的特征，当 epoch 比较小的时候，实验容易欠拟合，大的卷积核数量需要更多的 epoch 进行训练。这也是为什么减小卷积核数量可以一定程度上提升精度。

5.3 优化器

针对于优化器问题，前面一节已经提及，这里针对优化器的不同设置实验。为了比较 SGD 和 ADAM 的不同，将原参数保持不变，只针对优化器进行调整。实验结果如实验 17 所示。相比于原文的设置，SGD 的精度下降比较明显，具有 20% 的差距。这也是可以预料到的。因为前面提到过 SGD 是比较朴素的优化器，Adam 进行了进一步的优化，更加适合本次实验的进行。

6 总结与展望

本文对 KDD'2019 的论文:*Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation* 进行了复现，该论文是针对于传统推荐算法领域中存在的问题：冷启动和数据稀疏问题，以及以前对评分与评论相结合的模型中，物品和用户特征交互太少的问题，进行了进一步的优化。设计了局部注意力层、相互注意力层、特征交互、再利用 NFM 机制进行预测评分。该模型具备优秀的效果，在 amazon 的多个数据集中表现优异，均超过了当时的 baseline。在 2019 年之后，该模型经常作为 baseline，在推荐系统评论方向中被比较。本文复现的结果和原论文基本上一致，由于原论文没有参考代码给出，本文参考了其他人复现的代码进行了进一步实验，并且针对于其代码，改善了框架结构。针对于原论文各阶段、参数的不足，进行了创新调整。加入了 self-attention 机制、进行了卷积核维度的调整、对优化器进行进一步调整。本文的代码简单易用，符合操作习惯。

然而，本文还有许多尚未挖掘的方向需要进一步进行实验。在融合阶段中，本文已经对数据进行了三次自注意力的提取，这时再提取特征之间的高阶交互，难免有些冗余，因此，是否采用 MLP 替代 NFM 进行评分的预测，精度可能具有进一步提升的空间。此外，该模型训练时间过长。虽然针对于该问题已经调整了卷积核数量，但是对于大数据集，仍需要 23h 进行训练，不利于模型的调整。原文中并没有针对于模型进行消融实验，最后，在本文的实验中，并没有成功在原论文的参数上调整到和其相同的精度。在今后的探索中，将集中于这些方面。

参考文献

[1] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation. In *KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 344–352, Anchorage AK USA, 2019. ACM.

[2] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys ' 11. ACM, October 2011.

[3] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ' 18*, WWW ' 18. ACM Press, 2018.

[4] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ' 18*, WWW ' 18. ACM Press, 2018.

[5] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, December 2010.

[6] Yu Zheng. Methodologies for cross-domain data fusion: An overview. *IEEE Transactions on Big Data*, 1(1):16–34, March 2015.

[7] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys ' 16. ACM, September 2016.

[8] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ' 17. ACM, August 2017.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW ' 17. International World Wide Web Conferences Steering Committee, April 2017.

[10] Yichao Lu, Ruihai Dong, and Barry Smyth. Coevolutionary recommendation model: Mutual learning between ratings and reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ' 18*, WWW ' 18. ACM Press, 2018.

[11] Libing Wu, Cong Quan, Chenliang Li, Qian Wang, Bolong Zheng, and Xiangyang Luo. A context-aware user-item representation learning for item recommendation. *ACM Transactions on Information Systems*, 37(2):1–29, January 2019.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. https://arxiv.org/abs/1706.03762v7, 2017.

[13] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW ' 16. International World Wide Web Conferences Steering Committee, April 2016.