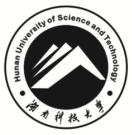
密　　级：公开

中图分类号：TP311



湖南科技大学

Hunan University of Science and Technology

硕士学位论文

基于图嵌入技术的推荐算法研究

研 究 生：郭珈铭

导　　师：文宏副教授

学　　科：计算机科学与技术

研究方向：信息安全基础理论与技术

2022 年 05 月

A Thesis Submitted for the Degree of Master

Research on Recommendation Algorithm based on Graph Embedding Technology

Candidate： Jiaming Guo
Supervisor and Rank： Professor Hong Wen

基于图嵌入技术的推荐算法研究

学位类型	学术型学位
作者姓名	郭珈铭
作者学号	19010501007
学科（专业学位类别）	计算机科学与技术
研究方向（专业领域）	信息安全基础理论与技术
导师姓名及职称	文宏 副教授
实践导师姓名及职称	
所在学院	计算机科学与工程学院
论文提交日期	2022 年 5 月

摘 要

近年来，随着互联网规模的持续扩大，以及基于网络的各种应用的激增，网络中的信息容量急剧增长。在这种情况下，如何快速准确地从海量数据及中找到用户所需的信息成为困扰互联网用户的一大问题。对于该问题，早期的解决方案是以 Google 为代表的搜索引擎，而推荐系统作为一种更主动、更为精准有效的信息过滤技术在近几年得到了广泛的关注与研究。本文对当前的主流推荐方法及其相关技术进行了较深入的研究，主要工作如下：

（1）本文研究发现，基于物品的协同过滤（Item-CF）算法对物品间的相似度要求非常高，若对物品的相似度计算不准确，将会对推荐性能产生很大的影响。针对这个问题，我们在 Item-CF 算法中引入了互信息和自注意力机制，在此基础上设计了一种融合深度游走与自注意力机制的协同过滤推荐算法——DWSA-CF。该算法主要分为两个部分，首先，利用物品间的互信息进行带权随机游走，生成大量物品序列，将其输入到融合自注意力机制的 Skip-Gram 中，用于生成物品的嵌入向量矩阵；然后将生成的物品嵌入向量矩阵结合 Item-CF 算法生成最终的 Top-k 推荐列表。我们在 PyCharm 下实现了 DWSA-CF 算法，同时在三个数据集上进行仿真实验。实验结果表明 DWSA-CF 算法能更精准的计算物品之间的相似度，并且对于数据稀疏度较高的数据集，有更好的推荐效果。

（2）本文研究发现，经典的图嵌入技术算法 GraphSage 存在如下缺陷：一是物品间的关系图是不断动态变化的，特别是当有新物品节点加入时，算法需对每个节点重新计算嵌入向量，大大增加了计算量；二是算法不能很好地解决物品冷启动问题。为解决上述问题，我们引入了分类算法和 LSTM 算法，提出了一种基于分类的 GraphSage 和 LSTM 推荐算法——CLS_GraphSage+LSTM。该算法首先对节点进行分类，采样目标节点的邻居节点，通过学习聚合表示来生成目标节点的嵌入向量；对于新节点，通过将其与现有节点直接相连的方式将新节点加入图中，保证采样时能覆盖到该节点，学习到新节点的嵌入向量，最后结合时间序列信息和 LSTM 算法实现 Top-k 推荐。我们在两个数据集上进行了分类仿真实验，实验结果表明，改进后的 GraphSage 算法节点采样的性能得到了提高；在三个数据集上进行的推荐仿真实验，实验结果表明，CLS_GraphSAGE+LSTM 算法不仅能较为有效地缓解物品冷启动问题，还能减少节点嵌入向量的计算量，提高推荐算法的整体效率。

关键词：推荐算法；图嵌入技术；互信息；自注意力机制；分类；聚合；LSTM

Abstract

In recent years, with the continuous expansion of the scale of the Internet and the explosion of various applications based on the Network, the information capacity in the network presents an explosive growth. In this case, how to quickly and accurately find the information needed by users from the mass data has become a big problem for Internet users. For this problem, the early solution is the search engine represented by Google, and recommendation system, as a more active, accurate and effective information filtering technology, has been widely concerned and studied in recent years. This paper conducts in-depth research on the current mainstream recommendation methods and related technologies, and the main work is as follows:

(1) The item-based collaborative filtering (Item-CF) algorithm has a very high requirement on similarity between items. If the similarity calculation of items is inaccurate, the recommendation performance will be greatly affected. To solve this problem, we introduce mutual information and self-attention mechanism into the Item-CF algorithm. On this basis, we design a collaborative filtering recommendation algorithm -- DWSA-CF, which combines deep walk and self-attention mechanism. The algorithm is mainly divided into two parts. First, the mutual information between items is used to carry out weighted random walk to generate a large number of item sequences, which are input into the Skip-Gram integrated with the attention mechanism to generate the embedded vector matrix of items. Second, the generated items are embedded into the vector matrix and item-CF algorithm is combined to generate the final Top-k recommendation list. We implemented DWSA-CF algorithm under PyCharm, and carried out simulation experiments on three public data sets. Experimental results show that DWSA-CF algorithm can calculate the similarity between items more accurately, and has a better recommendation effect for datasets with high data sparsity.

(2) The classical graph embedding algorithm GraphSage has the following defects: First, the relational graph between items is constantly changing dynamically, especially when new item nodes are added, the algorithm needs to recalculate the embedding vector for each node, which greatly increases the computation amount; Second, the algorithm can not solve the cold start problem well. To solve the above problems, we introduce the classification algorithm and LSTM algorithm, and propose a GraphSage and LSTM recommendation algorithm based on classification -- CLS_GraphSage+LSTM. Firstly, the algorithm classifies the nodes, samples the neighbor nodes of the target node, and generates the embedding vector of the target node by learning aggregate representation of neighbor nodes. For the new node, the new node is added to the graph by directly connecting it with the existing node to ensure that the node can be covered during sampling and the embedding vector of the new node can be learned. Finally, the Top-k recommendation is realized by combining the time series information and LSTM algorithm. We carry out classification simulation experiments on two data sets, and the

experimental results show that the node sampling performance of the improved GraphSage algorithm is improved. The results of recommendation simulation experiments on three data sets show that CLS_GraphSAGE+LSTM algorithm can not only effectively alleviate the cold start problem, but also reduce the computation of node embedding vector and improve the overall efficiency of the recommendation algorithm.

Keywords: Recommender algorithm; Graph embedding technology; Mutual information; Self-Attention mechanism; Classification; Aggregation; LSTM

目 录

第 1 章 绪论.....1
1.1 课题研究背景及意义.....1
1.2 国内外研究现状.....2
1.2.1 经典的推荐算法.....2
1.2.2 基于深度学习的推荐算法.....4
1.3 推荐算法面临的挑战.....6
1.4 本文研究内容.....7
1.5 论文的组织结构.....8
第 2 章 相关理论知识.....9
2.1 互信息.....9
2.2 图嵌入技术.....9
2.3 图神经网络(GraphNeural Networks).....10
2.4 注意力机制.....11
2.5 评价指标.....12
2.6 本章小结.....13
第 3 章 融合深度游走与自注意力机制的协同过滤算法.....15
3.1 算法框架图.....15
3.2 融合自注意力机制的深度游走算法.....15
3.2.1 物品关系图构建.....16
3.2.2 融合自注意力机制的嵌入向量生成.....18
3.3 生成 Top-k 推荐列表.....21
3.4 实验及分析.....21
3.4.1 实验数据集及评价指标.....21
3.4.2 参数设置.....22
3.4.3 基线方法.....22
3.4.4 实验结果分析.....23
3.4.5 网络消融实验.....26
3.5 本章小结.....27
第 4 章 基于分类的 GraphSage 和 LSTM 推荐算法.....29
4.1 相关算法.....29
4.1.1 GraphSage.....29
4.1.2 长短期记忆网络 LSTM.....30
4.2 CLSGraphSage+LSTM 算法.....32
4.2.1 分类采样流程.....32

4.2.2 新节点的采样.....34
4.2.3 聚合函数.....35
4.2.4 算法实现.....36
4.3 实验结果及分析.....39
4.3.1 数据集介绍.....39
4.3.2 实验结果及分析.....40
4.4 本章小节.....43
第 5 章 总结与展望.....45
5.1 总结.....45
5.2 展望.....45
参考文献.....47

第1章　绪论

1.1 课题研究背景及意义

据中国互联网信息中心发布的第四十九次《中国互联网络发展状况统计报告》显示，我国互联网普及率高达 73%，网民规模达到了 10.32 亿[1]。人们可以随时随地在网上查阅资料、发表观点、分享信息，这种现象造成信息规模的不断增长，产生了严重的信息过载问题。这使得用户很难快速地定位到所需的内容，而服务提供商（电影，电商）难以精准地向用户推荐其感兴趣的内容，极大影响了互联网提供服务的效率和质量。如何消除信息过载带来的上述不良影响是研究的重点。

随着用户对高效地信息检索的需求日益增高，研究者们提出了搜索引擎技术。该方法让用户能够快速地从海量数据中检索到所需信息。当用户有清晰的搜索对象或关键词时，就能利用搜索引擎很快地找到所需的信息。而当用户的搜索目标模糊或不明确时，搜索引擎无法根据已知条件推测用户所需内容，因而不能取得良好的效果。由于搜索引擎只能根据搜索的关键词检索相关内容，并不会分析用户喜好，提供个性化服务。在实际使用场景中，当用户搜索关键词“疤痕膏”时，对于孕妇来说，期待得到的结果是祛妊娠纹的疤痕膏；对于普通用户来说，期待的结果只是普通的疤痕膏。基于上述背景，研究者们提出了推荐系统[2]，首先收集行为数据并加以处理和分析，在此基础上建立通用模型挖掘用户兴趣，再根据个人兴趣的差异，推测每个用户的需求和喜好，从而帮助用户在接下来的行为中做出正确的选择。

目前，推荐系统已成为互联网的重要组成部分之一，影响着人们的工作和生活[3]。对用户来说，推荐系统是一个快速便捷的工具，省去从海量数据中搜索的过程。对服务提供商来说，算法可以帮助他们找到潜在用户，改善用户的满意度和留存度，从而提高推荐服务质量，带动收益的提高。例如，视频类平台通过推荐算法直接向用户推荐其感兴趣的视频，提高平台使用者的使用体验，吸引用户并使其更喜欢使用此平台，增加用户黏性。例如国外的 Netflix、Hulu、Youtube 和国内的抖音、快手等。电子商务平台建立的推荐系统通过直接向用户推荐商品，不仅可以增加商品的曝光度吸引潜在用户，提升商品的销售额；还可以捕捉到用户动态的兴趣变化，增加商品成交率。典型平台有国外的 eBay 和沃尔玛，国内的天猫、国美和唯品会等。同时，社交网络平台中通过利用推荐算法可以为用户找到兴趣相似的朋友。如何对主流推荐算法进行有效的改进，提高推荐的准确性和快速性，满足人们对高性能推荐的需求，是研究人员努力的方向。

1.2 国内外研究现状

1.2.1 经典的推荐算法

推荐算法最早由 Xerox 公司提出并应用于邮件的识别和过滤[4]。常见推荐系统的主要组成部分如下：

（1）用户信息：用户的性别、年龄等注册信息；收藏、分享、历史观看记录等交互信息。本文使用的是用户历史观看电影的数据信息。

（2）物品信息：候选物品集中物品的基本信息，例如电影的年份、类型、主演等。

（3）场景信息：用户所处的环境，包括用户的网络环境和现实环境，例如正在旅游、放假或者备战考试等。

（4）召回层：通过有效的召回算法，利用数据的部分信息，快速地从百万级的物品集中筛选得到一个数百级的候选物品集。

（5）排序层：对召回层得到的候选物品集中的物品进行排序。首先选取合适的排序算法，通过算法计算评分，最后按评分的高低对物品进行排序。

（6）再排序层：为了推荐列表中物品的新鲜度、多样性等因素，对排序层的列表进行修改，得到最终用户可见的列表。

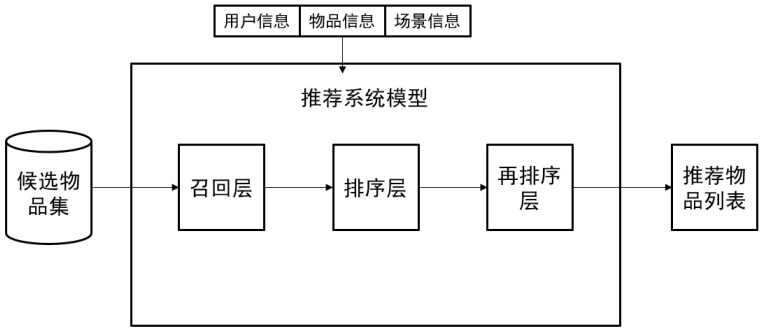


图 1.1 推荐系统逻辑框架图

Fig. 1.1 Recommendation system logic framework diagram

图 1.1 描述了一个典型的推荐过程，针对特定的用户，在特定的场景下，使用推荐算法在众多候选物品集中，为用户选出其感兴趣的物品。其中推荐算法是流程中的关键问题之一，接下来介绍几种经典的推荐算法。

（1） 协同过滤推荐算法

CF 算法的核心是计算相似度，根据相似度结果和历史评分数据，求得用户对未交互物品的预测得分并进行推荐[5]。该算法不需要对物品本身的信息进行提炼和分析，具有很强的普适性。但其推荐性能完全依赖于用户对物品的历史评分数据的质量，极易受到影响。其分为 User-CF 和 Item-CF。

User-CF 从用户角度出发，计算用户之间的相似度，挖掘与用户爱好相似的同类用户，根据同类用户的喜好物品集，然后把这些集合的并集作为给目标用户的推荐列表。

User-CF 算法的优点在于，算法挖掘用户喜好，根据用户之间的相似喜好挑选物品进行推荐，这个过程中物品不存在直接联系，因而生成的推荐列表中的物品具有多样性。但这类算法也有如下缺点：

1）对于不同的用户数量级，相似度矩阵的计算量不同。显然，当用户数量较少时，评分矩阵规模小，容易计算。。而当用户数量级过大时，对计算资源需求过大。

2）无法对用户的推荐请求及时响应。当用户对物品产生了新的交互，该算法并不会对推荐结果及时更新。

3）当新物品进入系统时，会产生冷启动问题。

Item-CF 从物品角度出发，计算物品间的相似度，来为用户进行物品推荐。

Item-CF 算法是通过计算物品之间的相似度，生成的是丰富的长尾物品列表，能更好地进行个性化推荐。但算法也有如下两个缺点：

1）当物品数据集非常大时，对计算资源的消耗过大。因此，该算法更适合用于物品数据集较小的情况下。

2）新用户进入系统时，只有当其在系统中留下交互记录后，才能建立该用户的推荐模型，从而为其进行推荐。

（2） 基于内容的推荐算法

该算法通过对内容进行分析，筛选和设置用做推荐的特征。核心思想是提取物品的部分特征作为物品的表示，计算特征之间的相似度[6]。它不依赖于交互数据的质量，因而算法性能较为稳定。在推荐过程中，算法通过计算特征之间的相似度，将值排序后选取高的特征推荐给用户。然而算法在提取物品信息中仍然存在着一些缺陷。例如，在电影推荐中，少量的标签难以全面的表达电影信息，因此如何给电影分类和打标签是一个相对困难的问题，对于此类多媒体信息，本算法很难实现有效的推荐。

该算法的核心围绕物品的相关内容和用户的喜好，具有以下优点：

1）可解释性强。该算法根据与物品有关的内容和用户的兴趣进行推荐，因而具有较好地可解释性。

2）不会产生物品冷启动问题。对于新进入系统的物品，算法仅需要根据物品的特征信息便可进行推荐，并且得益于算法机制，新物品与老物品被推荐的概率相等，且不会产生物品冷启动问题。

3）对每个用户进行独立的个性化推荐。该算法对用户的推荐基于其喜好，因此某些用户对某物品的恶意操作或者过于偏爱并不会影响算法将该物品推荐给其他用户，这样能够建立每个用户独立的偏好模型，实现个性化推荐。

