

# 本科生毕业设计（论文）参考文献译文本

译文出处：

院 系 计算机科学与技术

专业班级 CS1501

姓 名 莫海

学 号 U201514489

指导教师 邵志远

**译文评阅**

|  |
| --- |
| **导师评语**  应根据学校“译文要求”，对学生译文翻译的准确性、翻译数量以及译文的文字表述情况等做具体的评价后，再评分。 |
| 评分：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_（百分制） 指导教师（签名）：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  年 月 日 |

# 基于图结构的协作排名

Bita Shams and Saman Haratizadeh

德黑兰大学

摘要

数据稀疏性，这是基于邻居的协同过滤中的常见问题，通常使项目推荐的过程复杂化。在基于排名数据计算用户相似度和推荐项的协同排名领域，这一问题更为严重。有人提出了一些基于图形的方法来解决数据稀疏问题，但它们存在两个缺陷。首先，它们不能正确地为用户的优先级建模，其次，当惟一可用的数据是一组排名而不是评级值时，就不能使用它们。

在本文中，我们提出了一种新的基于图形的方法，称为GRank，它被设计用于协作排名。GRank可以在一种新的三部分图结构中正确建模用户的优先级，并对其进行分析，直接推断出推荐列表。实验结果表明，与基于图形的推荐算法和其他协同排序技术相比，推荐质量有了显著的提高。

**关键词:协同排序，两两偏好，图形建模，推荐系统，个性化PageRank**

1.介绍

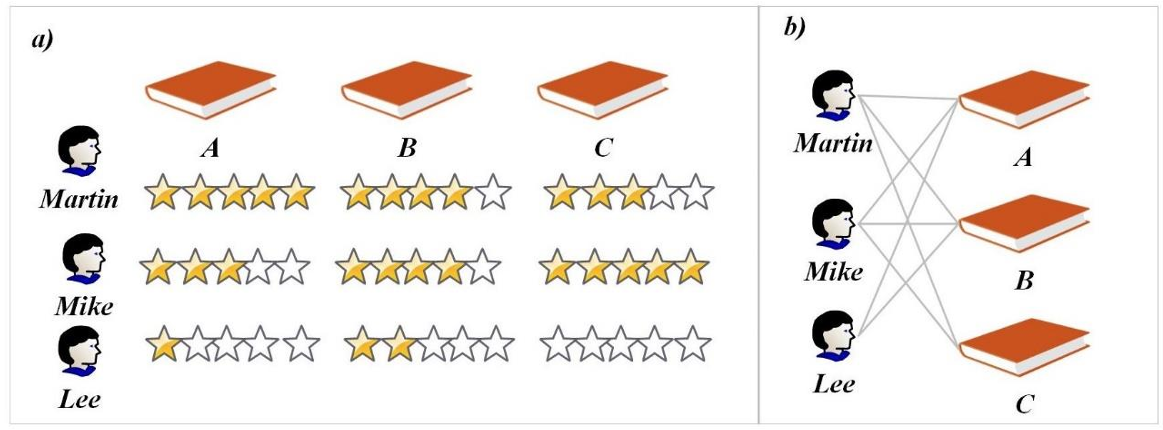
协同过滤(CF)技术是一种有效的算法，可以帮助人们过滤不相关的内容并提供有用的个性化推荐服务。这些技术试图学习模型，以根据用户过去的偏好预测用户将来需要的服务。

协作过滤技术可以分为两类：面向评级和面向排名的算法。面向评级的算法的目标是准确地预测用户的评级，然后推荐具有最高预测评级的项目给他。另一方面，称为协作排名的排名导向方法寻求从目标用户的角度直接预测项目的排名，而不明确地预测评级。要明白原因，请注意，推荐自然是排名任务，推荐算法真正需要的是提高Top-k排名的质量而不是预测费率(N. Liu&Yang,2008; Y Shi, Karatzoglou, & Baltrunas, 2012; Y Shi, Larson, & Hanjalic, 2010; Yue Shi,Larson, & Hanjalic, 2013). 此外，在许多应用程序中，我们所拥有的只是一组隐式反馈，而没有可用的评级数据，因此，在这种情况下不能使用基于评级的方法。注意，尽管评级和其他类型的显式反馈需要用户明确评估项目，但是可以通过跟踪用户与系统的交互（例如，点击，购买等）来自动收集隐式反馈。面向排名的协同过滤也可以应用于这种情况。

基于邻居的协同过滤（协同过滤的主要类别之一）根据相似用户的行为估计目标用户的排名/评级。尽管在这类算法中进行了多次研究，但它们仍无法精确计算用户的相似性。原因可以用稀疏性问题来解释，稀疏性问题是指在推荐系统中，用户已经对一小部分项目给出了反馈，因此，他们很少有足够的共同项目或成对比较来估计其真实的相似性/不相似性(Desrosiers& Karypis, 2011). 克服这个问题的一种方法是基于图形的推荐，其利用异构信息网络，即包含不同类型的节点和边缘的信息网络，以细化相似性度量(M. S. Shang, Fu, & Chen, 2008; Z.-K. Zhang, Zhou, & Zhang, 2010; Zhou, Ren, Medo,& Zhang, 2007) ，扩大社区，并直接计算用户和物品的亲密度(Chiluka, Andrade, & Pouwelse, 2011; Silva & Zaki, 2013; Xiang et al., 2010; Yao, He, Huang, Cao, & Zhang, 2013).

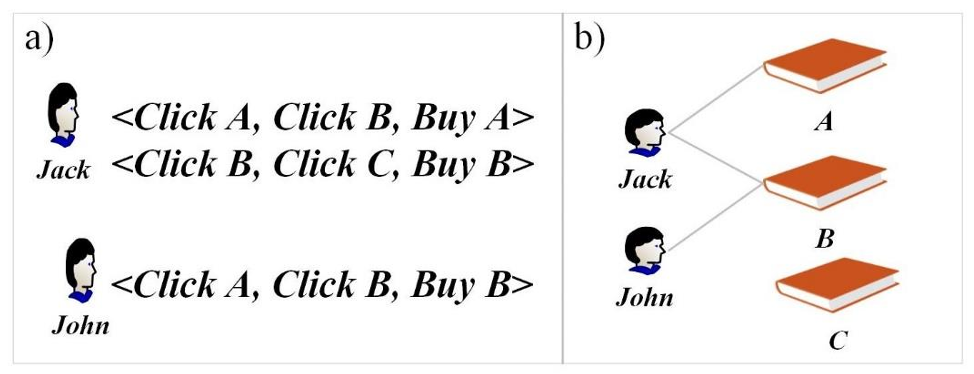
基于图形的推荐算法将用户和项目之间的关系表示为二分图，其中用户与他评定的每个项目之间存在加权或未加权的链接(Li & Chen, 2013; M. S. Shang et al., 2008; M.-S. Shang, Zhang, Zhou, & Zhang, 2010; Ting, Yan, & Xiang-wei, 2013; Xiang et al., 2010; Z.-K. Zhang et al., 2010; Zhou et al., 2007). 不幸的是，这种方法基本上是为评级/二进制反馈而设计的，并且对于基于邻居的协同过滤的排序导向类具有至关重要的不足。

第一个问题是当前基于图的方法无法捕获用户的偏好顺序。我们参考图1a的例子来说明这个缺点。对于项目A和B，Mike和Lee具有相同的首选顺序，而Mike和Martin对所有项目具有完全相反的首选顺序。当前基于图的算法将该数据表示为图1b（Sawant，2013; M.Shang Shang等人，2008）。直观地说，在这种图形建模下，大多数众所周知的图形邻近度量（例如，常见的邻居，距离，Katz和个性化的PageRank）将表明Mike更接近Martin而不是Lee，这是违反直觉的。



**图1。说明当前基于图形的结构无法捕获以评级形式收集的首选项数据的示例。**

已经针对二元隐式反馈提出了当前基于图的方法的第二个缺点，并且它们不能捕获由不同隐式反馈生成的用户的成对偏好（即，选择上下文）。很明显，选择上下文是可用于提高推荐质量的有价值的信息。要查看当前图表表示数据时如何丢失此类信息，您可以在图2的示例中观察到，John在一个会话中优先选择A项，而在另一个会话中优先选择项目B，而Jack， 喜欢B选项超过A。当前基于图形的隐式反馈表示，在用户和接收正反馈的项目之间建立联系(Chen, Wang, Huang, & Mei, 2012; Xiang et al., 2010; Z.-K. Zhang et al., 2010)。因此，这些算法无法区分异构隐式反馈（即购买，点击）。更重要的是，他们无法澄清杰克和约翰在比较项目A和B时不同意的事实，如图2b所示。