该算法也有如下缺点：

1）仅仅从物品的特征上进行相似度分析计算，只能得到客观上相似度最高的物品，较难挖掘用户的潜在兴趣，推荐结果较单一。

2）对于新用户，由于缺乏用户的历史交互记录，无法根据其兴趣进行推荐。

（3） 混合推荐算法

由于上述两种算法存在各自的优势和劣势，在实际推荐中单独使用一种算法不能很好的应用于所有推荐场景，很难取得较好的效果。因而混合推荐将多种推荐算法通过不同的组合方式组合起来，吸取各算法的优势，给出更合理的推荐列表，提高算法的推荐性能。李雪婷等人[7]采用级联方式将基于矩阵分解的协同过滤算法和基于内容的推荐算法组合起来，用于解决数据稀疏性和冷启动问题。张润莲等人[8]为解决推荐中存在的隐私泄露问题，将差分隐私和混合相似度计算方法融合起来，构建了混合协同过滤推荐算法，不仅能保护用户的隐私，还能提高推荐的准确率。Campos 等人[9]结合贝叶斯网络算法，将两种推荐算法的特征进行组合来解决推荐问题，并在 Moivelens 数据集上进行验证。王粤等人[10]首先得到用户的喜好，然后计算出用户对不同属性物品的权重和均值，用于填补评分矩阵中的缺失项，并通过实验结果证明此方法能有效缓解数据稀疏性问题。田保军等人[11]通过融合隐迪利克雷分布和卷积神经网络构造概率矩阵分解模型，该方法考虑物品评论文本的主题信息和深层语义信息的不同，将用户属性信息和深层次的物品特征结合应用于概率矩阵分解模型中，通过计算得到用户对物品的评分数据。Dong 等人[12]将新闻热点参数添加到用户相似度中，改进预测参数，采用混合推荐技术预测用户收视率。

1.2.2 基于深度学习的推荐算法

深度学习能够自动学习复杂的非线性网络构成[13-14]，采用矩阵计算进行计算，能够学习到各种对象的非线性特征，在构建复杂模型和处理异构数据上具有天然优势，因此，基于深度学习的推荐算法[15-19]受到了广泛的关注。在 ACM 推荐系统年会中，研究者们也强调，深度学习相关算法与传统推荐算法的结合将是未来推荐系统领域重要的研究方向之一[20-24]。接下来介绍几类基于深度学习的推荐算法。

（1） 基于图嵌入技术的推荐算法

该算法把图映射为一个嵌入向量矩阵。方法是将图中的每个顶点赋予一个初始向量，这些向量包含了图的拓扑结构信息[25-29]、空间信息[30-35]等。在推荐系统中，可利用历史数据构造用户物品关系图，因此，研究者们能够有效地将图嵌入技术应用到推荐算法中。

一些研究通过生成物品的嵌入向量来为用户推荐物品。Item2Vec[34]将词嵌入技术融入到 Item-CF 算法中，利用词嵌入技术学习物品的嵌入向量，将相似度高的物品推荐给

用户。MetaProd2Vec[36]利用用户和物品的历史交互信息以及物品的属性来得到物品的嵌入向量。为解决数据稀疏性和冷启动问题，EGES[37]将不同权重的属性信息融入到图嵌入模型中。SISG[38]在 Skip-gram 算法中加入了辅助信息，挖掘用户行为的非对称性，得到更精准的物品嵌入向量。

另一些研究则是利用图嵌入技术来生成用户和物品的嵌入向量为用户提供个性化推荐。IGE[39]设计了一种包含属性信息和时序信息的交互图，该算法分为编码网络和预测网络两部分，第一部分通过将属性信息转换为定长的数组来处理属性信息的异质性，第二部分用来挖掘时序依赖信息。HERec[40]将基于元路径的随机游走应用于异构图中，用于生成节点序列，并通过融合函数学习得到节点的嵌入表示，将其扩展到矩阵分解模型中。RKGE[41]是一种基于知识图谱的嵌入学习方法，通过自动学习实体与实体之间的路径，以表示用户对物品的喜好。KGAT[42]将两个属性相连的节点连接起来，打破了传统算法将用户和物品间的联系看作是独立的假设，并在此基础上挖掘出了节点之间的高阶联系。

（2） 基于注意力机制的推荐算法

注意力机制[43]起源于生命科学，是对人眼视觉原理和人眼视线焦点转移的研究。在认知科学中，由于人脑存在着信息加工的“瓶颈”，人们往往只会把注意力集中在信息的某一方面，而忽视掉其它有效的信息。注意力机制的要点是确定哪些是需要关注的重要信息，从而合理分配信息处理资源。对于推荐算法来说，面对海量的物品，用户很难关注到所有的物品。因此，为了更好地为用户推荐物品，研究者们将注意力机制应用到推荐算法中用于捕捉用户兴趣的变化[44-47]。

DIN 算法[48]通过用户对历史数据进行分析来计算其对当前物品的注意力值，对用户的每个不同兴趣赋予不同的权重，然后再加权求和。但是 DIN 算法中用户兴趣是固定的，无法捕捉到用户兴趣的动态改变，而且无法确保用户对物品所产生兴趣的有效性。为此，DIEN 算法[49]被提出。它是 DIN 的改进版本，该算法提出了一种新的兴趣抽取层和兴趣进化层来解决上述两个问题。引用时间信息来更加精准地捕捉用户的长短期兴趣，在兴趣变化时提高相关兴趣的影响因子，削弱无关兴趣的影响因子，以提高用户兴趣表达的准确性。AttRec[50]是基于自注意力机制的序列推荐算法，不仅利用自注意力机制从用户的交互记录中学习短期兴趣，还利用度量学习的方法来保留用户的长期兴趣。AHNER[51]考虑到不同节点的属性信息对推荐结果具有不同的影响，利用自注意力机制来挖掘节点属性信息，学习更准确的嵌入表示来提高推荐准确性。

（3） 其他深度学习推荐算法

上述两种深度学习推荐算法虽然各有优势，并且在大部分的推荐场景中表现更优，但仍存在算法训练向量耗时等问题。因此，研究学者们将其他算法与深度学习算法融合

起来，各取所长，从而解决上述问题。例如，石宜金等人[22]将聚类算法与深度学习算法融合，建立关联规则作为评估用户之间相似性的标准，为解决局部最优解，将其与协同过滤算法融合。Liu 等人[23]利用循环神经网络来捕捉序列行为之间的依赖关系，从而用于解决位置社交网络中的行为预测问题。Wang 等人[24]将贝叶斯网络和降噪自编码器引入推荐算法中，实验结果表明该方法在稀疏数据集上的推荐表现更好，因而可作为解决数据稀疏性问题的方案之一。

基于上述研究，本文对基于图嵌入技术的推荐算法进行改进，提出了两种图嵌入技术推荐算法，不仅更精准地计算相似度，对稀疏度较高的数据集有较好的推荐结果；而且能较为有效地缓解物品冷启动问题以及加快计算速度，提高推荐算法整体效率。

1.3 推荐算法面临的挑战

近年来，众多研究者研究推荐算法，加快了其发展的速度。虽然与推荐算法有关的技术越来越多，但是由于算法的局限性，推荐算法还存在着以下问题。

（1） 冷启动问题

当新用户进入平台时，数据库中没有该用户任何历史信息，只包含用户注册时记录的部分个人信息，例如性别、年龄、职业等。经典的推荐算法无法对其进行推荐。同理，当新物品进入系统时，系统中只存在该物品的类别、日期、价格等基本信息，不存在任何该物品的交互信息。经典的推荐算法同样无法将其精准地推荐给潜在用户。针对上述问题，对于新用户，当前的方法是为用户提供当前热门的物品，等到后台收集到用户的相关喜好数据后，再为其进行个性化推荐。对于新物品，可以利用矩阵奇异值分解等矩阵降维办法，对矩阵进行填充，扩展出物品的交互数据，再进行推荐。

（2） 数据稀疏性问题

推荐系统中的数据稀疏性是指评分矩阵中出现越来越多空白的值或数值为零的现象，导致依赖于数据的推荐系统的性能下降，是推荐系统不可避免的问题。为缓解该问题，研究者们通常采用特殊方法选取或计算出一些固定数值，将其填充到矩阵的空处。经过实践检验，该方法有效缓解了部分场景中的数据稀疏性问题，但同时也带来了噪声问题。

（3） 兴趣变化问题

在实际场景中，用户的兴趣随着时间在时刻变化，而已有的推荐系统大都是基于用户的历史数据而构建，这种系统已经能够准确的捕捉用户过去的兴趣爱好，但不能很好地捕捉到用户近期兴趣的变化。因此，推荐算法需要充分考虑到影响用户兴趣变化的因素，以防止在推荐的过程中用户的兴趣已经发生改变，从而影响推荐准确性。目前，研究者们通过利用注意力机制来捕捉用户的长短期兴趣。

（4） 稳定性问题

当推荐算法进行推荐时，需要获得其他用户对物品的评价和打分，进行分析与建模。所以，推荐算法对评分数据的真实性非常敏感。如果出现恶意虚假的评分数据，这会导致推荐错乱，影响推荐结果。因此，一个优秀的推荐算法必须能敏感地识别出虚假评分信息，从而提高推荐稳定性。

（5） 推荐算法的实时性

实际场景中的推荐系统往往存储着海量的用户物品信息，因此，推荐算法必须及时有效地处理 这些数据，为用户提供更符合其期待的物品。推荐算法的更新速度越快，系统就越能了解最新的资讯，越能迅速掌握最新的潮流，越能更快为用户进行推荐。所以，一个优秀的推荐算法需要为用户提供实时的推荐服务。针对这一问题，研究者们将模型训练和推荐过程分离为线上和线下两个部分，线上采集用户各种交互数据，然后把它们提供给线下的模型进行训练，将最优模型部署到线上，进行实时推荐。该方法虽然取得了一定的成效，但未从源头上解决。

（6） 可解释性问题

向用户提供有解释性的推荐，不仅更好地为其进行推荐，而且还能使他理解推荐的过程，让用户更加清楚明了地使用平台，这对提高系统的准确性、用户满意度有一定的帮助。

1.4 本文研究内容

本文针对 Top-k 推荐，主要研究基于图嵌入技术的推荐算法。为了更好地计算物品之间的相似度，有效地解决物品冷启动问题，本文开展了一系列研究工作，主要研究内容如下：

（1）对基于物品的协同过滤、图嵌入技术、注意力机制等技术进行了深入的研究。

（2）为了更好地计算物品之间的相似度，提高推荐准确率。本文提出了一种融合深度游走与自注意力机制的协同过滤算法——DWSA-CF。DWSA-CF 算法首先利用用户和物品的历史交互记录以及时间序列信息构建物品间有向关系图，将互信息作为有向边的权重，生成物品序列；然后使用自注意力机制区别对待当前物品序列中的邻居，更关注于与当前物品更为相似的物品，从而能更精准地学习到物品的嵌入向量矩阵；最后结合 Item-CF 算法进行 Top-k 推荐。

（3）为了有效地解决物品冷启动问题，提高推荐算法的整体效率。本文提出了一种基于分类的 GraphSage 和 LSTM 推荐算法——CLS_GraphSage+LSTM。该算法对 GraphSage 算法的采样方法进行改进，在节点采样前根据节点的属性信息对其分类，再通过计算节点的度中心性得到每一类中的重要节点，当有新节点加入时，将其与重要节

点直接相连，使得算法能对新节点进行采样。最后结合时间序列信息与 LSTM 算法实现 Top-k 推荐。

1.5 论文的组织结构

本文共分为五章，为节省计算资源和解决冷启动问题，提出了两种图嵌入推荐算法，主要内容分为如下。

第一章：绪论部分。介绍了课题的研究背景及意义，国内外研究现状以及面临的挑战，最后介绍了本文的主要研究内容以及论文的组织结构。

第二章：相关理论知识部分。介绍了本文使用到的几种深度学习技术，并介绍了常用评价指标。

第三章：融合互信息的深度游走考虑了随机游走时，不同物品之间的跳转概率应是不一样的，将物品间的互信息设为物品间的跳转概率，能够更好地计算物品之间的相似度。结合 Item-CF 进行 Top-k 推荐。还通过实验得到了所需最优参数，在三个真实数据集上进行对比实验并对实验结果进行了分析，并且对提出的 DWSA-CF 算法进行消融实验，证明了融合的有效性。

第四章：对本文提出的基于分类的 GraphSage 和 LSTM 算法进行了详细的介绍。首先介绍了 GraphSage 算法，并对其进行分析，针对其无法很好地解决新物品的冷启动问题，对其采样方法进行改进。并且针对图中节点嵌入向量计算时，存在重复计算，消耗过多的计算资源这一问题，提出了 GraphSage 算法的改进优化方案。CLS_GraphSage+LSTM 算法首先对物品节点按属性进行分类，随后基于分类和相邻关系对每个节点的邻居进行采样，学习邻居节点的聚合表示，从而生成出目标节点的嵌入向量。对于新节点，将其与分类后度中心性高的节点直接相连，从而生成新节点的嵌入向量。最后结合时间序列信息和 LSTM 算法进行 Top-k 推荐。

第五章：总结与展望。对本文所研究的推荐算法进行总结。

第2章　相关理论知识

本章对后续涉及到的关键技术的原理进行介绍，主要包括互信息、图嵌入技术、图神经网络和注意力机制等，最后对常见的评价指标进行了介绍。

2.1 互信息

互信息最早由 Woods 于上世纪 90 年代在医学图像领域提出，是一种重要的信息度量方法[59]。通过采用两个随机变量共享的信息量来度量它们之间的关系。因此，既能表明两个变量是否有关系又能衡量变量之间的关联程度。将物品作为变量，采用互信息既能说明物品之间有关系，又能反应它们之间的强弱，这使得它能够很好地表示出两个物品之间的相似性。

2.2 图嵌入技术

图嵌入技术主要包含“图”和“嵌入”两个部分。图是图嵌入技术的处理对象，可分为同构图、异构图和语义图等。“嵌入”是指通过神经网络学习将图中的顶点映射到一个低维向量空间中，这个映射过程即为嵌入。图嵌入技术使得图更容易地存储和表示。图嵌入技术的目的是得到图中各个顶点的向量表示，并使其最大限度地保留图中顶点的结构相似性和顶点间的距离。图嵌入技术由于其可伸缩性和可适应性，能够快速地完成大量的图处理，即使图的规模不断扩大，也不需要反复学习。近年来，大量的图嵌入技术广泛应用于自然语言处理、节点分类、节点聚类、推荐系统等领域。

图嵌入技术主要分为直接在图上定义损失函数的方法、基于随机游走的方法和利用图神经网络模型对图嵌入学习的方法这三类。本文后续工作使用到的 DeepWalk[24]是基于随机游走的方法。它是一种广泛使用的图嵌入技术，自 2014 年提出之后被广泛使用到 NLP、推荐系统中。该算法通过对图的拓扑结构和语义信息进行建模，将图中每一个节点表示为蕴含着语义信息的低维稠密的向量。DeepWalk 参考了 Word2vec 训练词嵌入向量的思想，使用随机游走在图中采样，生成具有现实意义的节点序列，再利用 SkipGram 算法对序列进行向量化处理。

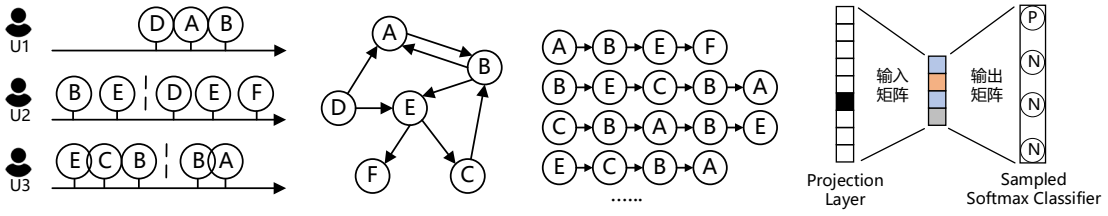


图 2.1 DeepWalk 工作示意图
Fig. 2.1 DeepWalk working diagram

2.3 图神经网络（Graph Neural Networks）

GNN 是一种新颖的深度学习技术，与其他神经网络相比较，它的优势是有较好的可解释性和推理能力；但其缺点是网络结构浅，不能构建深层次的网络，且对于非结构化数据和动态图的处理能力较差[52]。GNN 的特性使得它被广泛应用于商业推荐、自然语言处理、知识图谱和序列推荐等领域。

在图中每个节点可以由它自身的属性特征和其邻居节点的特征来定义，图神经网络利用这种特性来定义图中每一个节点。在图神经网络中，图中的每一个节点都被表示为一个嵌入向量。GNN 利用邻居信息学习出每个节点的嵌入向量表示，其核心思想是通过逐层聚合目标节点的邻居信息，最后与目标节点的信息结合，得到目标节点的嵌入表示。

近年来，图神经网络发展出多种框架，主要有图卷积网络[53]（Graph Convolutional Network，GCN）、图注意力网络[54]（Graph Attention Networks，GAN）、门控图神经网络（Gated Graph Sequence Neural Networks，GGNN）[55]、GraphSage[56]四种。下面简要介绍它们的原理。

设节点 v 是目标节点的第 k 层邻居中的一个节点，$h_v^{(k)}$ 为节点 v 的嵌入向量表示，N_v 为 v 的邻居节点集合，$n_v^{(k)}$ 为节点 v 在第 k 层的邻居节点聚合嵌入向量表示，$W^{(k)}$ 为第 k 层的线性变换参数矩阵。

（1）图卷积网络更新嵌入向量的计算公式如下。

$$H^{(k+1)}=\theta\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k)}W^{(k)}\right) \tag{2.1}$$

其中，$H^{(k)}$ 为第 k 层的嵌入矩阵，$\theta(\cdot)$ 为非线性激活函数，$\tilde{A}=A+I$ 为无向图附加自连接的邻接矩阵，A 为邻接矩阵，I 为单位矩阵，$\tilde{D}_{ii}=\sum_j \tilde{A}_{ii}$ 为度的正则化参数。

（2）图注意力网络利用注意力机制学习不同邻居的重要性，通过计算出目标节点和其邻居的关联程度，为邻居节点的信息赋予不同权重，更新节点的嵌入向量表示。计算公式为：

$$h_v^{(k+1)}=\theta\left(\sum_{j\in N_v}\alpha_{vj}W^{(k)}h_j^{(k)}\right) \tag{2.2}$$

其中 $h_j^{(k)}$ 表示第 k 层邻居中节点 h_v 的邻居 h_j 的嵌入表示，α_{vj} 表示邻居节点 h_j 的权重。

（3）门控图神经网络采用门循环单元 GRU 来更新节点信息，计算过程如下：

$$n_v^{(k)}=\frac{1}{|N_v|}\sum_{j\in N_v}h_j^{(k)} \tag{2.3}$$

$$h_v^{(k+1)} = GRU(h_v^{(k)}, n_v^{(k)}) \tag{2.4}$$

其中 GRU 是门控神经网络。

（4）GraphSage 首先采样目标节点的一定数量的邻居节点，随后提供多种聚合函数进行信息聚合，将聚合结果与目标节点的嵌入向量进行拼接，最后乘以一个非线性函数完成嵌入向量的更新。聚合过程如下：

$$n_v^{(k)} = Aggregator_t(h_u^{(k)}, \forall u \in N_v) \tag{2.5}$$

$$h_v^{(k+1)} = \theta\left(W^k \cdot CONCAT\left(h_v, n_v^{(k)}\right)\right) \tag{2.6}$$

其中， *Aggregator* 表示聚合操作， *CONCAT* 代表向量拼接操作。由于节点采样方法和聚合函数的类型繁多，因而 GraphSage 模型更有灵活性。

2.4 注意力机制

注意力模型是模拟人脑注意力的一种模型，最早应用于计算机视觉领域。当人类观察物体时，视野中不会出现物体全貌，而是聚焦于物体的一部分。注意力机制模拟了这个过程，主要关注重要的信息，而忽略不重要的信息。

2014 年，Google Mind 首次将注意力机制引入自然语言处理领域，利用该机制学习实验所需的属性特征信息，而将不相关的特征剔除，从而减少训练过程中的计算量。由于注意力机制能够很好的表达偏好，人们对它的研究也更加深入，应用领域也得到了极大的扩展。注意力机制不仅在自然语言处理领域取得极大的成功，还在图像处理、推荐系统等领域也取得了成功。例如在推荐领域中，用户不同行为所表达的实际意义并不相同。通过引入注意力机制，系统可为用户的不同行为赋予不同的权重，以获得更准确的物品嵌入向量，得到更好的推荐结果。

注意力机制[43]的本质是为不同的对象或特征赋予不同的权重。首先为每个特征构建 query（查询）、key（键）和 value（值）三个向量。通过计算两个不同特征间的 key 和 query 乘积来衡量二者的相似度，最后与 value 加权求和，求得每个特征的注意力值。过程如图 2.2 所示。

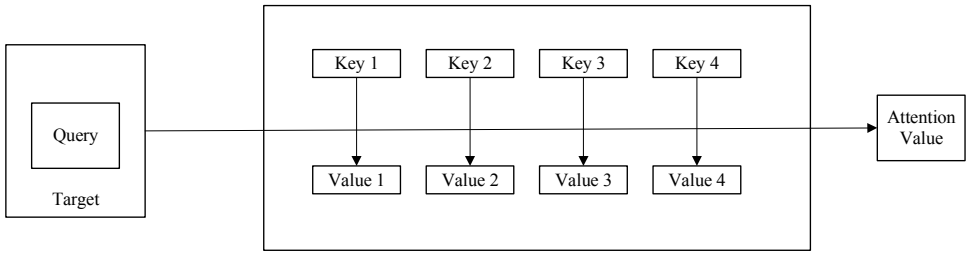


图 2.2 注意力机制结构图

Fig. 2.2 Structure of attention mechanism

近年来，随着深度学习不断发展，注意力机制也衍生出许多分支类型。自注意力机制是一种特殊的注意力机制，它的重点放在特征内部，其主要作用是得到特征内部各元素的注意力。在自注意力机制中，为了减少计算量，通常令 key 和 value 的值相同。图 2.3 为自注意力机制计算过程。

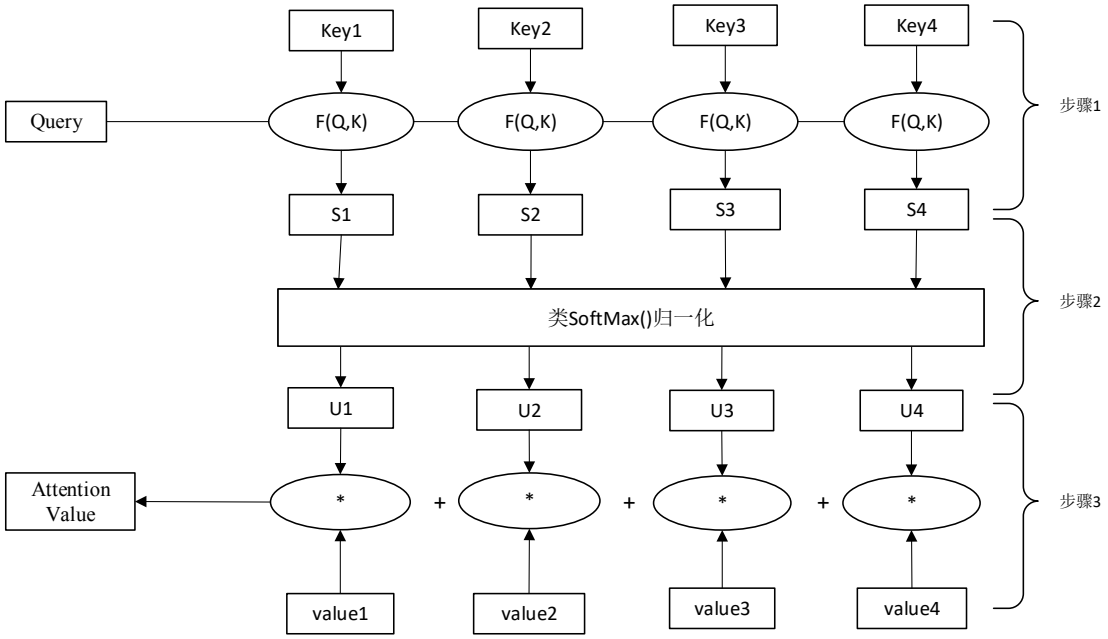


图 2.3 自注意力机制计算图

Fig. 2.3 Compute steps from the self-attention mechanism

2.5 评价指标

推荐算法是通过计算用户对物品的预测评分，从而生成推荐列表。在 Top-k 推荐列表中，我们要把最容易被用户喜欢的物品放在首位，所以，我们要对推荐算法进行评估，使得推荐列表中有更多精准的物品放在列表的前面。如何评价一个推荐算法的性能，常用召回率（Recall）、F1 值（F1-source）和覆盖率（Coverage）来评价。可依据表 2.1 所示的评价指标辅助表计算得到召回率和 F1 值。

表 2.1 评价指标辅助表

Tab.2.1 The auxiliary table of evaluation index

数据类别	预测为推荐物品	预测为过滤物品
实际为推荐物品	TP	FN
实际为过滤物品	FP	TN

准确率（Accuracy）是指被正确分类的物品在所有物品中的占比，通常来说，准确率越高，推荐性能越好：

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

精准率（Precision）用来衡量推荐的准确程度。根据算法得到一个推荐列表，用户实际交互过的物品在该列表中所占的比例即为精确率，计算公式如 2.8 示：

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.8}$$

召回率（Recall）用来衡量物品的查全率，是推荐成功的物品占用户交互列表的比例，计算公式如 2.9 所示：

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.9}$$

F1 值是精准率和召回率的调和平均数，用于对精确率和召回率进行整体评价。在实际推荐过程中，精确率和召回率可能会出现矛盾，无法用它们对推荐结果进行准确评价。研究者在二者基础上定义 F1 值，利用进行综合评价，计算公式如 2.10 所示：

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.10}$$

覆盖率（Coverage）是指推荐列表所提供的物品数量在整个物品集中的百分比，是一种用于描述长尾物品挖掘的评价指标。

2.6 本章小结

本章主要分为两个部分，一是介绍本文使用的相关方法，对其中的关键技术进行了较为详尽的分析。二是介绍推荐系统中的一些评价指标，为后续研究做好充足的准备工作。

第3章　融合深度游走与自注意力机制的协同过滤算法

本章结合深度游走和自注意力机制的特点和优势，提出了一种融合深度游走与自注意力机制的协同过滤推荐算法。详细描述了算法的设计思想及实现过程，在三个公开数据集上进行了仿真实验，并对实验结果进行了理论分析。

3.1 算法框架图

在 Item-CF 算法基础上，本章融合深度游走算法和自注意力机制，构建出融合深度游走与自注意力机制的协同过滤推荐算法（A Collaborative filtering Recommendation Algorithm Based on Self-Attention Mechanism and DeepWalk，DWSA-CF），框架如图 3.1 所示：

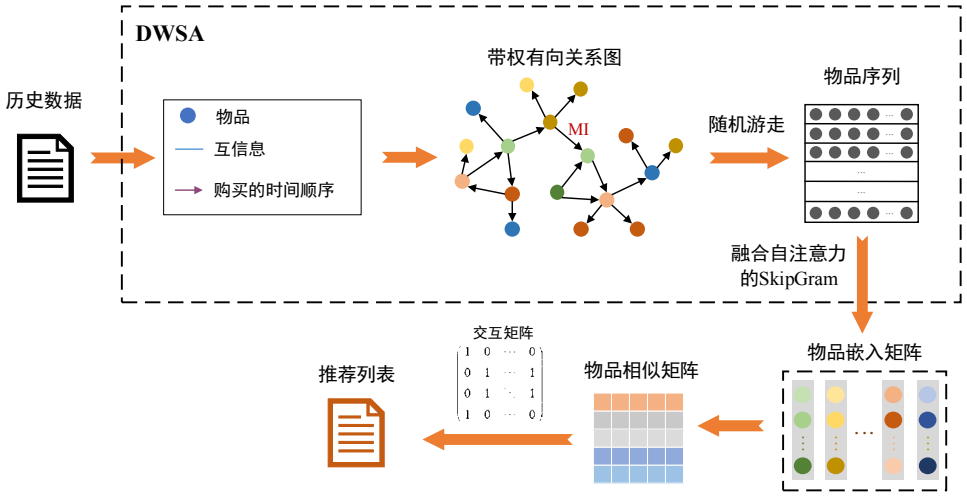


图 3.1 融合深度游走与自注意力机制的协同过滤推荐算法

Fig. 3.1 The DWSA-CF Framework

DWSA-CF 算法先从原始数据中抽取出用户的交互记录，将物品作为节点，根据交互时间的先后顺序确定有向边方向，以物品间的互信息作为权重，构建出物品之间的带权有向关系图；然后利用随机游走在关系图上生成大量的物品序列作为后续步骤的训练序列，同时将自注意力机制引入到 DeepWalk 中 SkipGram 算法的训练过程中，得到物品的嵌入向量矩阵；最后，采用 Item-CF 算法，根据嵌入向量矩阵计算出物品的相似矩阵，再结合用户的交互矩阵完成 Top-k 推荐。

3.2 融合自注意力机制的深度游走算法

DWSA 算法主要包含物品间有向图的构建、带权随机游走和融合自注意力机制的

SkipGram 训练过程三个部分，算法如下：

算法 3.1 DWSA 框架

输入：数据集 D，物品关系图 $G(V, E)$，随机游走中每一步的长度 l，从每个节点出发进行随机游走的次数 η，SkipGram 的窗口大小 w，嵌入向量的维度 d，以节点 v_i 为起点得到的序列 S_{v_i}。

输出：嵌入向量矩阵 M。

1. $G = DirectdGraph(D)$
2. 初始化 $M(|M| = |V| \times d)$
3. For $i = 0$ to η do：
4. For $v \in V$ do：
5. // 进行带权随机游走
6. $S_{v_i} =$ **WeightedRandomWalk**(G, T, u_i, l)
7. // 采用融合 Self-attention 的 SkipGram 操作更新 M
8. **SkipGram With SA**(M, S_{v_i}, w)
9. End For
10. End For
11. 返回 M

3.2.1 物品关系图构建

将数据集中用户对物品的交互记录转换为有向图，并计算节点间的互信息作为边上权重，具体过程如下：用 $U(u \in U)$ 和 $I(i \in I)$ 代表用户集和物品集，其中 $|U| = m$，$|I| = n$。根据用户历史交互数据，得到用户 u 的物品交互序列为 $L_u = (i_1, i_2, \ldots, i_h)$，该序列根据用户和物品交互的时间顺序进行排序。设物品关系图为 $G = (V, E)$，其中，V 表示与用户交互的物品集合，即 $V = I$。如果用户 u 与物品 i_a 的交互时间早于 i_b，那么存在有向边 $e(i_a, i_b) : i_a \to i_b$，所有这样的边集合 E 构成有向图 G。

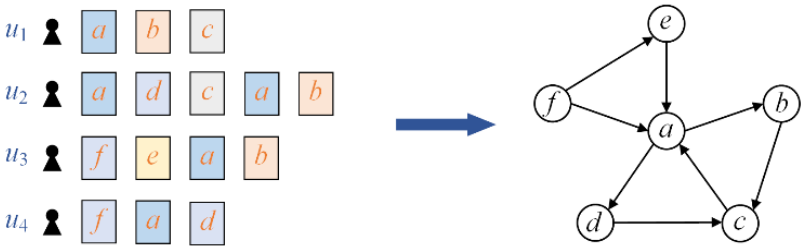


图 3.2 物品关系图

Fig. 3.2 Item relation diagram

如图 3.2 所示，用户 u_1 先后与物品 a、b、c 进行了交互，那么存在有向边 $a \to b$ 和 $b \to c$。由此将历史数据中用户与物品的交互记录转换成有向图 G。DWSA 的目标是对 G 中每个节点学习出一个嵌入向量表示 $\Phi: V \to \mathbb{R}^d$。其中，d 是向量长度，且 $d \ll |V|$，一般来说数据集越大则 d 越大。

采用随机游走方法在图 G 上生成固定长度的物品序列。由于 DeepWalk 算法在无向无权图上进行随机游走，无法保证生成的序列中相邻节点之间的相似性，并且随着序列数量的增加，会出现大量形如 *abcabc* 的重复无意义序列，浪费计算资源。为解决上述问题，DWSA 在图 G 中添加互信息作为边上的权重，根据权重计算节点间跳转概率，使得随机游走时跳转到相似物品的概率更大。这样能让生成的序列中物品之间具有较强相似性，提高嵌入向量的训练效率。

本章采用 2.1 节介绍的互信息来衡量物品间的相似性。

$$I(v_i, v_j) = \log \frac{p(v_i, v_j)}{p(v_i)\, p(v_j)} \tag{3.1}$$

其中，$p(v_i)$ 和 $p(v_j)$ 分别表示物品 v_i 和 v_j 在历史交互数据中出现的概率，$p(v_i, v_j)$ 表示物品 v_i 和 v_j 在 L_u 中同时出现的概率。$I(v_i, v_j)$ 越大表示 v_i 和 v_j 的相似性越强。

通过对节点间互信息进行归一化，得到 G 中节点 v_i 跳转到 v_j 概率为：

$$P(v_j \mid v_i) = \begin{cases} \dfrac{e^{I(v_i, v_j)}}{\sum_{j \in N_+(v_i)} e^{I(v_i, v_j)}}, & v_j \in N_+(v_i) \\ 0, & e_{ij} \notin E \end{cases} \tag{3.2}$$