**图2。一个例子来说明当前基于图形的结构无法捕获通过浏览/购买历史记录收集的选择上下文。**

本文提出了一个名为GRank的新框架，它使用新的三方首选图（TPG）结构捕获用户的偏好，该结构演示了用户，项目和成对偏好之间的关系。 GRank还提供了一种新的排名算法，该算法扩展了个性化PageRank以获得top-k推荐。 据我们所知，该算法是第一种基于图的方法，能够捕获隐式反馈提供的偏好信息。 实验结果表明，与现有的协同排序算法以及可用的基于图的推荐系统相比，GRank具有更高的准确性。

本文的其余部分安排如下。 在第2节中，讨论了基于图表的推荐和协作排名技术的相关工作; 然后，我们在第3节中介绍了GRank框架的细节。第4节介绍和分析了实验结果。在第5节中，我们讨论了GRank如何解决目前协作排序和基于图形的推荐方法的缺点。 最后，在第6节中，我们总结并介绍了我们未来的工作。

2.相关工作

可以从许多不同的角度分析推荐的质量，包括准确性，(Koren, Bell, & Volinsky, 2009; Weimer & Karatzoglou, 2007)，覆盖性(Bellogin & Parapar, 2012; Cacheda, Carneiro, Fernández, & Formoso, 2011)，多样性(Adomavicius & Kwon, 2012; Said, Kille, Jain, & Albayrak, 2012; Zhou et al., 2010)，意外性(Lu, Chen, Zhang, Yang, & Yu, 2012; Xiao, Che, Miao, & Lu, 2014)，不确定性(M. Zhang, Guo, & Chen, 2015)，不确定性(M. Zhang, Guo, & Chen, 2015)，先令攻击检测性(Z. Zhang & Kulkarni, 2014)和可扩展性(Jiang, Lu, Zhang, & Long, 2011)。尽管所有这些方面都是推荐系统成功的重要因素，但建议的准确性是这方面的关键因素，并且已形成一系列核心研究以实现更高水平的推荐准确性。本文在此类别中，提出了一种新颖的基于图形的框架，当唯一可用的信息是偏好数据时，该框架在没有上下文信息位置的情况下提高了推荐的准确性。上下文信息通常是指用户和系统的交互发生的环境状态（例如，时间，位置，情感等）。 另一方面，“选择上下文”反映了用户做出选择的选项。

在这里，我们将回顾与我们提出的算法的主要方面相关的现有研究：协作排名和基于图的推荐。

2.1 协作排名

协作排名是一类协作过滤算法，旨在预测用户如何对项目进行排名。 正如我们之前提到的，尽管存在一些相似之处，协作排名算法与面向评级的协同过滤（即协作评级）不同，因为协作评级算法依赖于评级数据并尝试最小化评级预测误差，而协作排名算法 不依赖于评级数据。 他们可以使用任何类型的偏好数据并尝试最小化秩预测误差。

在信息检索领域中，协作排名和学习到排名问题的两个概念之间也存在一些相似之处，因为它们都试图对on类型的实体进行排序，例如，文档/项目，用于另一种类型的目标实体，例如查询/用户。然而，这两种问题在实践中是不同的。在学习排名问题时，存在一系列明确的共同特征，例如两种类型的实体，查询和文档中的术语频率（Balakrishnan＆Chopra，2012; Fan＆Lin，2013; Y Shi et al。，2010; Volkovs＆ Zemel，2012）然而在协作排名问题中没有可用或用于关联用户和项目实体的这些功能（Balakrishnan＆Chopra，2012; Y Shi等人，2010; Volkovs＆Zemel，2012）。由于这些领域中问题的性质之间的这种重要差异，已经出现了用于解决这些问题的不同类别的算法。这些方法可以分为两类：矩阵分解（MFCR）和基于邻居的算法（NCR）。

协作排名中的矩阵分解技术，尝试学习代表性潜在特征，以准确预测每个用户的项目排名。 CofiRank是第一个使用矩阵分解技术优化排名指标的算法（Weimer，Karatzoglou，＆Smola，2008; Weimer＆Karatzoglou，2007）。另一种技术ListRank，通过估计前1的概率来推断项目的排名（Y Shi et al。，2010）。URM是另一种结合了ListRank和概率矩阵分解的模型，以提高系统在排名和评级方面的准确性（Yue Shi，Larson，et al。，2013）。BoostMF是另一种矩阵分解方法，它基于偏好数据顺序学习一组弱矩阵分解模型（Chowdhury，Cai，＆Luo，2015）。贝叶斯个性化排名及其变体，尝试优化贝叶斯预测模型的曲线下面积（AUC），该模型基于相关和不相关项目之间成对比较的一组预测生成（Lerche＆Jannach，2014; Pan，Zhong， Xu，＆Ming，2015; Rendle，Freudenthaler，Gantner，＆Schmidt-thieme，2009）。最近，已经提出了一些方法，其关注于正确预测具有最高等级的物品的成对偏好（Christakopoulou＆Banerjee，2015; Dhanjal，Clémençon，＆Gaudel，2015）。 Climf（Y Shi et al。，2012）和xClimf（Yue Shi，Karatzoglou，Baltrunas，＆Larson，2013）是另外两种利用矩阵分解技术优化推荐列表的平均倒数秩（MRR）的算法（Y Shi et al。，2012; Yue Shi，Karatzoglou，et al。，2013）。

虽然基于矩阵分解的协作排序方法由多种算法和方法组成（Weimer等，2008; Weimer＆Karatzoglou，2007），但GRank的概念在概念上与它们不同，因为它不代表潜在特征空间中的数据。相反，它以图形结构的形式对排名数据进行建模，使其能够直接估计用户和项目的紧密程度，并根据该结构进行推荐。因此，从某种意义上说，GRank位于另一类称为基于邻居的协作排名（NCR）的推荐算法中。 尽管第二类算法具有其优点，但是这种方法仍然没有被研究过，到目前为止很少有成功的NCR算法被提出。

EigenRank（N.Liu＆Yang，2008）是最着名的NCR技术，它基于类似于目标用户的用户的成对偏好来推断总排名。 EigenRank使用Kendall相关性计算用户的相似性，该相关性考虑了用户对成对比较的一致性和不一致性。在估计相似性之后，EigenRank估计偏好矩阵，其元素是邻居偏好的加权线性组合。最后，它使用贪婪或基于马尔可夫的方法来推断项目的总排名。据我们所知，所有NCR技术都遵循EigenRank提出的主要方法，稍作修改。 EduRank（AvSegal，Katzir，＆Gal，2014），WSRank（Meng，Li，＆Sun，2011）和Cares（Yang，Wei，Wu，Zhang，＆Zhang，2009）为不同的应用定制了EigenRank。 VSRank（Wang，Sun，＆Gao，2014）着重通过考虑相似度计算中每个成对比较的重要性来改进Kendall相似性度量。然而，这种方法仍然存在稀疏问题，因为它仍然依赖于共同的成对比较来计算相似性。

如前所述，GRank旨在通过引入一种新的基于图形的方法来建模和分析数据，从而解决基于邻居的协作排名的稀疏性问题。 它也不同于当前基于邻居的算法，因为它不遵循传统的三步框架，而是直接估计用户的偏好。

2.2 基于图形的推荐方法

尽管没有基于图形的方法设计用于协作排名，但是在推荐系统的其他领域中已经进行了许多最近的研究。 在这里，我们将简要回顾这些算法，并阐明当前工作与它们之间的主要区别。

基于图形的推荐算法由两个步骤组成：构建表示数据的图形并通过分析图形来提出建议。 这些推荐算法已经利用了不同类型的图。 但是，在所有这些中，图表的主要组成部分是用户与被他们评级的项目之间的关系。因此，最常见的方法是构建一个二分网络，其中连接是从一部分网络，用户到另一部分，项目。一旦构建了二分图，就可以使用几种方法使用来自目标用户的邻居的信息对项进行排名。 在这个领域中使用了诸如使用普通邻居，Katz相似度，扩散分数和个性化PageRank等方法（Huang，Li，＆Chen，2005; Z.-K. Zhang et al。，2010）。

最近的方法通过向其添加一些层来扩展双向网络。一些研究人员（Xiang et al。，2010）考虑使用会话层来考虑用户的长期和短期偏好，以便在特定时间内提出建议。其他人（Yao等人，2013）在多层结构中使用了不同类型的节点，以通过图中的随机游走来进行上下文感知推荐。在（Z.-K.Zhang等人，2010）中，三层图用于通过考虑用户使用基于扩散的分数（Zhou等人，2007）分配给项目的标签来改进推荐。在一些作品中（Lee，Park，Kahng，＆Lee，2013; Yu，Ren，Sun，＆Gu，2014），网络结构已经修改。他们考虑星形异构网络，其中用户和项目可以连接到不同类型的节点。他们使用此图结构来改进基于模型的建议（Yu et al。，2014），或通过改进异构网络中的个性化PageRank算法来提出建议。 （Lee et al。，2013）。我们强调，这些算法都不是为了捕获用户的选择上下文和偏好而设计的。此外，它们中的大多数依赖于在所有应用中系统不存在或不可用的上下文信息（例如，时间，内容等），并且收集可能是昂贵的。