其中，$N_+(v_i)$ 是节点 v_i 的所有出边的集合。如果 $N_+(v_i) = \varnothing$，那么跳转到起点。

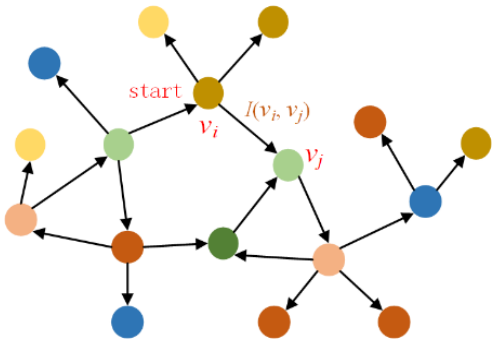


图 3.3 随机游走示意图

Fig. 3.3 Schematic diagram of random walk

随机游走过程：以节点 $v_i\,(v_i \in V)$ 为起点，根据公式 3.2 按概率随机得到下一个节点 v_{i+1}，重复 $l-1$ 次，得到一条长为 l 的物品序列 $seq = v_i v_{i+1} \ldots v_{i+l-1}$，如图 3.3 所示。设置随

机游走的步长l和次数η，以G中每个节点作为随机游走的起点，生成n条长度为l的物品序列；重复该过程η次，最终得到$n*\eta$条长度为l的物品序列。算法如下：

算法 3.2 带权随机游走算法——**WeightedRandomWalk**$(G,\ T,\ u_i,\ l)$

输入：物品关系图$G(V,\ E)$，随机游走中每一步的长度l，节点$u_i \in V$。

输出：节点序列seq。

初始化转移矩阵$T(n \times n)$

For $node \in G.nodes$ do：
    // 遍历$node$的邻居节点
    For $neigh \in node.neighbor$ do：
        // 得到随机游走的转移矩阵
        $T_{node,neigh} = sum\left((node : node.neigh).weight\right)$
    End For
End For
$seq = v_i$
For each $v_j \in v_i.neighbor$ do：
    // 根据转移矩阵得到随机游走的下一个节点
    $seq = seq + weightedRandom\left(v_j.neighbor,\ T_{v_j}\right)$
    $v_i = v_j$
End For
返回seq

3.2.2 融合自注意力机制的嵌入向量生成

用 SkipGram 算法实现对图 G 的嵌入表示$\Phi:\ V \to \mathbb{R}^d$。它通过在物品序列的最左侧设置起始窗口，用窗口中心词预测上下文出现的概率，并通过移动窗口更新嵌入向量，直至所有序列遍历完毕。首先将集合$V=\{v_1, v_2, \ldots, v_n\}$的每个节点映射为嵌入向量集合$\boldsymbol{E}=\{\boldsymbol{e}_1,\ \boldsymbol{e}_2,\ \ldots,\ \boldsymbol{e}_n\}$，设窗口大小为$t$，中心词为$v_c$，上下文$v_b=\{v_{c-t},\ldots,v_{c-1},v_{c+1},\ldots,v_{c+t}\}$，假设上下文之间是相互独立的，那么中心词与上下文共现的概率为：

$$P\left(v_b \middle| v_c\right)=\prod_{i \in v_b} P\left(v_i \middle| v_c\right) \tag{3.3}$$

SkipGram 通过最大化公式 3.3 的值来学习物品的嵌入向量，采用 softmax 函数计算条件概率$P\left(v_b \middle| v_c\right)$，由于 softmax 函数的最小值比最大值更容易求，定义目标函数 L 如下：

$$L=\min \sum_{j=-t,j\neq 0}^{t} -\log P\left(v_{c+j} \middle| v_c\right) \tag{3.4}$$

取窗口中所有词的嵌入向量$\{\boldsymbol{e}_{c-t},\ldots,\boldsymbol{e}_{c-1},\boldsymbol{e}_c,\boldsymbol{e}_{c+1},\ldots,\boldsymbol{e}_{c+t}\}$，得到：

$$P\left(v_b\middle|v_c\right)=\prod_{i\in v_b}\frac{e^{\boldsymbol{e}_i\cdot\boldsymbol{e}_c}}{\sum_{j\in v_b}e^{\boldsymbol{e}_j\cdot\boldsymbol{e}_c}} \tag{3.5}$$

其中，$P\left(v_b\middle|v_c\right)$是先验概率，通过在随机游走得到的物品序列中统计得到。公式 3.5 右侧的结果越接近$P\left(v_b\middle|v_c\right)$，嵌入向量就越能表示物品。

其次，采用梯度下降法更新向量参数，损失函数定义如下：

$$Loss=\log P\left(v_b\middle|v_c\right)=\boldsymbol{e}_b\boldsymbol{e}_c-\log\left(\sum_{i\in\{v_b,v_c\}}e^{\boldsymbol{e}_i\boldsymbol{e}_c}\right) \tag{3.6}$$

在用户的交互行为中，用户的当前交互行为在一定程度上会影响用户之后的交互行为。因此在计算序列中物品间的相似性时，需要考虑到每个物品对其他物品的影响，这里用自注意力来衡量这个影响。考虑到这点，DWSA 将自注意力机制引入 SkipGram 训练过程，计算窗口中每个词的注意力值，结合该值计算条件概率$P\left(v_b\middle|v_c\right)$，从而训练出嵌入向量，过程模型图如 3.4 所示。

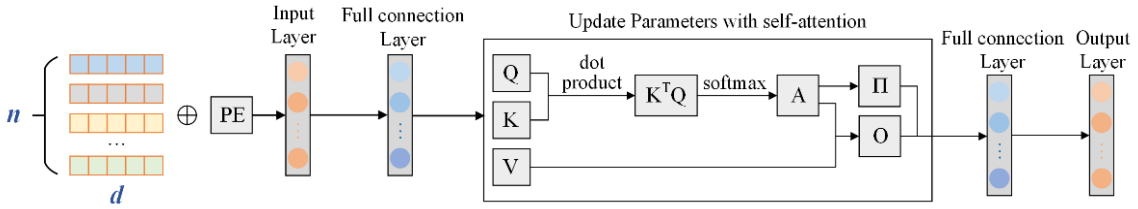


图 3.4 融合自注意力机制的嵌入向量训练过程

Fig. 3.4 SkipGram with self-attention

在图 G 上通过随机游走得到的物品序列中，相邻物品之间存在着很强的相似性，若序列顺序被打乱，则会丢失这种相似性，从而影响 SkipGram 的训练。为了防止这一点，我们采用如下策略：首先，在嵌入向量中加入位置编码，通过公式 3.7 计算出嵌入向量$\boldsymbol{e}_i\in\{\boldsymbol{e}_{c-t},\ldots,\boldsymbol{e}_{c-1},\boldsymbol{e}_c,\boldsymbol{e}_{c+1},\ldots,\boldsymbol{e}_{c+t}\}$中每一位的位置编码$PE_{pos}$。

$$PE_{pos}=\begin{cases}cos\left(pos/10000^{j/d}\right), & j=2i\\ sin\left(pos/10000^{j/d}\right), & j=2i+1\end{cases} \tag{3.7}$$

其中，pos 表示输入的位置，d 表示嵌入向量长度，i 表示维度。将PE_{pos}与向量中相应位置的值相加后得到新的嵌入向量。

其次，计算出每个词在窗口序列中的重要性。对窗口中的嵌入向量$\boldsymbol{e}_i\in\{\boldsymbol{e}_{c-t},\ldots,\boldsymbol{e}_c,\ldots,\boldsymbol{e}_{c+t}\}$初始化查询向量$\boldsymbol{q}_i$、关键词向量$\boldsymbol{k}_i$和值向量$\boldsymbol{v}_i$，根据这些向量计算出$\boldsymbol{e}_i$对窗口中其他向量的自注意力权重$Att_{ij}$为：

$$Att_{ij} = softmax\left(s\left(\boldsymbol{q}_i, \boldsymbol{k}_j\right)/\sqrt{d_k}\right) \tag{3.8}$$

其中，$\boldsymbol{k}_j \in \{\boldsymbol{k}_{c-t},\ldots,\boldsymbol{k}_c,\ldots,\boldsymbol{k}_{c+t}\}$，$s$ 为用点积来计算相似性，因此 $s(\boldsymbol{q}_i, \boldsymbol{k}_j) = \boldsymbol{q}_i \cdot \boldsymbol{k}_j$。并用 $\sqrt{d_k}$ 对 $s(\boldsymbol{q}_i, \boldsymbol{k}_j)$ 的结果进行收缩，缩小其值的取值范围。Att_{ij} 表示通过 *softmax* 函数计算出的 $\boldsymbol{e}_i$ 对 $\boldsymbol{e}_j$ 的注意力权重。

从而得到物品 $v_i\left(\in\{v_b, v_c\}\right)$ 在窗口序列中的注意力值 $W_{SA}(i)$ 为：

$$W_{SA}(i) = \sum_{i\in\{v_b,v_c\}} \boldsymbol{v}_i \cdot Att_i \tag{3.9}$$

由于注意力值在物品序列中代表一个物品相对于其他物品的重要性，所以最后结合每个物品的注意力值计算 $P\left(v_b \middle| v_c\right)$：

$$P\left(v_b \middle| v_c\right) = \sum_{i\in\{v_b,v_c\}} W_{SA}(i) \cdot \prod_{j\in\{v_b,v_c\}} Att_j \tag{3.10}$$

得到新的目标函数 L' 如下所示：

$$L' = -\min \sum_{i\in\{v_b,v_c\}} \boldsymbol{v}_i \cdot \left(\log P\left(\boldsymbol{k}_b \middle| \boldsymbol{q}_i\right)/\sqrt{d_k}\right) \cdot \log\left(\sum_{j\in\{v_b,v_c\}} \log P\left(\boldsymbol{k}_b \middle| \boldsymbol{q}_j\right)/\sqrt{d_k}\right) \tag{3.11}$$

通过对上式求导，得到损失函数 $Loss'$ 为：

$$Loss' = \log P\left(\boldsymbol{k}_b \middle| \boldsymbol{q}_c\right)/\sqrt{d_k} \cdot \log\left(\sum_{j\in\{v_b,v_c\}} \log P\left(\boldsymbol{k}_b \middle| \boldsymbol{q}_j\right)/\sqrt{d_k}\right) \tag{3.12}$$

算法如下：

算法 3.3 融合自注意力的 SkipGram 算法—— *SkipGram With SA*$\left(M, S_{v_i}, w\right)$

输入：嵌入矩阵 M，节点序列 S_{v_i}，窗口大小 w。

输出：更新后的嵌入向量矩阵 M。

1. 初始化矩阵 W_k, W_q, W_v
2. For each $v_c \in S_{v_i}$ do：
3. $\quad K, Q, V = \left(W_k, W_q, W_v\right) \cdot M$
4. $\quad Att = softmax\left(QK^{\mathrm{T}}/\sqrt{d_k}\right)$
5. $\quad W_{SA} = Att \cdot V$
6. $\quad$ For each $v_b \in S_{v_i}[c-t:c+t]$ do：
7. $\quad\quad$ // 更新嵌入矩阵 M
8. $\quad\quad J(M) = -\log \sum_{v_b\in S_{v_i}[c-t:c+t]} W_{SA}(v_b) P\left(v_b \middle| M(v_c)\right)$
9. $\quad\quad M = M - \alpha * \partial J/\partial M$
10. $\quad$ End For
11. End For
12. 返回 M

3.3 生成 Top-k 推荐列表

由于 Item-CF 无法挖掘出用户兴趣和物品之间的隐藏联系，而 DWSA 能够很好地解决这两个问题。因此，本章构建了融合自注意力机制与深度游走的协同过滤推荐算法（DWSA-CF）。

首先利用 DWSA 训练出所有物品的嵌入向量 M，通过计算嵌入向量间的余弦值来表示物品间相似性，见公式 3.13。

$$sim_{ij} = \frac{\boldsymbol{e}_i \cdot \boldsymbol{e}_j}{\|\boldsymbol{e}_i\| \cdot \|\boldsymbol{e}_j\|} \tag{3.13}$$

用 $p_{(u,i)}$ 表示用户 u 对物品 i 的兴趣，得到每个用户对所有物品的兴趣度，公式如下。

$$p_{(u,i)} = \sum_{j \in I(u) \cap S(i,k)} sim_{ij} r_{uj} \tag{3.14}$$

其中，$I(u)$ 表示用户喜欢的物品的集合，$S(i,k)$ 表示与物品 i 最相似的 k 个物品的集合，sim_{ij} 表示物品 i 和 j 的相似度，根据公式 3.13 计算得到，r_{uj} 表示用户 u 对物品 j 的兴趣，一般置为 1。

最后将 $p_{(u,i)}$ 中排名最高的前 k 个物品作为推荐列表给用户。

3.4 实验及分析

3.4.1 实验数据集及评价指标

为了评估 DWSA-CF 算法的推荐性能，本章在 Movielens-1M、Movielens-100k 和 Lastfm 三个公开数据集上进行实验。数据集信息表见表 3.1。按照 80%：20%的比例随机生成训练集。

表 3.1 数据集信息表

Tab.3.1 Datasets information

	用户数	物品数	评分数	数据稀疏率
Movielens-100k	943	1682	100000	0.9370
Movielens-1M	6040	3952	1000209	0.9581
Lastfm	1892	17632	186479	0.9944

其中，数据稀疏率表示用户评分矩阵的数据缺失率，根据公式 3.15 计算得到。稀疏率过高会大大影响推荐算法的推荐性能，而表 3.1 中数据集的稀疏率超过百分之九十，因此通过实验评价 DWSA-CF 算法在数据稀疏率较高时的推荐性能。

$$\text{Sparsity} = \frac{T(u)}{|U| * |V|} \tag{3.15}$$

评价指标采用 F1 值和覆盖率。

3.4.2 参数设置

实验中最重要的参数是嵌入向量的长度 d，不同的数据集最优的嵌入向量长度不同。通过调整 d 的大小，计算 DWSA-CF 在三个数据集上的 F1 值，如表 3.2 所示。

表 3.2 不同长度嵌入向量下的 F1 值对比

Tab.3.2 Comparison of F1 values under embedding vectors of different lengths

嵌入向量长度	Movielens-100k	Movielens-1M	Lastfm
16	0.2543	0.1058	0.1730
24	0.2790	0.1091	0.1705
32	0.2744	0.1116	**0.1735**
40	**0.2973**	0.1112	0.1706
48	0.2876	0.1034	0.1724
56	0.2855	0.1138	0.1706
64	0.2485	**0.1243**	0.1706
72	0.2559	0.0953	0.1710
80	0.2687	0.0847	0.1705

从上表可以看出，对于 Movielens-100k 数据集，嵌入向量长度为 40 时 F1 值最大，因此取 $d_1 = 40$；Movielens-1M 数据集的数据量更大，且嵌入向量长度为 64 时 F1 值最大，因此取 $d_2 = 64$。同理，Lastfm 数据集的嵌入向量长度 $d_3 = 32$。

算法中还需要计算调优的参数有随机游走的序列队长度 l、随机游走的次数 η 以及滑动窗口大小 w。通过多次实验，得出最优的参数表见表 3.3，后续实验均采用这些参数。

表 3.3 算法最优参数表

Table 3.3 Algorithm optimal parameter table

算法参数	序列队长度 l	游走次数 η	窗口大小 w
最优值	70	80	7

3.4.3 基线方法

本章提出的 DWSA-CF 结合了 Item-CF、DeepWalk 和 Self-Attention 算法，因此选取以下算法作为基线算法：

Item-CF：根据用户的历史交互行为得到物品相似性矩阵，通过该矩阵得到与用户交互过的物品最相似的物品，将它们推荐给该用户。

DW-CF：采用 DeepWalk 学习出物品的嵌入向量，然后结合协同过滤算法进行推荐。

基于自注意力机制的序列推荐算法（AttRec）：采用自注意力机制学习用户对每个物品的兴趣，并考虑了用户的长短期意图。

3.4.4 实验结果分析

我们设 k 分别为 10、15 和 20。实验结果如图 3.5-图 3.10 和表 3.4 所示，分析如下：

（1）从表 3.1 可以看出，实验数据集的稀疏性较高，因此实验结果表明，DWSA-CF 在稀疏数据集上的整体推荐性能高于基线方法。

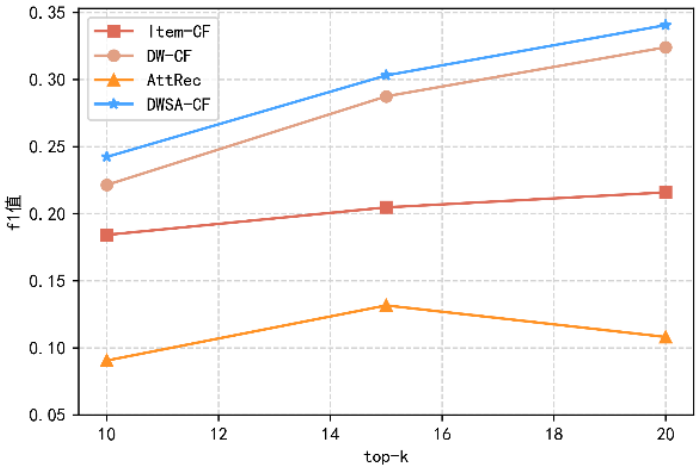


图 3.5 Movielens-100k 数据集上 F1 值对比

Fig. 3.5 Comparison of F1 value in Movielens-100k

（2）使用互信息和自注意机制可以提高推荐性能。它们可以使物品之间的相似度计算更加准确，从而获得更好的推荐性能。我们可以看到使用 F1 值当评价指标时，DWSA-CF 算法的优越性，基线方法与 DWSA-CF 算法的具体值见表 3.4。从图 3.5-3.10 可以看出，随着推荐物品数量的增加，DWSA-CF 的 F1 值增大，而 Item-CF 和 AttRec 的 F1 值减少。