3.GRank：基于图形的协作过滤框架

已经表明，异构信息网络具有很强的能力来模拟推荐系统的不同实体之间的关系（Cong，2009; Sun，Han，Yan，＆Yu，2011; Yu等，2013,2014）。 在本文中，我们寻求为面向排名的推荐系统提出一种有效的图形方法，称为基于图的协作排名，或GRank。 在下文中，我们首先定义基于图形的协作排名问题及其目的。 然后，我们提出了理解算法所需的一些定义。 接下来，我们介绍一种新颖的异构图结构，称为三方偏好图（TPG），它在用户，偏好和聚合结构中的项之间嵌入不同类型的关系。 最后，我们建议一种有效的算法来利用TPG来为每个目标用户排序项目。

3.1 问题定义

从协作排名的角度来看，推荐系统可以用一组用户𝑈= {𝑢1，...，𝑢𝑀}，项集𝐼= {𝑖1，...，𝑖𝑁}和观察集𝑂= {<𝑢，𝑖， 𝑗>}这是用户偶尔声明的一组首选项。 通常，我们定义观察𝑜= <u，i，j>其中u∈U，i∈I和j∈I表示用户u优先于i而不是j。 为简单起见，我们称第一项为理想项，第二项为不良项。

请注意，成对比较是排名数据的一般形式，并且可以使用以下规则将各种偏好（例如评级，浏览历史）转换为一组成对比较：

**规则1.**设L为评级矩阵，其中𝐿𝑢𝑖表示项目i的用户u的评级。 偏好观察集可以通过𝑂= {<𝑢，𝑖，𝑗> |𝐿𝑢𝑖≠0，𝐿𝑢𝑗≠0，𝐿𝑢𝑖>𝐿𝑢𝑗}获得。

**规则2.**设L是正反馈的矩阵（例如，Like），其中非零元素𝐿𝑢𝑖表示用户u喜欢项目i。 类似地，令D是负反馈的矩阵（例如不喜欢），其中非零元素𝐷𝑢𝑖表示用户u不喜欢项目i。 偏好观察集可以通过𝑂= {<𝑢，𝑖，𝑗> |𝐿𝑢𝑖≠0，𝐷𝑢𝑗≠0}获得。

**规则3.**设W为定义为𝑊= {𝑤1，...𝑤|𝑤|}的会话集，我们可以将观察集创建为  
𝑂= {<𝑢，𝑖，𝑗> |∃𝑤∈𝑊𝑊=𝑤.𝑢，𝑖∈𝑤𝑤𝐵，𝑗∈𝑤.𝐶}  
w.u是会话w中的用户，w.B是在会话w中购买的项目集合，w.C是点击但未在会话中购买的项目集合w。

给定偏好数据集O，基于图的框架将面临一个关键问题，即如何对图结构中的偏好数据集中可用的信息进行建模。 为了回答这个问题，我们首先对一些信息进行分类，即面向等级的推荐系统的有效图形建模应该捕获。

• **用户在优先级方面的相似性：**面向排名的推荐系统的两个用户，当他们对某些成对比较有相似的观点时，被认为是相似的（N.Liu＆Yang，2008; Wang et al。，2014 或者在一些比较中更喜欢特定的项目A，即使这两个用户对其优先选择的项目是不同的（Meng等人，2011）。 排序数据的组织良好的图形模型应反映用户之间的两种相似性。

• **比较之间的相关性：**相关的成对比较是那些由用户同样投票的偏好。 应通过分析数据的图形表示来简单地发现这些比较。

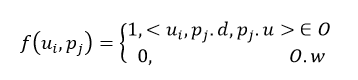
• **项目的相似之处：**类似项目是由一组类似用户同样支持/不喜欢的项目。 有效的图形建模应该清楚地反映这些项目的接近程度。

• **预测用户的优先级：**面向排名的推荐系统的最终目标是推断目标用户对未见项目的总排名，并推荐前k项。 因此，这些系统的图形表示负责向目标用户的有效和有效推荐，并且理想地，秩数据的图形表示可用于直接预测排名。

3.2 图构造

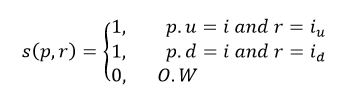
在我们继续解释GRank如何构建和利用基于首选项数据集的图结构之前，我们需要定义一些基本概念：

**Definition.1.** 成对偏好p是元组<𝑖，𝑗>表示i优于j的偏好。 我们称i为p中的理想项，由p.d表示，j为p.u表示的p中的不良项。 成对偏好集P形式上定义为𝑃= {<𝑖，𝑗> |𝑖∈𝐼，𝑗∈𝐼，𝑖≠𝑗}。

**Definition.2.** 用户可能对两个项目有一定的偏好。 协议功能𝑓：𝑈×𝑃→{0,1}表示用户是否同意偏好and，定义为：  
   
其中𝑢𝑖∈𝑈，𝑝𝑗∈𝑃𝑃O是第3.1节中定义的偏好观察集。

**Definition.3.** 抽象地说，每个项目都有两个方面：理想的一面和不希望的一面。 我们将项目的合意性定义为𝐼𝑑= {𝑖𝑑|𝑖∈𝐼}，其中𝑖𝑑代表项目i的理想方面。 此外，项目的不受欢迎集合被定义为𝐼𝑢= {𝑖𝑢|𝑖∈𝐼}，其中𝑖𝑢代表项目i的不希望的一面。 我们还将代表集定义为=𝐼𝑑∪𝐼𝑢，每个项目包含两个元素，每个元素一个元素。

**Definition.4.** 支持函数𝑠：𝑃×𝑅→{0,1}表示偏好p是否支持代表r。 在形式上，我们将s定义为：



其中𝑝∈𝑃和𝑟∈𝑅。

基于这些定义和概念，我们现在可以解释GRank如何使用称为Tripartite Preference Graph的结构对偏好数据进行建模。 形式上，三方偏好图（TPG）是一个三方图TPG（𝑈∪𝑃∪𝑅，𝐸𝑈𝑃∪𝐸𝑃𝑅），其中U是用户集，P是成对偏好集，R是代表集。 EUP = {（𝑢，𝑝）|𝑓（𝑢，𝑝）= 1，𝑢∈𝑈，𝑝∈𝑃}，是U和P中节点之间的边集，𝐸𝑃𝑅= {（𝑝，𝑟）| 𝑠（𝑝，𝑟）= 1，𝑝∈𝑃，𝑟∈𝑅}，是将P中的节点连接到R中的节点的边集。

更清楚的是，TPG包含三个层，每个层包含不同类型的节点：

• **用户：**TPG的第一层包含每个用户的一个节点。

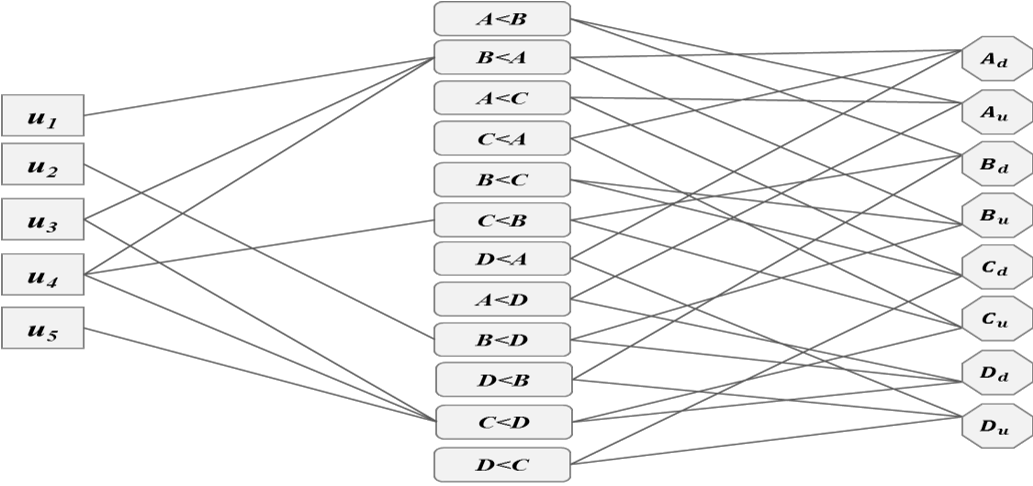
• **首选项：**首选项层包含与所有可能的成对首选项集合对应的节点𝑝∈𝑃。 简单，每个偏好的对应节点𝑝= <𝑖，𝑗>以（“𝑖>𝑗”）的形式标记，清楚地表明了i对j的偏好。

• **代表：**代表层包含分别由𝑖𝑑和represented表示的项目的理想和不良代表的集合。

TPG还包含两种类型的链接：

• **用户首选项链接：**𝐸𝑈𝑃是将每个用户u连接到其声明的首选项的边集。 更清楚的是，对于每个偏好数据<𝑢，𝐴，𝐵>∈𝑂，在用户𝑢和标有“𝐴>𝐵”的偏好节点之间存在链接。

• **偏好代表链接：**𝐸𝑃𝑅是将每个首选项与其支持的代表相关联的一组链接。 例如，标有“𝐴>𝐵”的偏好节点连接到对应于desirable的理想情况的节点，即𝐴𝑑和B的不希望的情况，即𝐵𝑢。 该链接用于模拟“𝐴> preference”偏好的事实，隐含地支持“𝐴”的项目期望的一面和“𝐵”的不期望的一面。



**图3所示。一个例子:TPG由一个包含5个用户、4个项目和9个按对分配的首选项的系统构建。**

如前所述，TPG是一个三方图，其中首选项层连接到其他层：用户层和项层。 通过不同类型的路径遍历TPG揭示了面向排名的推荐系统中的不同类型的信息，其中的一些示例在表1中给出。

**表1. TPG中的元路径及其语义**

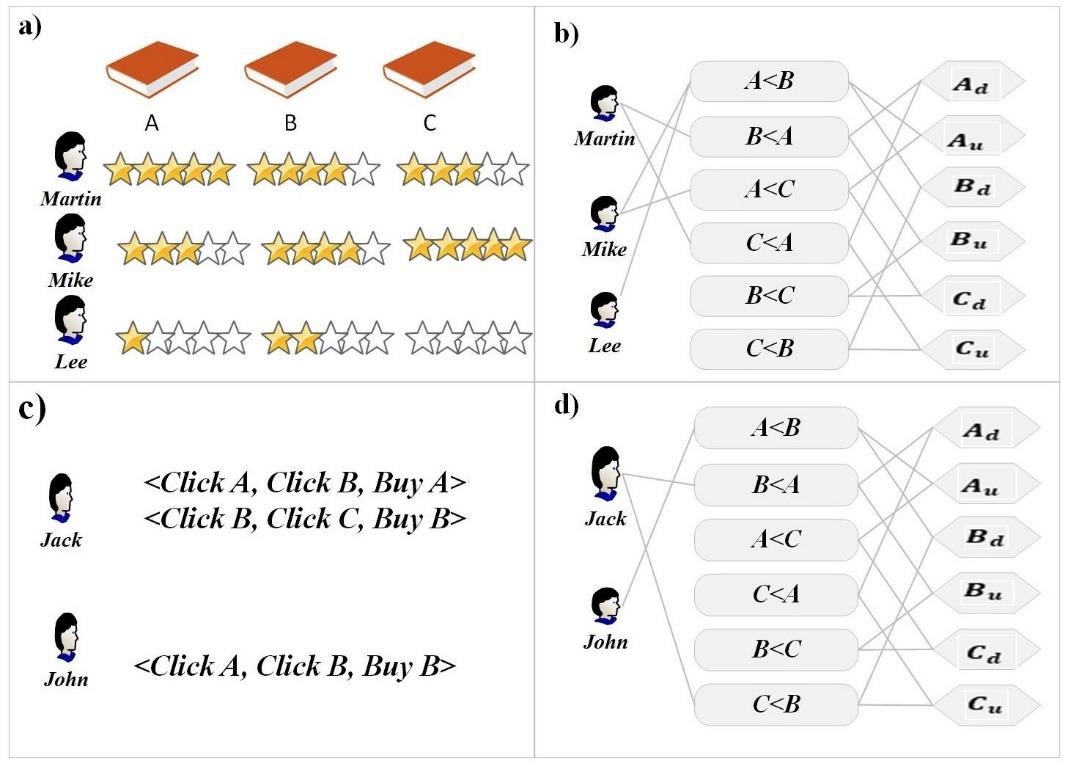
|  |  |
| --- | --- |
| **元路径** | **语义** |
|  | 用户在成对偏好方面的相似性 |
| 𝑼 − 𝑷 − 𝑰𝒅 − 𝑷 − 𝑼  𝑼 − 𝑷 − 𝑰𝒖 − 𝑷 − 𝑼 | 用户对不同项目的项目的可取性/不可取性的协议。 |
| 𝑷 − 𝑼 − 𝑷 | 成对比较之间的相关性 |
| 𝑰𝒅 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒅  𝑰𝒖 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒖 | 项目之间的直接关系; 同时受用户青睐/不受欢迎的项目 |
| 𝑰𝒖 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒅  𝑰𝒅 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒖 | 项目之间的间接关系。 相反按用户排名的项目。 |

构造TPG的伪代码在Algorithm.1中给出。它首先生成对应于𝑀用户的𝑀节点和2𝑁节点，以获得𝑁项目的理想和不合需要的情况。然后，它为每个可能的偏好数据生成𝑁（𝑁 - 1）个节点。每个偏好数据具有期望的和不期望的边，分别由p.d和p.u表示。下一步是添加用户和首选项之间的链接。它扫描首选项数据库，并为每个<user，item1，item2>三元组，它在用户的相应节点和首选项层之间添加一个链接。更详细地说，getUserNode（u）将相应的节点返回给用户u，而getPreferenceNode（i，j）返回表示i优先于j的节点p。之后，在节点u和p之间创建链接。最后，算法扫描偏好节点，其中偏好节点p表示“p.d> p.u”。对于这样的节点，算法使用getDesirableNode（p.d）和getUnDesirableNode（p.u）找到相应的代表节点，并在p和每个代表之间插入边。

**Algorithm 1. Construction of Tripartite Preference Graph (TPG)**

|  |
| --- |
| **Input:** Set of users U, Set of items I, Observation set of preference (O) **Output:** Tripartite graph (*G*) |
| Initialize a graph G  *//Initializing user layer*  For each item 𝑢 ∈ 𝑈  Create a node 𝑢 in user layer    *//Initializing representative layer*  For each item 𝑖 ∈ 𝐼  Create a desirable-node 𝑖𝑑 in the representative layer  Create an undesirable-node 𝑖𝑢 in representative layer    *//Initializing preference layer*  For each item 𝑖 ∈ 𝐼  For each item 𝑗 ∈ 𝐼  If (𝑖 ≠ 𝑗)  Create a node *p* in preference layer *p.d = i;*  *p.u=j;*    *// Connecting user and preference layer* For each o:(u,i,j) ∈ 𝑂 *u* = G. GetUserNode (u); *p* = G. GetPreferenceNode (i ,j); Connect node *u* to node *p*.    *//Connecting preference and item layer*  For each node v ∈ preference-layer  𝑟𝑑= G.GetDesirableNode (p.d)  𝑟𝑢= G.GetUnDesirableNode (p.u)  Make a link between *p* and 𝑟𝑑  Make a link between p and 𝑟𝑢. |

**Example.1.** 图4说明了TPG如何反映图1和图2中提到的偏好数据。 如图4b所示，TPG清楚地表明Mike和Martin（图1a和图4a）具有不同的偏好，因为它们在TPG中既没有共享任何邻居也没有相互之间的路径。 另一方面，迈克和李分享一个共同的邻居，表示他们对A和B的比较达成一致。对于图4c的例子中的杰克和约翰也是如此，其中TPG表示（图4d）显然可以反映出他们 对A和B的比较没有相同的看法。注意，当前基于图的方法不能模拟用户在这些样本中的偏好，如图1b和图2b所示。



**图4 TPG表示的例子分别如图1和图2所示。**

3.3 使用TPG图表的Top-k推荐

在推荐系统中通常的假设是用户对他们的邻居喜欢的项目感兴趣或者与他们喜欢的项目类似。 如前所述，TPG提供了一个富裕的平台来确定用户的相似性和项目关系。 GRank利用用户与期望/不期望的代表的亲密度来估计目标用户喜欢/不喜欢特定项目的程度。 换句话说，给定目标用户u和描述观察集的TPG，GRank定义函数𝐺𝑅：𝑈×𝐼→ℝ，用于根据节点u的接近度为每个用户u预测每个看不见的项目i的好坏。 项目i的期望和不期望的代表性节点。 然后GRank建议你为u计算出最高GR值的项目。