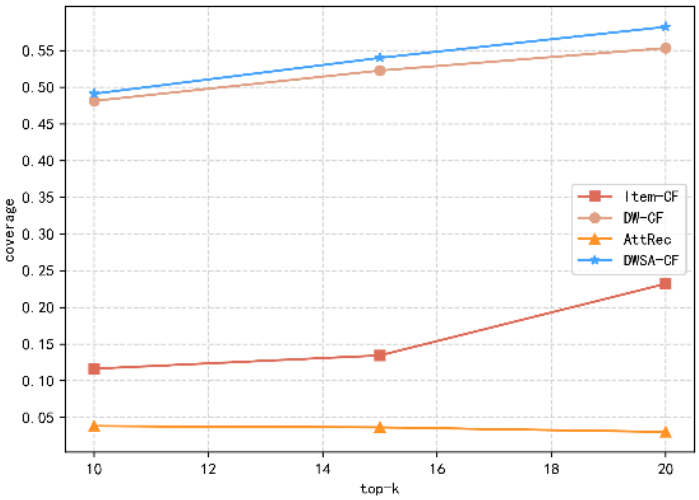


图 3.6 Movielens-100k 数据集上 Coverage 值对比

Fig. 3.6 Comparison of Converage in Movielens-100k

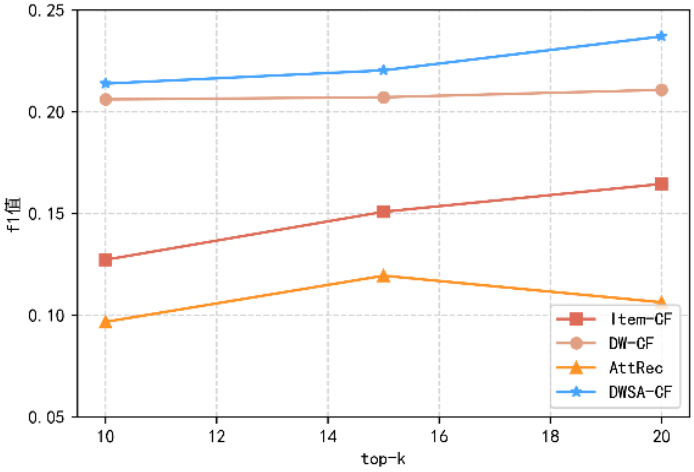


图 3.7 Movielens-1M 数据集上 F1 值对比

Fig. 3.7 Comparison of F1 value in Movielens-1M

（3）采用 F1 值作为评价指标时，在 Moivelens-100k 数据集中，DWSA-CF 算法在整体上比基线方法提高了 6.38%；在 Moivelens-1M 数据集中，DWSA-CF 算法在整体上比基线方法提高了 7.60%；在 Lastfm 数据集中，DWSA-CF 算法在整体上比基线方法提高了 11.73%。这说明，本章提出的 DWSA-CF 能够有效地提高推荐的准确率和精确率。

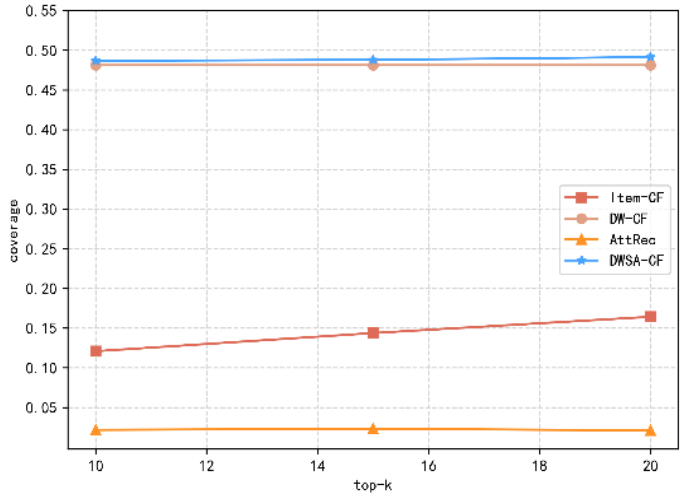


图 3.8 Movielens-1M 数据集上 Coverage 值对比

Fig. 3.8 Comparison of Coverage in Movielens-1M

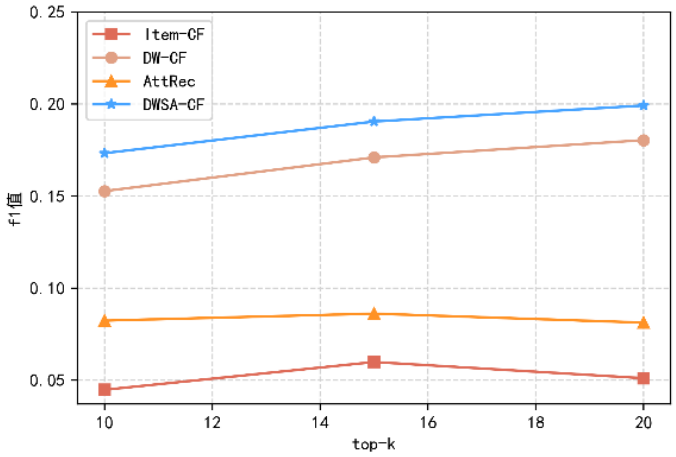


图 3.9 Lastfm 数据集上 F1 值对比

Fig. 3.9 Comparison of F1value in Lastfm

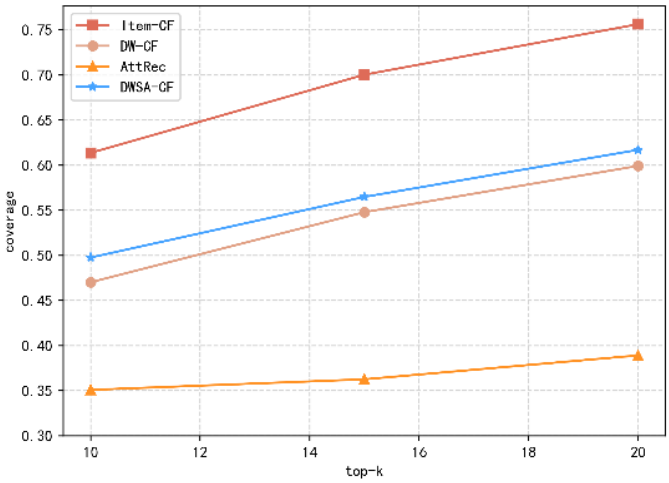


图 3.10 Lastfm 数据集上 Coverage 值对比

Fig. 3.10 Comparison of Coverage in Lastfm

（4）采用覆盖率作为评价指标时，在 Moivelens-100k 数据集中，DWSA-CF 算法在整体上比基线方法提高了 3.62%；在 Moivelens-1M 数据集中，DWSA-CF 算法在整体上比基线方法提高了 1.54%；在 Lastfm 数据集中，DWSA-CF 算法在整体上比基线方法提高了 3.84%。这说明，本章提出来的 DWSA-CF 算法对解决冷启动问题有积极作用。覆盖率值变大，则更有几率为用户推荐不那么受欢迎的物品。

（5）从表 3.4 可以看出，DWSA-CF 算法评价指标平均值远高于 AttRec，表明单独使用自注意机制无法很好地捕捉用户的兴趣。但结合 DeepWalk 并考虑物品之间的关系可以更准确地捕捉用户的兴趣，从而提高推荐的性能。

表 3.4 DWSA-CF 评价指标平均值对比

Tab.3.4 Comparison of average indicator value with DWSA-CF

Dataset	Indicators	Item-CF	DW-CF	AttRec	DWSA-CF	Improv.
Movielens-100k	F1 value	0.2016	0.2776	0.1101	**0.2953**	6.38%
	Cov.	0.1611	0.5192	0.0351	**0.5380**	3.62%
Movielens-1M	F1 value	0.1476	0.2080	0.1076	**0.2238**	7.60%
	Cov.	0.1431	0.4815	0.0221	**0.4889**	1.54%
Lastfm	F1 value	0.0518	0.1680	0.0832	**0.1877**	11.73%
	Cov.	0.6901	0.5392	0.3655	0.5599	3.84%

3.4.5 网络消融实验

为了验证本章分别加入的深度游走和自注意力机制算法的有效性，我们在 Moivelens-1M 数据集上进行了消融实验。评价指标为精准率和召回率。实验结果如表 3.5 和图 3.11--图 3.12 所示。

DWMI-CF：融合互信息的深度游走协同过滤算法。

表 3.5 推荐实验结果对比表

Table 3.5 Comparison of recommended experimental results

模型	k=10		k=15		k=20	
	Precision	Recall	Precision	Recall	Precision	Recall
DW-CF	0.0806	0.0817	0.0812	0.0934	0.0828	0.1217
SA-CF	0.0657	0.0646	0.0763	0.0848	0.1208	0.1251
DWMI-CF	0.3104	0.2030	0.2434	0.2024	0.2007	0.2019
DWSA-CF	**0.3628**	**0.3357**	**0.3274**	**0.4036**	**0.2936**	**0.4318**

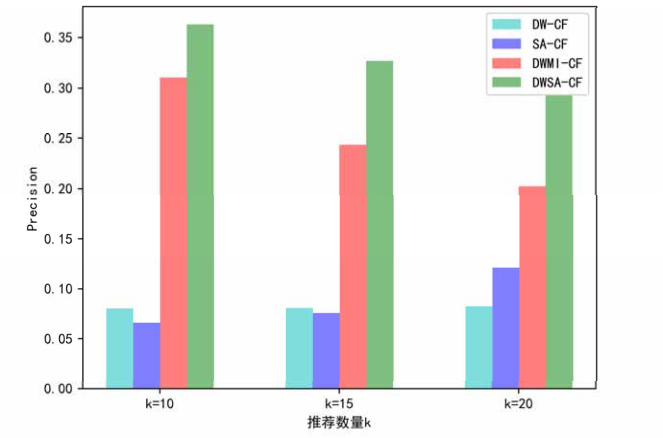


图 3.11 消融实验 Precision 对比图

Fig 3.11 Comparison of Precision

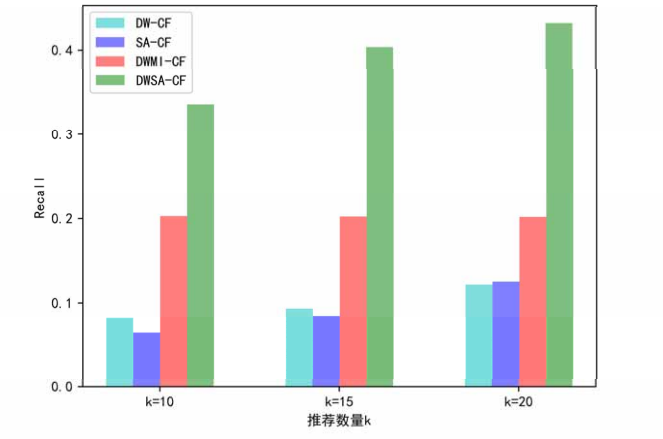


图 3.12 消融实验 Recall 对比图

Fig 3.12 Comparison of Recall

从表 3.5 和图 3.11 我们可以看出，当以准确率为评价指标时，DWMI-CF 算法比 DW-CF 和 SA-CF 算法在不同的推荐数量（Top-k）上表现都更好。这说明，将互信息作于物品关系图的有向边权重，能生成更准确的物品序列。而本章提出的 DWSA-CF 算法准确率高于 DWMI-CF 算法，这说明，用自注意力机制区别对待物品序列中的物品，能更好地衡量物品之间的相似度，生成更精确的嵌入向量，使得推荐结果更精确。从图 3.12 可以看出，以召回率为评价指标时，DWMI-CF 算法的推荐表现优于 DW-CF 和 SA-CF 算法，这说明采用互信息计算物品之间的相似度的准确性优于传统的相似度计算方法。DWSA-CF 算法远优于 DWMI-CF 算法，这说明，自注意力机制能更好地捕捉到物品之间的隐藏联系，提高推荐性能。

3.5 本章小结

本章提出了一种基于图嵌入技术的推荐算法（DWSA-CF 算法），其主要思想是采用深度游走和自注意力机制算法来学习物品的嵌入向量，以提高物品嵌入向量的准确性，然后结合基于物品的协同过滤算法进行推荐。该算法通过用户和物品的历史交互记录构造带权有向关系图，用来挖掘物品之间的隐藏联系，还能结合自注意力机制来挖掘用户的兴趣偏好。实验结果表明，该算法能较准确地计算物品之间的相似度，对稀疏度较高的数据有较好的表现。

第4章　基于分类的 GraphSage 和 LSTM 推荐算法

图嵌入技术不能很好地解决物品冷启动问题，并且对于动态变化的物品关系图，需要对节点的嵌入向量多次计算，导致计算量增加。因此，本章在 GraphSage 算法的基础上，提出了一种基于分类的 GraphSage 和 LSTM 推荐算法，描述了算法的设计思路和实现过程。最后，对算法的性能进行了仿真实验，并对实验结果进行了分析。

4.1 相关算法

图卷积网络[53]（Graph Convolutional Network, GCN）是一种用来学习图中节点嵌入向量的方法，它的核心思想是：在每次迭代时，让每个节点都能包含其邻居的信息，随着这些信息的迭代，每个节点的嵌入向量都将包含图的远端节点的信息。然而 GCN 是一种直推式学习的方法，只能在一个确定的图中去学习节点的嵌入向量，无法直接泛化得到未出现节点的嵌入向量。为了解决该问题，文献[56]提出了 GraphSage（Graph Sample and Aggregate）算法，它是一种利用节点的属性信息高效获取未知节点嵌入向量的归纳式学习框架。其核心思想是利用聚合目标节点的邻居节点信息来学习目标节点的嵌入向量。本章对 GraphSage 中节点的采样方法进行改进，在采样前对节点进行分类，使得采样时的覆盖面尽可能广，加快采样的速度。

循环神经网络[57]（Recurrent Neural Network，RNN）是一种能够对过去信息进行记忆的神经网络，广泛应用于个性化推荐中并取得了良好的效果。虽然 RNN 能利用历史信息，但是无法很好地处理用户与物品两次交互之间的时间间隔，而这些时间间隔对于捕捉用户的兴趣变化非常重要。例如，用户在半个小时内购买的一些物品很可能是相似的，这对实现用户的实时推荐具有重要的作用。针对这个问题，本章结合时间序列和 LSTM 算法，利用用户与物品交互的时间信息，捕捉用户的长期和短期兴趣，从而提高推荐性能。并结合我们对 GraphSage 的改进，提高推荐算法的整体效率。

4.1.1　GraphSage

GraphSage 算法的核心是将在图上进行的整体节点采样优化到对目标节点的局部邻居节点进行采样。首先定义图$G(V, E)$是一个无向的物品关系图，其中V为节点集，V中元素可以代表任意物品，例如商品、电影等。E为边集，E中的边表示两个物品之间的关系，这里代表两个物品与同一个用户产生过交互。

GraphSage 算法的流程如下：

（1）在图G中选取目标节点$v_i \in V$，对v_i的邻居节点进行采样。为了加快计算效率，为v_i采样一定数量的邻居节点，作为聚合过程中v_i的信息来源。设采样的邻居层数为K，

每层节点定义为当前节点的一阶邻居。设在每层邻居中采样 k 个节点，当邻居数小于 k 时，采用有放回的抽样方法选取邻居，直到采样出 k 个节点。当邻居数大于 k 时，采用无放回的抽样方法，采样出 k 个节点。

（2）根据任务目标的不同，选取合适的聚合函数，用于聚合采样得到的邻居节点蕴含的信息，得到每个节点的聚合表示。

（3）利用步骤（2）中得到的节点 v_i 的嵌入表示。返回步骤（1），选取图中其他节点作为目标节点。

（4）重复步骤（1）～（3），直到以 V 中所有节点为目标节点，得到它们的嵌入向量表示，以供下游任务使用。在算法实现过程中，以上步骤通过采用向量矩阵进行并行运算来实现。