可以基于两种一般类型的路径来估计用户u的项目i的期望/不期望：可取性和不可保持性路径。 期望路径是<𝑢，𝑣1，𝑣2，...，𝑣𝑚，𝑖𝑑>的形式，并且表示目标用户u与项目i的期望情况的接近程度。 类似地，不受欢迎的路径是<𝑢，𝑣1，𝑣2，...，𝑣𝑚，𝑖𝑢>的形式，并描绘了目标用户u的接近程度和项目i的不良情况。

直观地，在目标用户u和项目i之间存在大量的期望/不期望路径。 因此，需要某种形式的邻近度量来比较目标用户和项目之间的期望/不期望路径的数量。

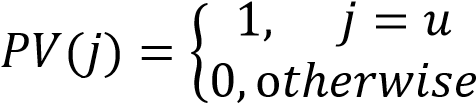
最近，已经提出了一些用于分析异构网络的邻近度量。 不幸的是，这些方法严重依赖于元路径的定义，并且需要将不同元路径的重要性权重作为输入参数来计算节点之间的接近度（Lee et al。，2013; Sun et al。，2011）。 另一方面，衡量网络中节点接近度的一般方法是个性化PageRank或PPR（Page，Brin，Motwani，＆Winograd，1999），它被认为是基于节点对节点进行排名的最有效措施之一。 它们可以从网络中的某组节点中获得。 它为与目标用户更接近的项目提供高分数，这些项目涉及广泛的图形属性，例如距离或它们之间的路径数量（Lee et al。，2013）。 GRank定义了一个基于PPR的度量来计算邻近度，但在引入该度量之前，首先我们简要回顾一下PPR的概念。

形式上，节点的个性化PageRank表示具有给定倾斜重启分布的随机游走者将跳转到该节点的概率。 PPR可以被视为具有重启的马尔可夫过程，并且由等式1定义.

PPR(t) = α · T · PPR(t − 1) + (1 − α) · PV (1)

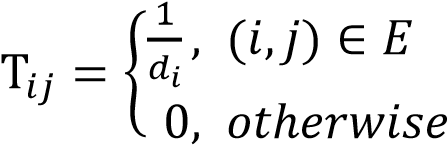
其中PPR（t）表示𝑡 - 𝑡ℎ迭代的秩向量，𝑇是转移矩阵，α是阻尼因子，“𝑃𝑉”是用户特定的个性化向量。 在大多数应用中，α设定为0.85。

要计算某些图节点（如项目节点）与特定节点（如目标用户节点）的紧密程度，个性化PageRank需要定义个性化向量𝑃𝑉，如公式2所示。

 (2)

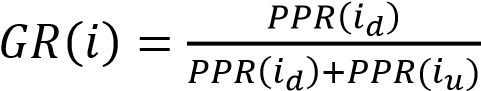
其中𝑢是目标节点。

给定具有所有节点V的集合和所有边缘E的集合的图G（V，E），转换矩阵的每个元素从等式3获得。

 (3)

其中𝑑𝑖是𝑖 - 𝑡ℎ节点的度数。

如上所述，PPR可用于在TPG中找到每个节点与目标用户节点的接近程度。 更清楚地，TPG中的目标用户的个性化PageRank估计从目标用户开始的随机游走者将遵循到每个项目的期望和不期望的代表的路径的概率。 GRank基于i的期望和不期望的情况的PPR来定义目标用户的项目i的良好性，如在等式4中那样。

 (4)

人们可以预期𝐺𝑅（𝑖）会给那些理想案例的个性化PageRank远远高于其不良案例的项目提供最高分。 为了向目标用户提出建议，根据其GR值对项目进行排序，并建议使用前k个项目。

算法2总结了GRank的top-k推荐方法。 GRank需要为每个目标用户u计算节点的个性化PageRank。 为此，它首先分别使用方程3和方程2定义TPG的转移矩阵和u的个性化向量。 然后，GRank随机初始化PPR值，然后将它们归一化为它们的总和。 接下来，它使用Eq.1更新PPR值并迭代直到收敛。 之后，使用等式4计算每个项目的GR值。 最后，基于它们的GR值对项目进行排序，并且向目标用户u推荐前k个项目。

**Algorithm 2. 关于TPG的Top-K建议**

|  |
| --- |
| **Input:** Tripartite graph (*G*), Target user *u, number of recommended items K, set of Items I* **Output:** The best *k items.* |
|  |
| Initialize transition matrix *T* through Eq.3  Initialize personalized vector *PV* through Eq.2  Randomly initialize  t=1  Repeat until convergence    For each item  Calculate GR value for items through Eq.4  Put items in descending order of their GR values in list L  Return as the recommendation list the first k items in L |

3.4 计算复杂性

GRank由两个阶段组成：图形构建和推荐任务。 假设𝑀是用户数，𝑁是项目数，𝑆是所有用户分配的总成对偏好数。 显然，可能的成对比较的总数将是𝑁（𝑁 - 1）。

TPG包含𝑁（𝑁 - 1）+𝑀+2𝑁顶点和𝑆+2𝑁（𝑁 - 1）边。 构建图的时间复杂度取决于实现方法。 如果我们使用邻接表，那么它的时间复杂度为O（𝑁2+ M + N + S +2𝑁2）=𝑂（𝑁2+𝑀+𝑆）。。 另外，我们经常知道𝑆=𝑐𝑁2其中c是一个小常数（例如，MovieLense100K为2.48）。 因此，图形构建阶段所需的时间为O（𝑀+𝑁2）。

推荐任务的时间复杂度等于个性化PageRank（PPR）计算的时间复杂度。 个性化PageRank的计算复杂度为𝑂（𝑡𝐸），其中𝐸是图形边缘的数量，𝑡是个性化PageRank收敛前所需的迭代次数。 在TPG中，我们有（𝑆+2𝑁2 - 2𝑁）边，正如我们所提到的，对于一个小的常数c，我们期望𝑆=𝑐𝑁2。 因此，GRank中推荐的时间复杂度预计为𝑂（𝑡𝑁2）。 注意，对于诸如TPG的稀疏图，𝑡是一个小数字。 在我们的实验中，它不超过20。

请注意，GRank的推荐具有比EigenRank更好的计算复杂度，EigenRank是最公认的基于内存的CR。 EigenRank的计算复杂度大约为𝑂（𝑀𝑁2+𝐾𝑁2+𝑁2），其中𝑂（𝑀𝑁2）用于计算目标用户与所有其他用户之间的相似度，𝑂（𝐾𝑁2）用于估计偏好矩阵，，（𝑂（ 𝑁2）用于推断总排名。

4. 实验设置和结果

我们进行了一系列评估GRank算法的实验。 在这里，我们将首先详细描述实验方案。 然后，我们将分析GRank的排名质量和可扩展性。

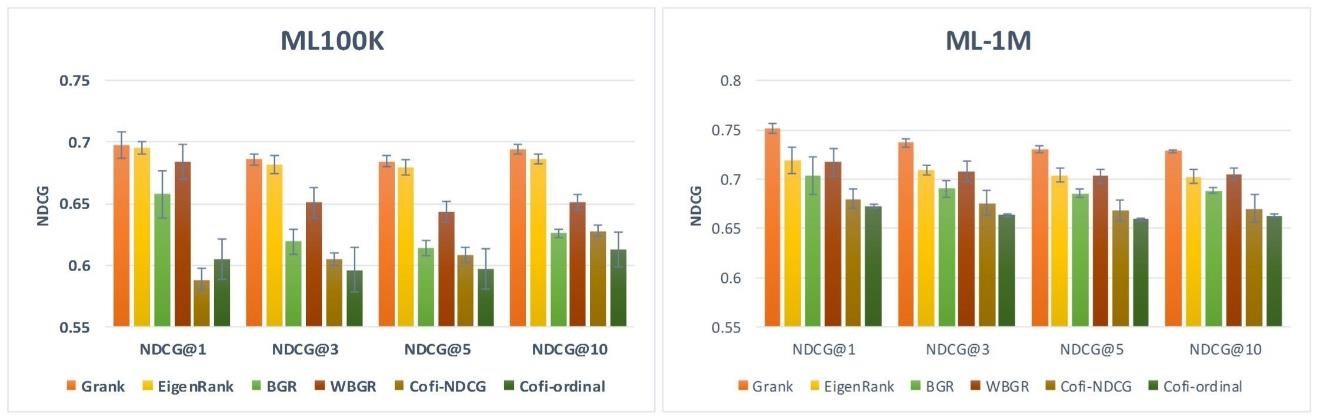
4.1 实验设置

#### 4.4.1 数据设置

我们对两个广泛用于相关工作的公开数据集进行了所有实验（Fan＆Lin，2013; Rendle等，2009; Yue Shi，Larson等，2013; Volkovs＆Zemel，2012; Wang等 。，2014）：这两个数据集都是由Movielens组生成的，但包含不同数量的用户，项目和评级。 第一个数据集Movielens-100K包含100,000个评级（1-5级），由943个用户分配给1,682部电影。 第二个数据集Movielens-1M由3,052个电影组成，评分为6,040位用户。 此数据集中有一百万个评级。 由于GRank旨在使用成对偏好数据，因此我们使用规则1将评级信息转换为一组成对比较：如果项目＃1已被评级，我们创建了数据<u，项目＃1，项目＃2）的首选项实例 用户u高于第2项。

#### 4.4.2 评估方法

在我们的实验中，我们遵循了相关工作中广泛使用的标准协议（Balakrishnan＆Chopra，2012; Fan＆Lin，2013; J. Liu，Wu，Xiong，＆Liu，2014; Rendle et al。，2009; Y Shi et al 。，2012; Volkovs＆Zemel，2012）。 我们分析了我们的算法在不同用户配置条件下的有效性，关于用户评级的数量; 对于每个用户，随机抽取固定数量的1/3评级并将其放置在训练集中，其余评级进入测试集。 我们的实验涉及𝑇= 20,30,40,50项。 对于每个𝑇，我们确保我们可以比较测试集中每个用户至少10个评级项目的算法。 因此，分别少于30,40,50和60项的用户从列车和测试装置中掉落。 我们通过随机抽样生成了两个数据集的5个变体，并报告了测试集的所有变体的平均性能。



**图5.在NDCG方面的算法性能比较，其中T = 20。**

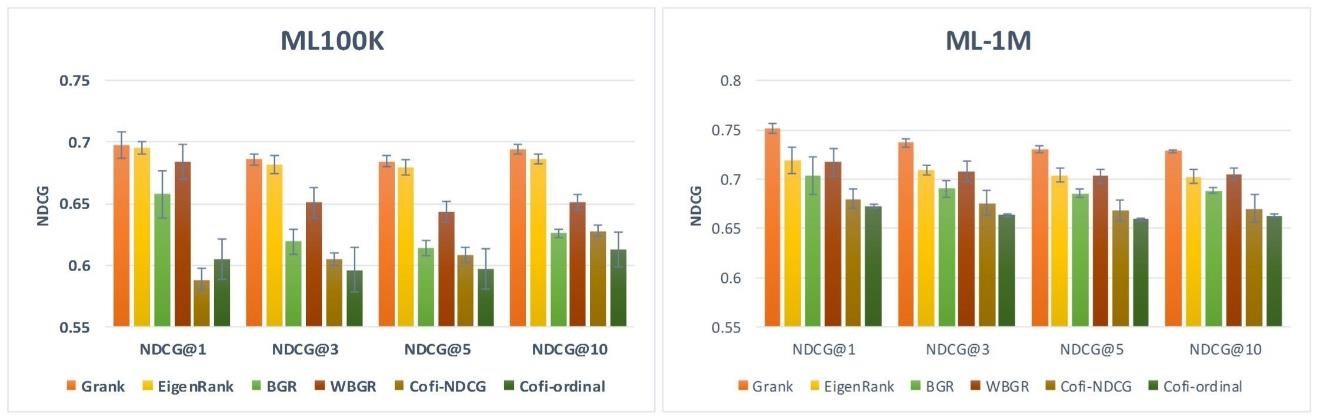
#### 4.1.3 基线算法

由于GRank在基于邻居的协作排名和基于图的推荐之间建立了联系，我们将其性能与这两类方法的最新方法进行了比较。 为了进一步分析，我们还将我们的算法与Cofi-Rank进行比较，Cofi-Rank是能够基于成对偏好数据进行推荐的最先进的矩阵分解技术。 以下简要介绍了这些算法：

**• CofiRank：**CofiRank（Weimer＆Karatzoglou，2007）是最先进的MFCR技术之一，它提取潜在的表示以优化结构化损失函数。 CofiRank有几个扩展。 在我们的实验中，我们使用CofiRank-Ordinal和CofiRank-NDCG作为我们的基线算法。 （Weimer等，2008）。 CofiRank-Ordinal最小化预测排名列表中不一致偏好的数量，同时CofiRank-NDCG试图最大化NDCG。 我们使用了CofiRank的公开代码，并采用了（Weimer等，2008）中建议的参数的最佳值。

**• EigenRank：**EigenRank（N.Liu＆Yang，2008）是NCR技术家族中另一种着名的算法。 我们已经使用邻域大小100和ε= 0.85实现了EigenRank的随机游走版本，据报道这是该算法的最佳参数值（N.Liu＆Yang，2008）。

**• Graph-based recommendation:** 此外，我们将GRank与基于图的推荐算法进行比较，该算法利用双向图结构来模拟用户项交互（如图1）。 然后，使用重启的随机游走来对双分图中的项目进行排名（Chiluka等，2011）。 我们已经实现了该算法的两个版本，缩写为BGR和WBGR，它们分别将用户链接到他评定的未加权和加权链接的项目。 在加权版本中，用户u和项目i之间的边缘权重被设置为等于u到i的等级。



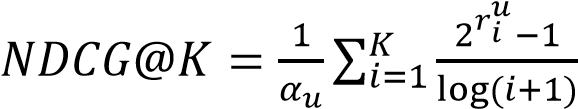
**图6.在NDCG方面的算法性能比较，其中T = 30**

4.2 结果

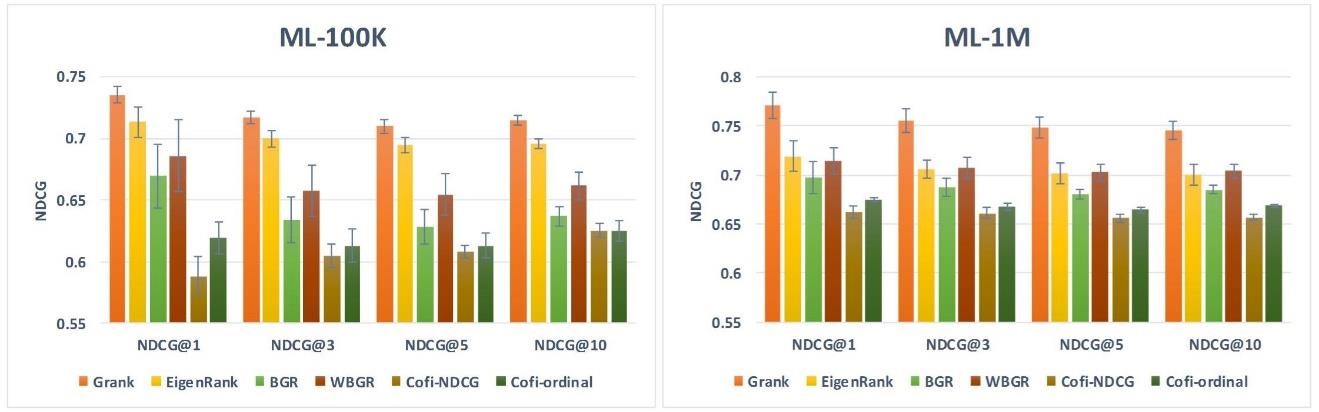
#### 4.2.1 精度

遵循应用于推荐系统的标准评估策略，我们通过比较他们的top-k建议的质量来评估模型的推荐性能。

归一化折扣累积增益（NDCG）是一种广泛用于评估CR技术的评估指标。 对于用户u的top-K建议中NDCG的定义如下:

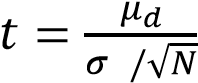
 (5)