此外，在 GraphSage 的基础上还拓展出很多算法，在特定任务中有着良好表现。例如，E-GraphSage 算法将 GraphSage 应用于边分类任务。PinSage 算法结合随机游走和 GraphSage，通过引入边权重，减小了嵌入向量计算量，在商业推荐系统中取得了良好的效果。

4.1.2 长短期记忆网络 LSTM

LSTM 是 RNN 的一种衍生算法，二者都能对带有时间信息的序列进行处理，且具有一定的记忆能力[58]。RNN 对序列中蕴含的所有信息进行记忆，而 LSTM 则有选择性地记忆序列中的信息，这种选择性的记忆通过引入门控机制来实现。

门控机制是 LSTM 的特有机制，能够控制序列中节点特征信息的清除和保留，学习到长序列中节点之间的依赖关系，从而使得 LSTM 具有选择性的记忆功能。为实现上述功能，LSTM 组合了遗忘门、输入门、输出门和一个单元状态，如图 4.1 所示。

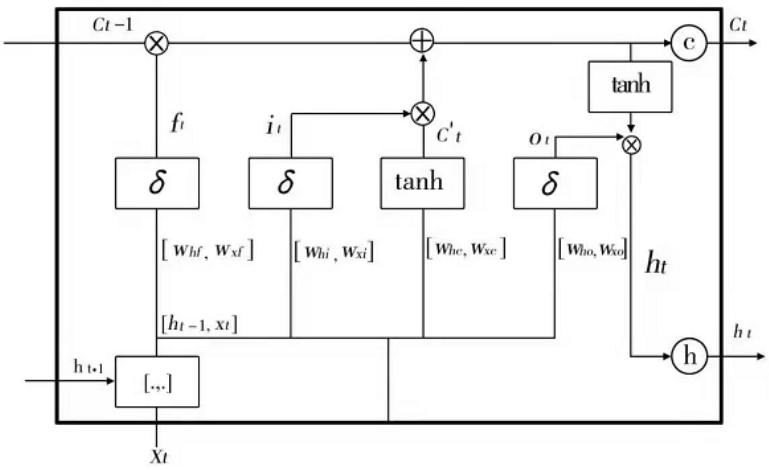


图 4.1 LSTM 示意图

Fig. 4.1 Diagram of LSTM

（1）遗忘门

遗忘门的主要作用为筛选信息，它接受一个长期记忆 C_{t-1}，从 C_{t-1} 中选择需要保留的信息，并将不需要的信息丢弃，如图 4.1 所示。计算过程如公式 4.1 所示。其中，σ 为激活函数，W_f 为权重参数，h_{t-1} 为上一时刻的隐藏状态信息，x_t 为当前时刻的输入，b_f 为偏置。我们称 f_t 为遗忘因子，其值域为 $[0, 1]$。当 f_t 的值趋近于 0 时，遗忘门选择丢弃部分信息；当 f_t 的值趋近于 1 时，遗忘门选择记忆该部分信息。最后输出经过选择了的长期记忆 i_t，计算过程如公式 4.2 所示。

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1},\, x_t\right] + b_f\right) \tag{4.1}$$

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1},\, x_t\right] + b_i\right) \tag{4.2}$$

（2）输入门

输入门的主要作用为选择记忆现在的新信息。输入门接受遗忘门输出的长期记忆 i_t 以及短期记忆 $\tilde{C}_t$，并将二者结合起来用于更新信息，如公式 4.4 所示。对于长期记忆 i_t，因为 i_t 的值域为 $[0, 1]$，输入门将 1 记为关键信息，将 0 记为非关键信息。当 i_t 的值接近 1 时，输入门更新信息，得到 C_t；否则不进行信息的更新。对于短期记忆 $\tilde{C}_t$，输入门采用 tanh 函数将 $\tilde{C}_t$ 的值域映射为 $[-1, 1]$，如公式 4.3 所示。当候选信息的 $\tilde{C}_t$ 值接近 1 时，将其保留为候选信息。最后将 i_t 和 $\tilde{C}_t$ 相乘，得到当前记忆的信息，与遗忘门的输出相加之后作为输出门的输入。

$$\tilde{C}_t = \tanh\left(W_c \cdot \left[h_{t-1},\, x_t\right] + b_c\right) \tag{4.3}$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{4.4}$$

（3）输出门

输出门接收输入门的输出，从中选择需要输出的信息，如图 4.1 所示。输入门首先通过 sigmoid 函数确定需要输出的状态，如公式 4.5 所示。再用 tanh 函数对输入门的输出进行处理。最后将前两步的结果相乘，得到 LSTM 最终的输出，如公式 4.6 所示。

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \tag{4.5}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{4.6}$$

上述内容是最基础的 LSTM 的结构。此外，针对不同的任务，LSTM 还存在许多变体。例如，在自然语言处理中，通常会使用双向 LSTM（BiLSTM）对语言序列的特征

进行提取。还有没有遗忘门、没有输入门、没有输出门、以及遗忘门和输入门结合等 LSTM 变体。

4.2 CLS_GraphSage+LSTM 算法

本章融合 GraphSage 和 LSTM 算法构建了 CLS_GraphSage+LSTM 算法，框架图如图 4.2 所示。

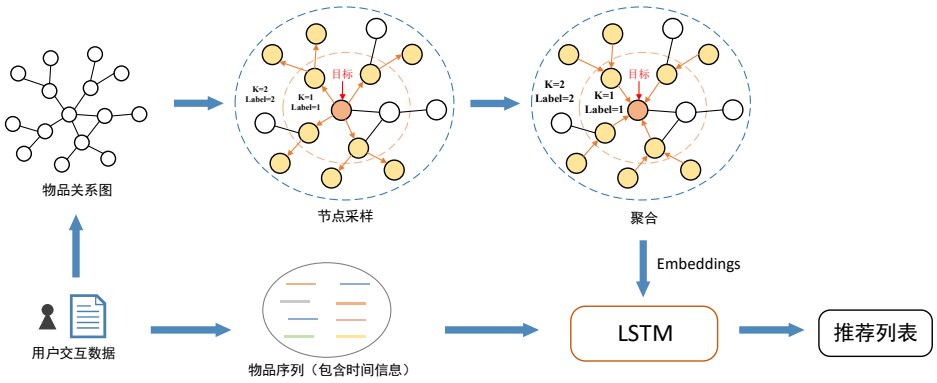


图 4.2 CLS-GraphSage+LSTM 算法框架图

Fig. 4.2 Framework of CLS-GraphSage+LSTM

该算法首先从数据集中抽取出用户的交互记录，将记录中用户交互过的物品作为节点，构造物品关系图；其次，选取数据集中物品的一个属性，根据该属性对图中各节点进行初步分类。然后，选取目标节点进行节点的采样与聚合，得到目标节点的表示方式 f，通过 f 计算出目标节点的嵌入向量；最后，从数据集中得到用户交互的物品序列，并结合各节点的表示方式 f 和 LSTM 算法捕捉用户兴趣，实现 Top-k 推荐。

4.2.1 分类采样流程

本节介绍基于分类的节点分类与聚合算法的步骤：（1）构建物品关系图；（2）节点分类；（3）采样邻居节点；（4）聚合。

设用户集合为 $U=\{u_1, u_2, \ldots, u_m\}$，物品集合为 $I=\{i_1, i_2, \ldots, i_n\}$，$c$ 为物品的一个属性，该特征的类别集合为 $\{c_1, c_2, \ldots, c_m\}$。采样步骤如下：

（1）构建物品关系图：根据用户交互记录构建物品关系图 $G(V,E)$。其中，$V=I$，E 中边的定义如下：如果用户 u_i 与物品 v_i 和 v_j 有过交互，那么 v_i 和 v_j 之间存在一条无向边 $e(v_i, v_j): v_i - v_j$。

（2）节点分类：根据物品的属性 c 的值对物品节点进行分类。令属性 c 的值相同的物品为同一类物品，否则不是同一类物品。

（3）采样邻居节点：在采样之前，对物品关系图进行微调，首先选取目标节点 v_c，

将与v_c同类且相连的节点作为一阶邻居。与一阶邻居相连且属于同一类的节点作为二阶邻居，以此类推，直到找出v_c的每一阶邻居，如图 4.3 所示。

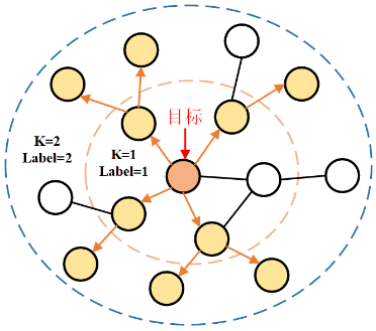


图 4.3 分类采样图

Fig. 4.3 Diagram of classifing and sampling

接下来采样v_c的K阶邻居节点，为聚合过程做准备。一般来说，离v_c越近的邻居与v_c的关系越大，与v_c同类的节点与v_c的关系更大。为了尽量减少聚合过程中计算的复杂程度，既需要采样的覆盖面尽可能大，又不能采样过多节点。因此在采样过程中，每一次采样不会采样当前节点的所有邻居，而是随机选取不超过k个邻居节点。

采样方法：在v_c所属类中随机采样k个一阶邻居，记为集合$neighs\ 1=\{v_i,\ldots,v_j\}$。随后，分别在$neighs\ 1$中各节点所属类中随机采样$k$个一阶邻居作为$v_c$的二阶邻居，记为集合$neighs\ 2=\{v_g,\ldots,v_h\}$。重复以上步骤，直至选取完$v_c$的$K$阶邻居，如图 4.4 所示。

上述采样过程融合了分类和随机采样两种方法，存在如下两种特殊采样情况：

a：如果某类节点中存在节点v_s，使得从v_c到v_s需要经过不止一条边，那么添加边$v_c \rightarrow v_s$，并删除边$v_{s-1} \rightarrow v_s$，使得v_s成为v_c的一阶邻居，如图 4.4 所示。这样处理的目的是在聚合过程中，能够将与v_c相似的节点v_s的信息迅速聚合起来。

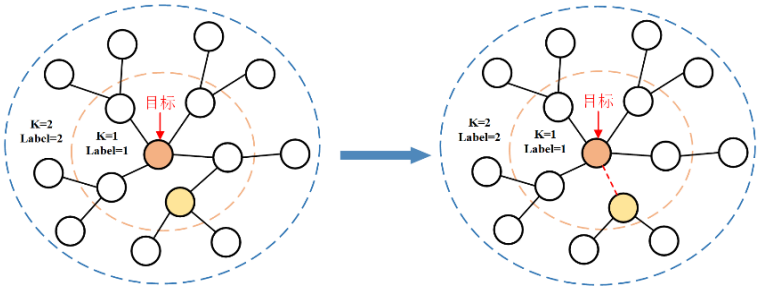


图 4.4 第一类特殊采样情况

Fig. 4.4 Special sampling of class 1

b：如果当前目标节点v_c所属类中与其相连的节点数量过少，那么从其他类中选取其直接相连的节点作为一阶邻居，如图 4.5 所示。此种情况下，与v_c最相似的节点数量

少，无法聚合到足够的信息，因而选取有相同交互关系但不同类的节点作为一阶邻居。

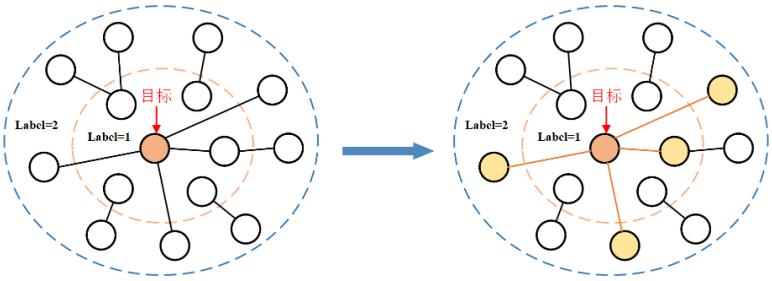


图 4.5 第二类特殊采样情况

Fig. 4.5 Special sampling of class 2

（4）聚合：将选取出来的邻居节点的嵌入向量输入到聚合函数 F 中，逐层聚合，最后得到目标节点 v_c 的嵌入向量，如图 4.6 所示。聚合的计算过程见 4.3.4 节。

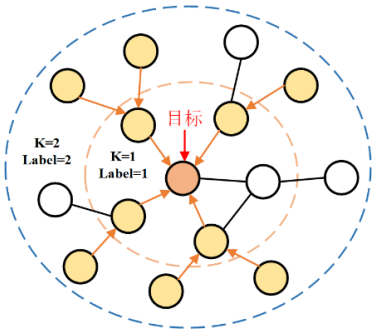


图 4.6 聚合过程图

Fig. 4.6 Process of polymerization

通过以上步骤得到每个节点的聚合函数 F，学习出 F 中的参数。根据 F 可以快速计算出图中各节点的嵌入向量，并用于自然语言处理、计算机视觉、分类、推荐等任务。

4.2.2 新节点的采样

当有新节点加入图中时，如果数据集中存在与它相关的交互记录，那么 GraphSage 根据交互记录直接将其加入物品关系图中。如果不存在与新节点的交互记录，那么 GraphSage 无法直接将其加入图中，这会使得采样时无法覆盖到该节点，导致聚合过程丢失新节点的信息。针对这个问题，当不存在交互记录的新节点加入时，我们从新节点的同类节点中选取度中心性最高的一些节点，将这些节点与新节点直接相连。这样能增加新节点被采样的概率，聚合新节点的信息。从而学习到新节点的嵌入向量表示，这对于解决推荐系统中存在的冷启动问题具有重要的作用。

度中心性是刻画节点关键性最直接且有效的局部特征方法，衡量节点在图的局部的重要性。一个节点的度越大就意味着它的度中心性越高，该节点在当前局部图中就越重要。我们从每一类节点中选取度最大的 m 个节点作为该类的重要节点。

加入新节点的步骤如下：

（1）通过计算节点的度中心性得到物品关系图中每一类的重要节点。

（2）当新节点与已有节点之间存在边时，按照 GraphSage 已有的方法，直接将其加入图中，如图 4.7 所示。

（3）当新节点与已有节点之间不存在边时，选取同类中度中心性最高的 m 个节点与其相连，如图 4.7 所示。

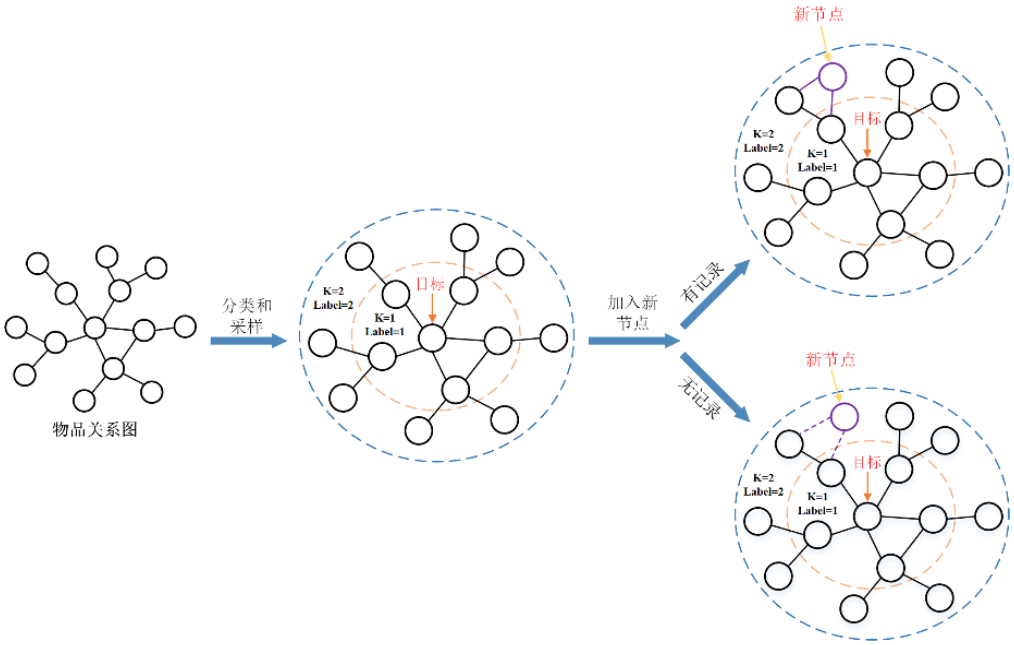


图 4.7 聚合过程图加入新节点后的采样图

Fig. 4.7 Sampling after adding new nodes

4.2.3 聚合函数

聚合函数是 GraphSage 算法实现节点信息聚合的核心。由于邻居节点的采样是随机且无序的，为了排除采样节点的不同顺序对信息聚合产生的影响，即当改变采样节点的顺序时，聚合函数的输出保持不变。因此，聚合函数必须是对称函数，且对节点的信息具有较高的表达能力。根据目标任务的不同，聚合函数有着许多类型。