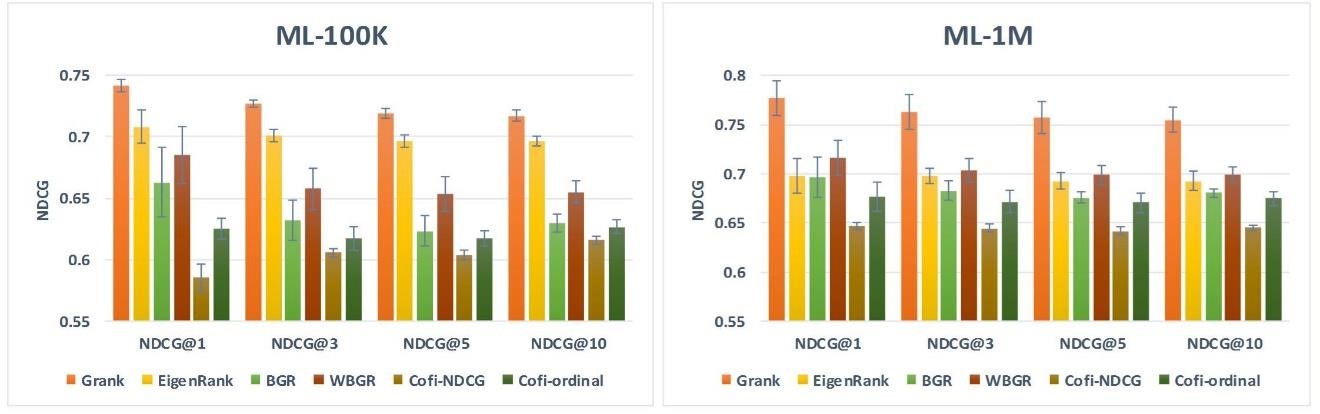
其中𝐾是推荐列表的长度，𝑟𝑖𝑢是用户u给推荐列表中第i个项目的评级，α𝑢是确保𝑢的理想推荐的NDCG等于1的归一化因子。 在本文中，我们报告NDCG @ 3，NDCG @ 5和NDCG @ 10在所有用户中平均性能。



**图7.在NDCG方面的算法性能比较，其中T = 40。**

ML-100k和ML-1M算法的性能如图5，图7和图8所示。 在大多数实验中，GRank优于其他算法。 基于对足够大的数据样本的一组实验报告结果，并且在常见的样本集上测试不同的算法。 因此，为了测试GRank和其他算法的性能之间的差异是否显着，我们可以对结果进行一组配对t检验。 （Fouss，Pirotte，Renders，＆Saerens，2007; Guo，Zhang和Yorke-Smith，2015; Kim＆Ahn，2008; Shani＆Gunawardana，2011; Ting-Peng，Hung-Jen，＆Yi-Cheng，2006; Zhen et al。，2009）。

给定{𝑥1，...，𝑥𝑁}是从数据集导出的样本集，让μ𝑑和σ𝑑作为差的平均值和方差𝑑𝑖=𝑎𝑖 - 𝑏𝑖其中𝑎𝑖和𝑏𝑖表示i上两个方法A和B的性能 样本。 为了确定A是否明显优于B，我们认为零假设是μ𝑑= 0而替代假设是μ𝑑≠0如果由t-获得的p值，则拒绝零假设以支持备选假设。 统计量 ，低于显着性阈值（例如0.01）。 表2显示了p值，表明GRank w.r.t算法的性能显著优于其他算法。



**图8.在NDCG方面的算法性能比较，其中T = 50。**

实验结果可归纳如下：

• 在大多数评估条件下，FRank明显优于所有算法。 然而，Eigen Rank和WBGR在ML-100k和Ml-1M中显示出小于1％的改善，其中T = 20。

• GRank的性能分别比WBGR和BGR高出6％和8％。 该结果再次强调了捕获用户偏好及其推荐选择背景的重要性。 值得注意的是，这个特征在增加T时会更显著。在少量训练数据的情况下，用户很少有共同的项目来表明它们的相似性。

• GRank在ML-100K中将EigenRank提高约1％-4％，而其性能比ML-1M中的EigenRank高出8％。 这一结果可以用Kendall相关度量与稀疏数据的自然问题来解释。 在固定的T，ML-1M中的项目数是ML-100K的两倍，因此是稀疏的两倍。因此，EigenRank在ML-1M中有效计算相似性时面临更多困难，因此错误的用户可能被选中 EigenRank作为目标用户的邻居。 另一方面，不依赖于用户之间的直接相似度计算的GRank可以很好地处理这种稀疏数据集。

**表2.在每种评估条件下对于配对t检验获得的P值。 P <0.01表示GRank w.r.t区别于其他算法的显着性能**

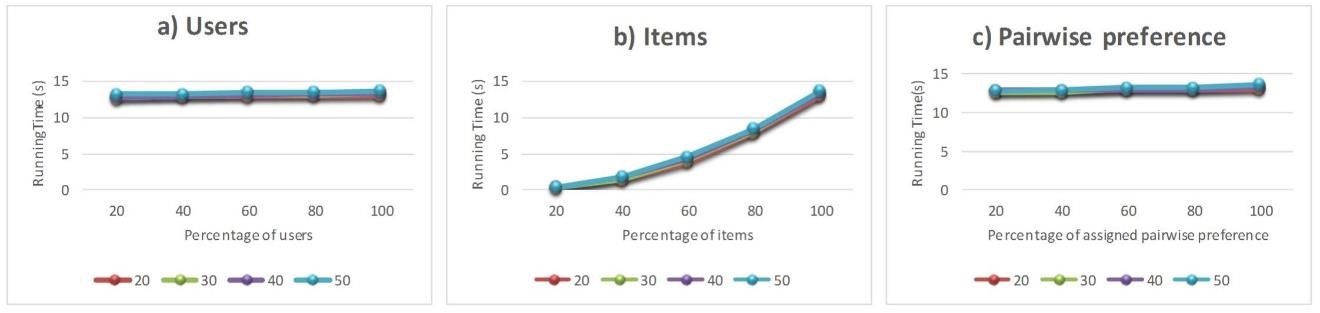
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | ML-100K | | | | | ML-1m | | |  |  |
| Ta | Kb | Eigen Rank | BGR | WBGR | Cofi-  NDCG | Cofi-  Ordinal | Eigen Rank | BGR | WBGR | Cofi-  NDCG | Cofi-  Ordinal |
| 20 | 1 | 0.5396 | 0.038 | 0.3653 | 0.0003 | 0.0020 | 0.8310 | 0.7465 | 0.5417 | 0.0959 | 0.0006 |
| 30 | 1 | 0.6952 | 0.0040 | 0.0214 | 0.0002 | 0.0012 | 0.0100 | 0.0095 | 0.0071 | 0.0035 | 0.0000 |
| 40 | 1 | 0.0131 | 0.0040 | 0.0143 | 0.0002 | 0.0001 | 0.0002 | 0.0000 | 0.0003 | 0.0005 | 0.0007 |
| 50 | 1 | 0.0112 | 0.0040 | 0.0072 | 0.0000 | 0.0000 | 0.0051 | 0.0013 | 0.0015 | 0.001 | 0.0009 |
| 20 | 3 | 0.2577 | 0.0020 | 0.3033 | 0.0044 | 0.0005 | 0.2033 | 0.5991 | 0.3539 | 0.0481 | 0.0001 |
| 30 | 3 | 0.2908 | 0.0003 | 0.0041 | 0.0000 | 0.0007 | 0.0008 | 0.0007 | 0.0064 | 0.0073 | 0.0000 |
| 40 | 3 | 0.0069 | 0.0010 | 0.0064 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0001 | 0.0003 | 0.0007 |
| 50 | 3 | 0.0014 | 0.0002 | 0.0009 | 0.0000 | 0.0000 | 0.0015 | 0.0006 | 0.0007 | 0.0008 | 0.0010 |
| 20 | 5 | 0.9047 | 0.0003 | 0.1455 | 0.0099 | 0.0001 | 0.2910 | 0.5021 | 0.2865 | 0.0321 | 0.0003 |
| 30 | 5 | 0.1842 | 0.0000 | 0.0014 | 0.0000 | 0.0004 | 0.0021 | 0.0002 | 0.002 | 0.0039 | 0.0000 |
| 40 | 5 | 0.0015 | 0.0007 | 0.0051 | 0.0000 | 0.0000 | 0.0000 | 0.0004 | 0.0002 | 0.0003 | 0.0007 |
| 50 | 5 | 0.0060 | 0.0000 | 0.0005 | 0.0000 | 0.0000 | 0.0007 | 0.001 | 0.0008 | 0.0005 | 0.0009 |
| 20 | 10 | 0.3772 | 0.0009 | 0.1446 | 0.0139 | 0.0000 | 0.5587 | 0.6757 | 0.2618 | 0.2020 | 0.0002 |
| 30 | 10 | 0.0070 | 0.0000 | 0.0007 | 0.0000 | 0.0002 | 0.0063 | 0.0000 | 0.0039 | 0.0069 | 0.0000 |
| 40 | 10 | 0.0005 | 0.0000 | 0.0010 | 0.0000 | 0.0000 | 0.0000 | 0.0003 | 0.0000 | 0.0003 | 0.0005 |
| 50 | 10 | 0.0046 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0006 | 0.0004 | 0.0004 | 0.0005 |