设 v 为采样到的第 $k-1$ 阶邻居中的一个节点，$N(u)$ 是其在第 k 阶邻居中的所有邻居节点。h_v^k 表示 v 的信息，即嵌入向量，从 $N(u)$ 中所有节点聚合而来，σ 为激活函数。

（1）平均聚合：首先对 $N(u)$ 中所有节点的嵌入向量的每个维度取平均值；再与节点 v 的嵌入向量进行拼接；最后将拼接的结果进行非线性变换，得到 v 的嵌入向量。计算公式如下：

$$h_{N(u)}^k = mean\left(\left\{h_u^{k-1}, u \in N(u)\right\}\right) \tag{4.7}$$

$$h_v^k = \sigma\left(W^k \cdot CONCAT\left(h_v^{k-1}, h_{N(u)}^k\right)\right) \tag{4.8}$$

（2）最大聚合：与平均聚合类似，首先取 $N(u)$ 中所有节点的嵌入向量的每个维度的最大值，如式 4.9 所示。再与节点 v 的嵌入向量进行拼接。最后进行非线性变化，得到节点 v 的嵌入向量，计算过程如式 4.8 所示。

$$h_{N(u)}^k = \max\left(\left\{h_u^{k-1}, u \in N(u)\right\}\right) \tag{4.9}$$

（3）归纳式聚合：将目标节点 v 与 $N(u)$ 中所有节点的嵌入向量的每个维度取平均值，再进行非线性变换，从而得到 v 的嵌入向量。计算公式如下：

$$h_v^k = \sigma\left(W^k \cdot mean\left(\left\{h_v^{k-1}\right\} \cup \left\{h_u^{k-1}, \forall u \in N(v)\right\}\right)\right) \tag{4.10}$$

（4）Pooling 聚合：首先对 $N(u)$ 中所有节点的嵌入向量进行非线性变换；其次对非线性变换的结果进行 pooling 操作（分为 max pooling 和 mean pooling 两种方式：max pooling 是取局部接受域中最大的点；mean pooling 为取局部接受域中值的平均值）；随后将 pooling 的结果与目标节点 v 的嵌入向量进行拼接；最后进行一次非线性变换，从而得到 v 的嵌入向量。计算过程如下：

$$h_{N(v)}^k = max\left(\left\{\sigma\left(W_{pool} h_{u_i}^k + b\right)\right\}, \forall u_i \in N(v)\right) \tag{4.11}$$

$$h_v^k = \sigma\left(W^k \cdot CONCAT\left(h_v^{k-1}, h_{N(u)}^{k-1}\right)\right) \tag{4.12}$$

本章采用平均聚合和最大聚合这两种聚合函数来学习节点的嵌入向量表示。

4.2.4 算法实现

本节首先给出 CLS_GraphSage 算法的伪代码，再将其与 LSTM 算法结合，实现 Top-k 推荐。

CLS_GraphSage 算法分为三个部分，基于分类方法的结点采样是第一部分，见算法 4.2。第二部分是节点信息的聚合，见算法 4.1 的 10-19 行。第三部分是加入新节点时，物品关系图的更新算法，见算法 4.3。

算法 4.1 CLS_GraphSage 算法框架

输入：物品关系图 $G(V, E)$，新节点集合 $nodes$，特征集合 $\{\boldsymbol{x}_v, \forall v \in B\}$，采样深度 K，权重矩阵 W^k, $\forall k \in \{1, 2, \ldots, K\}$，非线性函数 σ，可微聚合函数 $AGGREGATE_k$。

输出：节点嵌入表示 $\boldsymbol{z}_v(v \in V)$。

1. **Update Item Graph**$(G(V, E))$ // 更新物品关系矩阵 $G(V, E)$
2. // 节点采样过程

3. $B^K \leftarrow B$
4. For $k = K, \ldots, 1$ do：
5. $\quad B^{k-1} \leftarrow B^k$
6. $\quad$ For $u \in B^k$ do：
7. $\quad\quad B^{k-1} \leftarrow B^{k-1} \bigcup CLS\_NeighborSampling(G, u)$
8. $\quad$ End For
9. End For
10. // 节点信息聚合过程
11. $\boldsymbol{h}_u^0 \leftarrow x_v, \forall v \in B^0$
12. For $k = 1, \ldots, K$ do：
13. $\quad$ For $u \in B^k$ do：
14. $\quad\quad \boldsymbol{h}_{N(u)}^k \leftarrow AGGREGATE_k\left(\left\{\boldsymbol{h}_{u'}^{k-1}, \forall u' \in N_k(u)\right\}\right)$
15. $\quad\quad \boldsymbol{h}_u^k \leftarrow \sigma\left(W^k \cdot CONCAT\left(\boldsymbol{h}_u^{k-1}, \boldsymbol{h}_{N(u)}^k\right)\right)$
16. $\quad\quad \boldsymbol{h}_u^k \leftarrow \boldsymbol{h}_u^k \Big/ \left\|\boldsymbol{h}_u^k\right\|_2$
17. $\quad$ End For
18. End For
19. $\boldsymbol{z}_u \leftarrow \boldsymbol{h}_u^K, \forall u \in B$
20. 返回 $\boldsymbol{z}_v$

基于分类的节点采样算法如下。

算法 4.2 CLS_NeighborSampling 算法

输入：物品关系图 $G(V, E)$，目标节点 u。

输出：采样的邻居节点集合 $neighs$。

1. 初始化 $neighs = \{\}$
2. For i in $u.neighbor$ do：
3. $\quad$ // $u.cls$ 表示节点 u 所属的类别
4. $\quad$ If exist $i.cls == u.cls$ then：
5. $\quad\quad$ // 将与目标节点类别相同的节点加入 $neighs$
6. $\quad\quad neighs.append(i)$
7. $\quad$ Else：
8. $\quad\quad$ For j in $i.neighbor$ do：
9. $\quad\quad\quad$ If $j.cls == u.cls$ then：

10. $neighs.append(j)$
11. End If
12. End For
13. End If
14. End For
15. 返回 $neighs$

当有新节点加入图中时，在进行节点采样之前，需要对物品关系图进行更新，算法伪代码如下。

算法 4.3 物品关系图更新算法

输入：物品关系图 $G(V, E)$，新节点集合 $nodes$，新节点与现有节点之间的边数 N。

输出：新的物品关系图 $G(V, E)$。

1. 初始化节点的度集合 $degrees$
2. // 计算图中每个节点的度中心性
3. For v in V do：
4. // 将节点的度加入 $degrees$
5. $degrees.append(\{v : v.degree\})$
6. End For
7. For $node$ in $nodes$ do：
8. $tmp = \{\}$
9. // 将同类节点加入集合 tmp，用于计算节点的度
10. $tmp.append(lambda\ i : for\ i\ in\ V\ if\ i.cls == node.cls)$
11. // 将 tmp 中度最大的前 N 个节点作为候选集合
12. $maxDe = degrees.value().sort(reverse = True)[0 : N-1]$
13. // 将新节点与 $maxDe$ 中的节点相连，更新 $G(V, E)$ 的边集
14. For j in $maxDe.keys()$ do：
15. $E.update(\{e_{node,j} : (node, j)\})$
16. End For
17. End For

18. $V.append(nodes)$ // 更新$G(V, E)$的顶点集

19. 返回更新后的$G(V, E)$

本节第二部分结合 CLS_GraphSage 算法与 LSTM 算法实现 Top-k 推荐。

用户在某段时间内交互过的物品可以代表用户在这段时间内的兴趣，例如某个用户在一周内观看了电影《让子弹飞》、《美人鱼》、《功夫》和《大话西游》，那么就认为该用户近期内的兴趣为观看喜剧电影，且更喜欢周星驰的喜剧电影，因而可以为其推荐周星驰的其他喜剧电影。基于上述背景，我们利用 LSTM 算法挖掘用户在各个时间段的兴趣，分别给出推荐。

下面给出 CLS_GraphSage+LSTM 的实现过程。首先利用 CLS_GraphSage 算法对物品关系图进行处理，得到所有物品的嵌入表示；其次，从数据集中抽取用户的交互记录，将这些记录转换为物品序列，并按时间段进行划分，得到训练数据集；最后，将数据集中的每个物品映射为 CLS_GraphSage 算法学习到的嵌入向量，作为 LSTM 算法的输入，从而给出每个用户在各个时间段内的推荐。

这里对上述过程进行形式化描述。设数据集为D，从中抽取出用户集合$U(|U|=m)$和物品集合$I(|I|=n)$，用户与物品的交互记录记为集合S：

$$S=\{S_{u_1}:i_a i_b \ldots i_h,\ S_{u_2}:i_c i_d \ldots i_t,\ \ldots,\ S_{u_i}:i_e i_f \ldots i_n,\ \ldots,\ S_{u_m}:i_g i_l \ldots i_k\}。$$

其中，$u_i \in U$，$i_a, i_b, \ldots, i_l \in I$，$S_{u_i}$表示用户$u_i$的交互序列。由于$D$中包含用户交互的时间序列信息，我们按时间段对$S$中的序列进行切分，得到集合$DS$：

$$DS=\{S_{u_1}:(i_a \ldots i_t,\ i_c \ldots i_h),\ \ldots,\ S_{u_i}:(i_e \ldots i_g,\ \ldots,\ i_k \ldots i_n),\ \ldots,\ S_{u_m}:(i_g \ldots i_r,\ i_s \ldots i_k)\}。$$

将DS作为 LSTM 算法的输入，得到推荐列表R：

$$R=LSTM(DS)。$$

其中，LSTM 算法在 4.1.2 节中被定义，R中包含给每个用户的推荐物品。

4.3 实验结果及分析

4.3.1 数据集介绍

首先在数据集 Cora 和 PubMed 上进行节点分类实验，用来验证改进采样方法后的 GraphSage 算法对节点分类任务的影响。其中，Cora 是一个论文数据集，包含 2708 篇论文，涵盖“基于案例”、“遗传算法”等 7 个方向，论文之间存在 5429 条引用链接。PubMed 是糖尿病研究领域的一个论文数据集，涵盖该领域的 3 个研究方向，论文之间存在 44338 条引用链接。数据集详细信息如表 4.1 所示。

表 4.1 分类实验数据集介绍

Tab.4.1 Datasets of the classification experiment

数据集	节点数	边数	特征数	类别数
Cora	2708	5429	1433	7
PubMed	19717	44338	500	3

其次在推荐实验中，我们选取包含时间序列信息的数据集，包括 3.4.1 节介绍过的 Lastfm 和 Movielens-1M 数据集，以及 IMDB 数据集。IMDB 数据集是一个电影评分数据集，包含 2113 个用户的信息、10197 部电影的信息，以及 855598 条电影评分信息。平均每个用户贡献 404 条评分，平均每部电影有 84 条评分。

4.3.2 实验结果及分析

我们在数据集 Cora 和 PubMed 上进行节点分类实验。分别采用非监督学习和监督学习方法，计算两种方法下节点的分类效果，用 F1 值作为评价指标。在非监督学习情况下，对节点进行聚类，从而把每个节点归到一个类中。而在监督学习情况下，已知节点的类数，将节点进行归类。显然，由于聚类的类数与节点的类数不一定相同，采用监督学习方法的分类效果更好。

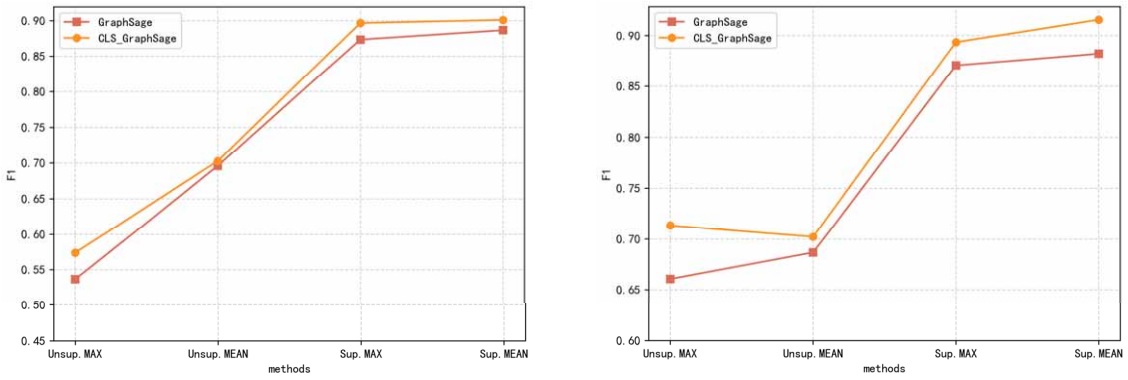
设置 GraphSage 为基线算法，CLS_GraphSAGE 为对比算法，实验中设置训练轮数 Epoch=50，节点分类的结果见表 4.2。

表 4.2 节点分类实验

Tab.4.2 The classification experiment of nodes

Methods		Cora		PubMed	
		Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
GraphSAGE	MAX	0.5365	0.8736	0.6606	0.8706
	MEAN	0.6962	0.8869	0.6898	0.8822
CLS_GraphSAGE	MAX	0.5742	0.8969	0.7136	0.8935
	MEAN	0.7029	0.9013	0.7042	0.9157

结果表明，引入分类方法优化节点采样过程之后，GraphSage 算法的节点分类性能得到了提升。将实验结果绘制成图 4.8，从图中可以看出，在有监督与无监督学习的方法下，聚合函数使用 MEAN 聚合和 MAX 聚合，我们的算法对比 GraphSage 算法均有不同程度的提升。



(a). Cora 数据集分类效果对比　　(b). PubMed 数据集分类效果对比

图 4.8 两个数据集上分类实验结果对比

Fig. 4.8 Comparison of classification experimental results on two datasets

下面进一步结合 LSTM 算法进行推荐。我们在数据集 LastFM、Movielens-1M 和 IMDB 数据集上进行推荐实验，以 LSTM 算法以及 GraphSage+LSTM 算法作为基线算法，设置每次推荐的物品数量为 20，实验结果见表 4.3。

（1）在结合了 GraphSage 算法之后，LSTM 算法在两个数据集上的推荐性能有了一定程度的提升，其中召回率提升约 5%，精准率提升约 10%。这表明，GraphSage 算法能够更精确的学习到物品的嵌入表示，有利于 LSTM 算法对用户兴趣的挖掘。

（2）表 4.3 的最后两行数据是基于分类的 GrapSage 与 LSTM 算法结合实现推荐的实验结果。显然，对比另外两种算法，其召回率和精准率更高。由此可知，在三个实验数据集上，采用分类的算法进行节点采样能提高信息聚合的速度，得到了更好的嵌入表示，使得 LSTM 得到了更精确的嵌入向量，进一步提升了推荐性能。

（3）我们在 GraphSage 中设置了 MAX 和 MEAN 两种聚合函数，从实验结果可以看出，在使用 MEAN 聚合函数时，两个数据集上的推荐性能更高。因此，MEAN 更适合与 LSTM 算法组合实现 Top-k 推荐。

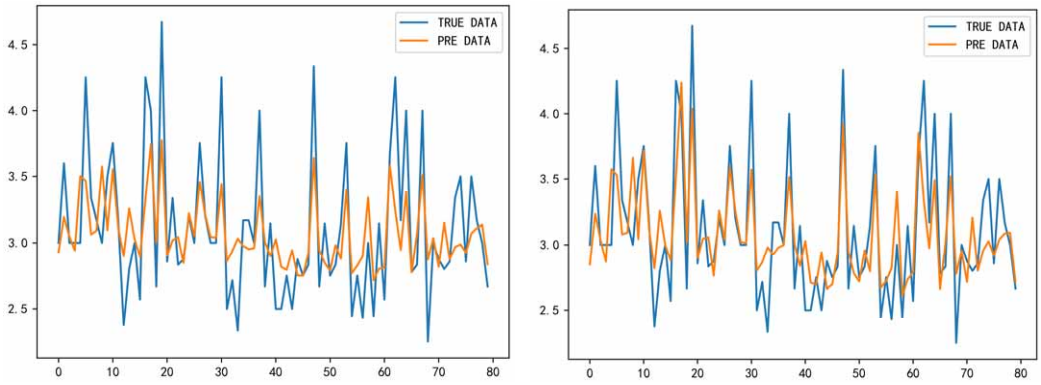
表 4.3 CLS_GraphSAGE+LSTM 实验结果表

Tab.4.3 Experiment Results of CLS_GraphSAGE+LSTM

Methods		Lastfm		Movielens-1M		IMDB	
		Rec.	Pre.	Rec.	Pre.	Rec.	Pre.
LSTM		0.2451	0.0892	0.2046	0.2148	0.2821	0.2746
GraphSage+LSTM	MAX	0.2568	0.1694	0.1930	0.2580	0.2513	0.2289
	MEAN	0.2976	0.1922	0.2398	0.2566	0.2637	0.2481
CLS_GraphSage+LSTM	MAX	0.2742	0.1882	0.3671	0.4017	0.3816	0.3927
	MEAN	0.3759	0.2352	0.3715	0.4829	0.4497	0.4986

为了更清晰地显示算法的对比效果，我们从数据集中随机选取百分之一的数据绘制

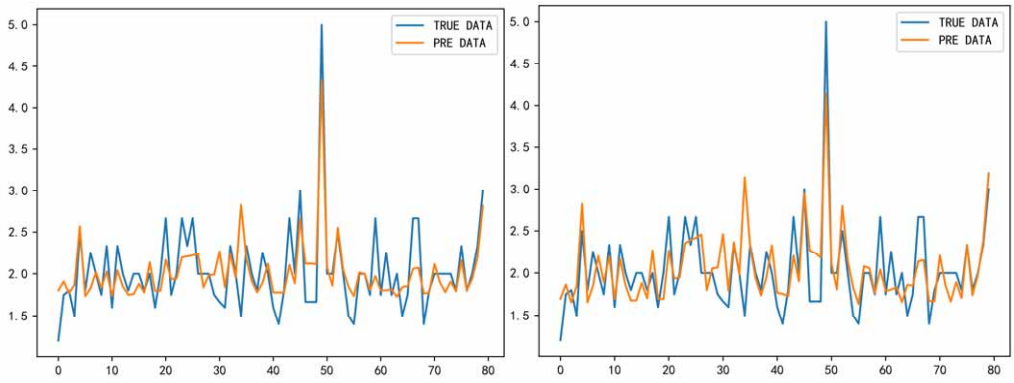
出图 4.9-图 4.11，用来展示可视化推荐结果。



(a). LSTM (b). CLS_GraphSage+LSTM

图 4.9 Lastfm 数据集上模型的拟合曲线

Fig. 4.9 Fitting curve of Lastfm

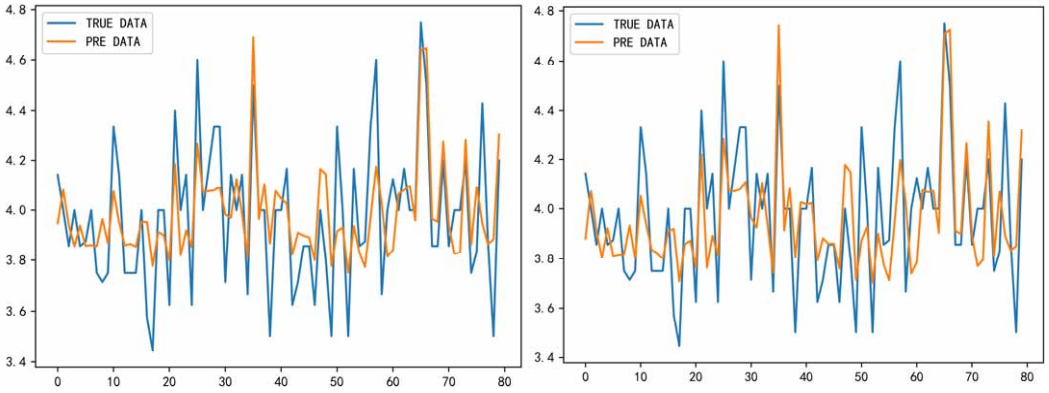


(a). LSTM (b). CLS_GraphSage+LSTM

图 4.10 Movielens-1M 数据集上模型的拟合曲线

Fig. 4.10 Fitting curve of Movielens-1M

图中蓝色曲线代表数据集中用户对物品的打分数据，橙色曲线代表算法得到的预测值，两条曲线的拟合程度反应了算法的预测效果。



(a). LSTM (b). CLS_GraphSage+LSTM

图 4.11 IMDB 数据集上模型的拟合曲线

Fig. 4.11 Fitting curve of IMDB

（1）三个数据集均为用户观看电影的数据，包含用户观影的时间信息，我们将用户的观影记录按时间段划分，采用 LSTM 算法挖掘每个时间段内用户的观影兴趣，再结合 GraphSage 算法进行推荐。从图 4.9-图 4.11 中可以看出，CLS_GraphSage+LSTM 算法比原始的 LSTM 算法拟合效果更好。

（2）对于数据集中大部分的数据，两种算法的拟合效果都较好，而对于数据集中的离群数据，CLS_GraphSage+LSTM 算法能够更好地学习到这些离群点的信息，在推荐时能够覆盖到这些节点，曲线拟合效果也更好，这说明能较好地解决物品冷启动问题。

4.4 本章小节

本章设计了一种新节点的采样方法，其思想是将节点分类后与度中心性高的节点直接相连，以解决物品的冷启动问题；同时为加快计算速度，对 GraphSage 算法的采样方法进行改进，然后结合时间序列信息和 LSTM 算法，实现 Top-k 推荐。实验表明，改进后的算法，能较好地解决物品的冷启动问题，同时还能加快邻居节点的聚合速度，提高节点的表示能力。

第5章　总结与展望

5.1 总结

为了解决推荐算法中数据稀疏性问题和冷启动问题，同时为了提高推荐算法的计算效率和节约计算资源。本文首先对推荐算法的研究现状进行了介绍，然后对几种经典的推荐算法进行了详细的分析，并对其优缺点进行对比研究，最后提出了两种基于图嵌入技术的推荐算法。主要研究内容如下：

（1)研究了推荐算法的发展现状，对几种经典推荐算法的原理与优缺点进行了分析，针对推荐算法中普遍存在的问题和面临的挑战进行了归纳和研究。

（2）为了更精准的计算物品之间的相似度，本文提出了 DWSA-CF 算法。该算法首先利用用户物品历史信息和时间先后顺序，构建物品关系有向图；然后利用互信息计算物品之间的相似度作为有向边的权重，通过随机游走生成物品关系序列队；为了相似物品更相邻，将自注意力机制引入 SkipGram 模型中，使得最后得到的物品嵌入向量矩阵更准确；最后融合 Item-CF 算法，生成前 Top-k 个物品推荐给用户。对比实验表明，以 F1 值和覆盖率为评价指标，DWSA-CF 算法优于基线算法，并且对于数据稀疏率较高的数据集，DWSA-CF 算法有较高的表现，说明该算法对解决数据稀疏性问题有不错的效果。

（3）针对新物品的冷启动问题，本文首先根据物品的特征属性对其进行分类，然后将其与各类节点中度中心性高的节点直接相连，该方法较好地解决了新物品的冷启动问题。针对图动态变化时，需重复计算节点的嵌入向量这个问题，本文对 GraphSage 算法的采样方法进行改进，提出了一种基于分类的 GraphSage 和 LSTM 推荐算法。该算法不仅能提高计算效率，而且能更好地挖掘用户的兴趣。在两个公开数据集上进行分类仿真实验，实验结果表明，引入分类方法优化节点采样过程之后， GraphSage 算法的节点分类性能得到了提升。在三个公开数据集上进行推荐的仿真实验，实验结果表明，CLS_GraphSage+LSTM 算法能学习新物品节点的嵌入向量表示，并对其进行推荐，对于离群的节点，有较好的拟合效果，在精准率和召回率上都有所提高。

5.2 展望

本文所提出的推荐算法能够较好地计算物品之间的相似度，能精确计算物品的嵌入向量。但是仍然存在以下不足，可以作为今后进一步的研究内容。

（1）本文只使用了节点之间的时间信息，忽略了节点的属性信息，而节点的属性信息包含了用户和物品的详细信息，可以提高推荐的准确率，因此，在今后的研究中，可以将节点的属性信息考虑进去。

（2）本文只对同构图进行编码，而现实场景中，大多存在的都是异构图，因此，今后将考虑直接在用户物品异构图上进行嵌入计算，直接得到用户和物品的嵌入向量，提高推荐性能。

参 考 文 献

[1] 中国互联网络信息中心. 第 47 次《中国互联网络发展状况统计报告》(全文) [EB/OL]. http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/202202/t20220225_71727.html.

[2] Liu J, Choi W H, Liu J, et al. Personalized Movie Recommendation Method Based on Deep Learning[J]. Mathematical Problems in Engineering, 2021.

[3] 黄立威, 江碧涛, 吕守业, 等. 基于深度学习的推荐系统研究综述[J]. 计算机学报, 2018, 41(07): 1619-1647.

[4] Goldberg D., Nichols D., Oki B. M., et al. Using collaborative filtering to weave an information tapestry[J]. Communications of the ACM. 1992, 35(12): 61-70.

[5] Linden, G, Smith, et al. Amazon.com recommendations: item-to-item collaborative filtering[J]. Internet Computing, IEEE, 2003, 7(1):76-80.

[6] Pazzani, Michael J., Daniel B. Content-based recommendation systems[C]. In: The adaptive web. Berlin: Springer, 2007.

[7] 李雪婷, 杨抒, 赛亚热 •迪力夏提,等. 融合内容与协同过滤的混合推荐算法应用研究[J]. 计算机技术与发展, 2021, 31(10):7.

[8] 张润莲, 张瑞, 武小年,等. 基于混合相似度和差分隐私的协同过滤推荐算法[J]. 计算机应用研究, 2021, 38(8):6.

[9] De Campos, Fernandez Luna, Huete J, et al. Coming content-based and collaborative recommendations: A hybrid approach based on Bayesian networks[J]. International Journal of Approximate Reasoning, 2010, 51(7):785-799.

[10] 王粤, 黄俊, 郑小楠,等. 基于用户兴趣和评分差异的改进混合推荐算法[J]. 计算机工程与设计, 2021, 42(10):7.

[11] 田保军, 杨浒昀, 房建东. 融合信任和基于概率矩阵分解的推荐算法[J]. 计算机应用, 2019, 39(10):7.

[12] Dong Y, Liu S, Chai J. Research of hybrid collaborative filtering algorithm based on news recommendation[C]. Process of Biomedical Engineering and Informatics. IEEE, 2016: 898-902.

[13] Zheng L, Noroozi V, Yu P S. Joint deep modeling of users and items using reviews for recommendation[C]. Proceedings of the 10th ACM International Conference on Web Search and Data Mining. IEEE,2017:425-434.

[14] He X, Liao L, Zhang H, et al. Neural Collaborative Filtering[J]. International World Wide Web Conferences Steering Committee, 2017.

[15] Shan Y, Hoens T R, Jiao J, et al. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features[C]// the 22nd ACM SIGKDD International Conference. ACM, 2016.

[16] Wang R, Fu B, Fu G, et al. Deep & Cross network for ad click prediction.

[17] Zhu J, Shan Y, Mao J C, et al. Deep embedding forest: Forest-based serving with deep embedding features.

[18] Convington P, Adams J, Sargin E. Deep neural networks for youtube recommendations[C]. Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016:191-198.

[19] Li S, Kawale J, Fu Y. Deep collaborative filering via marginalized denoising auto-encoder[C]. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM, 1025:811-820.

[20] Wu C Y, Ahmed A, Beutel A, et al. Recurrent recommender networks[C]. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. 2016: 7-10.

[21] Xiao J, Ye H, He X, et al. Attentional factorization machines: Learning the weight of feature interactions via networks[C]. Proceedings of the 25th International Joint Conference on Artificial Intelligence. IEEE, 2017:3119-3125.

[22] 石宜金, 马瑞英, 贾志洋. E-Learning 系统的个性化推荐方法研究[J]. 信息与信息化, 2012(2):51-54.

[23] Liu Q, Wu S, Wang L. Multi-behavioral sequential prediction with recurrent log-bilinear model[J]. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(6): 1254-1267.

[24] Wu Y, DuBois C, Zheng A X, et al. Collaborative denoising auto-encoders for top-n recommender systems[C]. Proceedings of the 9th ACM International Conference on Web Search and Data Mining. ACM, 2016:153-162.

[25] Perozzi B, Al-Rfou R, Skiena S. DeepWalk: Online learning of social representations[C]. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014: 701-710.

[26] Grover A, Leskovec J. Node2vec: Scalable feature learning for networks[C]. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016: 855-864.

[27] Tang J, Qu M, Wang M, et al. LINE: Large-scale information network embedding[C]. Proceeding of the 24th International Conference on World Wide Web, WWW,2016:1067-1077.

[28] Wang D, Cui P, Zhu W. Structural deep network embedding[C]. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016: 1225-1234.
[29] Cao S, Lu W, Xu Q. GraRep: Learning graph representations with global structural information[C]. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM, 2015:891-900.
[30] Ribeiro L F, Saverese P H, Figueiredo D R. Sturc2vec: Learning node representations from structural identity[C]. Proceeding of the 23rd ACM SIGDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017: 386-394.
[31] Tang J, Qu M, Mei Q. PTE: Predictive text embedding through large-scale heterogeneous text networks[C]. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015: 459-467.
[32] Yang C, Liu Z, Zhao D, et al. Network representation learning with rich text information[C]. Proceeding of the 24th International Joint Conference on Artificial Intelligence, 2015: 2111-2117.
[33] Liao L, He X, Zhang H, et al. Attributed social network embedding[J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 30: 2257-2270.
[34] Barkan O, Koenigstein N. Item2vec: Neural item embedding for collaborative filtering[C]. 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2016: 1-6.
[35] Grbovic M, Radosavljevic V, Djuric N, et al. E-commerce in your Inbox: Product recommendations at scale[C]. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015: 1809-1818.
[36] Vasile F, Smirnova E, Conneau A. Meta-Prod2Vec: Product embeddings using side-information for recommendation[C]. Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016: 225-232.
[37] Wang J, Huang P, Zhao H, et al. Billion-scale commodity embedding for e-commerce recommendation in Alibaba[C]. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2018: 839-848.
[38] Chen C M, Wang C J, Tsai M F, et al. Collaborative similarity embedding for recommender systems[C]. The World Wide Web Conference, ACM, 2019:2637-2643.
[39] Zhang Y, Xiong Y, Kong X, et al. Learning node embeddings in interaction graphs[C]. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management,2017:397-406.
[40] Shi C, Hu B, Zhao W X, et al. Heterogeneous information network embedding for recommendation[J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 31(2):357-370.
[41] Sun Z, Yang J, Zhang J, et al. Recurrent knowledge graph embedding for effective recommendation[C]. Proceedings of the 12th Conference on Recommender Systems,2018:297-305.
[42] Wang X, He X, Cao Y, et al. KGAT: Knowledge graph attention network for recommendation[C]. Proceeding of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019:950-958.
[43] Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need[J]. arXiv, 2017: 6000–6010.
[44] 刘纵横,汪海涛,姜瑛,等.基于自注意力机制与知识图谱的序列推荐算法[J]. 传感器与微系统, 2021, 40(2): 132-135.
[45] 申晋祥,鲍美英.基于注意力机制的深度学习推荐算法[J]. 计算机系统应用, 2021, 30(6): 220-225.
[46] 唐宏,范森,唐帆,等.融合知识图谱与注意力机制的推荐算法[J].计算机工程与应用. 2022(1): 1-13.
[47] Shaw P, Uszkoreit J, Vaswani A. Self-Attention with Relative Position Representations[C]. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018(2): 464-468.
[48] Zhou G, Song C, Zhu X, et al. Deep Interest Network for Click-Through Rate Prediction[J]. 2017.
[49] Zhou G, Mou N, Fan Y, et al. Deep Interest Evolution Network for Click-Through Rate Prediction[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019(33):5941-5948.
[50] Shuai Z, Yi T, Yao L, et al. Next Item Recommendation with Self-Attention[J]. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 311-320.
[51] 王宏琳, 杨丹, 聂铁铮, 等. 基于自注意力机制的属性异构信息网络嵌入的商品推荐[J].计算机研究与发展. 2022(1): 1-12.
[52] Wu Z, Pan S, Chen F, et al. A Comprehensive Survey on Graph Neural Networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2019.
[53] Wu F, Zhang T, Souza A, et al. Simplifying Graph Convolutional Networks[J]. 2019.
[54] Velikovi P, Cucurull G, Casanova A, et al. Graph Attention Networks[J].2017.

[55] Li Y, Tarlow D, Brockschmidt M, et al. Gated Graph Sequence Neural Networks[J]. Computer Science, 2015.
[56] Hamilton W L, Ying R, Leskovec J. Inductive Representation Learning on Large Graphs[J]. 2017.
[57] Li S, Wu C, Li H, et al. FPGA Acceleration of Recurrent Neural Network Based Language Model[C]. IEEE International Symposium on Field-programmable Custom Computing Machines. IEEE, 2015:111-118.
[58] Shi X, Chen Z, Wang H, et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting[J]. MIT Press, 2015.
[59] Rui Liu, Loannis Vlachos. Mutual information in the frequency domain for the study of biological systems[J]. Biomedical Signal Processing and Control. 2018.