#### 4.2.2可扩展性

最后的实验研究了GRank的可扩展性，如3.4节所述，它取决于三个因素：用户数量（𝑀），项目数量（𝑁）和分配的偏好数量（𝑆）。 为了研究GRank在每个评估条件下的可扩展性，即训练集中每个用户可用的评级数（参见第4.1.2节），我们通过在修复其他两个参数的同时改变一个因子来测量运行时间。 例如，为了评估用户数量对算法运行时间的影响，我们修复了训练数据集中的项目数和首选项数，同时通过随机选择20％到100％来改变用户数量。 可用的用户。 然后，我们计算每个推荐的平均运行时间。 已经遵循类似的步骤来评估成对偏好的数量和项目的数量对运行时间的影响。 结果如图9所示。

如图9所示，当改变𝑀和𝑆时，计算复杂度几乎是恒定的，但是当项目数量增加时，运行时间有二次上升。

请注意，GRank的计算复杂度是项目数量的二次方，因为它考虑了所有可能的成对比较。 然而，在许多实际应用中，成对比较的实际数量远小于𝑁2，因为项目通常形成聚类，并且比较通常发生在这些聚类中。 这意味着在实践中，我们可以通过省略任何用户尚未完成的比较来修剪TPG。



**图9. GRank在用户分配的用户数，项数和成对偏好方面的可扩展性分析**

5.讨论

引入FRank框架来解决NCR相似度计算技术的稀疏性问题。 在下文中，我们将简要讨论如何完成FRank以解决问题。

• 使用TPG，GRank隐含地聚合不同类型的用户的相似性：一种类型的用户相似性是基于它们的共同比较来计算的。 这种类型的相似性反映在通过成对比较节点连接两个用户的<𝑈 - 𝑃 - 𝑈>之后的路径上。 另外，如果两个用户在一些比较中都优选特定项目A，则假设两个用户是相似的，即使这两个用户对其优先选择的项目是不同的。 对于两个用户比某个特定项目A更喜欢不同项目的情况也是如此。这些类型的相似性可以通过跟踪<𝑈 - 𝑃 - 𝐼𝑑 - 𝑃 - 𝑈>和<𝑈 - 𝑃 - form形式的路径来发现。 - 𝑃 - 𝑈>，分别。 这些元路径的组合和复制定义了实体之间的许多不同关系，并且GRank捕获并在其等级计算过程中聚合它们。

• 遵循TPG中的元路径<𝑃 - 𝑈 - 𝑃>，GRank隐式地找到相似偏好，这是相似用户的偏好。相关偏好通过<𝑃 - 𝑈 - 𝑃>之后的路径高度连接。因此，通过将目标用户的已知偏好传播到与它们相关的那些未知偏好来迭代地估计未知偏好。即使在没有共同的成对比较的情况下，该信息也可用于计算用户的相似性。我们参考一个例子来阐明这个概念：如图3所示，𝑢1，𝑢2和𝑢5，每个都有一个链接到偏好节点B <A，B <D和C <D。因此，根据肯德尔相关性，它们中的每对之间的相似性将为零。然而，TPG反映出{𝐴>𝐵}与偏好节点{𝐶<𝐷}高度相关，而与B <D无关。因此，𝑢1更类似于𝑢5，因为遵循从{𝐴>𝐵}和{𝐶>𝐷}路径传递的路径。

• 利用TPG和PageRank算法，GRank直接估算用户对看不见项目的排名。 TPG中的个性化PageRank计算使GRank能够汇总从不同形式的元路径<𝑢，𝑣1，...，𝑣𝑚，𝑖𝑑>和<𝑢，𝑣1，...，𝑣𝑚，𝑖𝑢>获得的排名信息，以便快速准确地预测用户 ' 排行。 注意，该方法不同于在所有基于邻居的方法中使用的典型的3步框架（相似度的计算，偏好矩阵的生成以及总排名的推断）。

#### 6.结论

在本文中，我们研究了如何设计和利用基于图形的框架来解决当前基于邻居的协作排序算法的缺点。为此，我们建议将偏好数据建模为一个新的三分图结构然后进行探索可以帮助我们捕捉排名偏好数据集中存在的不同类型的关系（例如用户的相似性，项目的相似性等）。我们还提出了一种随机游走方法，根据提出的结构进行推荐。实验结果显示，建议的框架，GRank相对于其他最先进的基于图形和基于邻居的协作排序方法有显着改善。似乎GRank的基于图的方法在稀疏和密集数据集中都是有益的。在密集数据集中，它可以通过探索实体之间存在的不同路径来更精确地形成邻域。在稀疏数据集中，用户很少进行共同的成对比较，直接邻域通常非常小，它仍然可以遍历边缘以找到更远的邻居并使用它们的信息进行推荐。建议的图形结构主要用于查找用户和项目之间的紧密程度，但它也可以用于其他目的，例如查找类似用户和类似项目的集群，以及发现相关偏好，这是该领域的一些基本概念推荐系统。

参考文献

Adomavicius, G., & Kwon, Y. (2012). Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. *IEEE Transactions On Knowledge And Data Engineering*, *24*(5), 896–911.

AvSegal, I., Katzir, Z., & Gal, K. (2014). EduRank : A Collaborative Filtering Approach to Personalization in E-learning. In *Educational Data Mining 2014* (pp. 68–75).

Balakrishnan, S., & Chopra, S. (2012). Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining. ACM* (pp. 143–152).

Bellogin, A., & Parapar, J. (2012). Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *Proceedings of the sixth ACM conference on Recommender systems* (pp. 213–216).

Cacheda, F., Carneiro, V., Fernández, D., & Formoso, V. (2011). Comparison of collaborative filtering algorithms. *ACM Transactions on the Web*, *5*(1), 1–33. http://doi.org/10.1145/1921591.1921593

Chen, B., Wang, J., Huang, Q., & Mei, T. (2012). Personalized video recommendation through tripartite graph propagation. *ACM Transactions on Multimedia Computing, Communications and Applications*, *10*(4), 1133. http://doi.org/10.1145/2393347.2396401

Chiluka, N., Andrade, N., & Pouwelse, J. (2011). A Link Prediction Approach to Recommendations in Large-Scale User-Generated Content Systems. In *Advances in information retrieval* (pp. 189–200).

Chowdhury, N., Cai, X., & Luo, C. (2015). BoostMF : Boosted Matrix Factorisation. In *Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)* (pp. 3–18). http://doi.org/10.1007/978-3-319-23525-7

Christakopoulou, K., & Banerjee, A. (2015). Collaborative Ranking with a Push at the Top. In *WWW 2015: Proceedings of the 24th International Conference on World Wide Web* (pp. 205–215). http://doi.org/10.1145/2736277.2741678

Cong, J. (2009). RankClus : Integrating Clustering with Ranking for Heterogeneous Information Network Analysis. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM,* (pp. 565–576).

Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook* (pp. 107–144.).

Dhanjal, C., Clémençon, S., & Gaudel, R. (2015). Collaborative filtering with localised ranking. In *Proceedings of Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI’15)* (Vol. 4, pp. 2554–2560).

Fan, C., & Lin, Z. (2013). Collaborative Ranking with Ranking-Based Neighborhood. *Web Technologies and Applications*, 770–781.

Fouss, F., Pirotte, A., Renders, J. M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *19*(3), 355–369. http://doi.org/10.1109/TKDE.2007.46

Guo, G., Zhang, J., & Yorke-Smith, N. (2015). Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *Knowledge-Based Systems*, *74*, 14–27. http://doi.org/10.1016/j.knosys.2014.10.016

Huang, Z., Li, X., & Chen, H. (2005). Link Prediction Approach to Collaborative Filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries* (pp. 141–142).

Jiang, J., Lu, J., Zhang, G., & Long, G. (2011). Scaling-Up Item-Based Collaborative Filtering Recommendation Algorithm Based on Hadoop. *2011 IEEE World Congress on Services*, 490–497. http://doi.org/10.1109/SERVICES.2011.66

Kim, K. jae, & Ahn, H. (2008). A recommender system using GA K-means clustering in an online shopping market. *Expert Systems with Applications*, *34*(2), 1200–1209. http://doi.org/10.1016/j.eswa.2006.12.025

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, *42*(8), 30–37.

Lee, S., Park, S., Kahng, M., & Lee, S. (2013). PathRank : Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems. *Expert Systems with Applications*, *40*, 684–697.

Lerche, L., & Jannach, D. (2014). Using Graded Implicit Feedback for Bayesian Personalized Ranking. In *Proceedings of the 8th ACM Conference on Recommender systems. ACM* (pp. 353–356).

Li, X., & Chen, H. (2013). Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, *54*(2), 880–890. http://doi.org/10.1016/j.dss.2012.09.019

Liu, J., Wu, C., Xiong, Y., & Liu, W. (2014). List-wise probabilistic matrix factorization for recommendation. *Information Sciences*, *278*, 434–447.

Liu, N., & Yang, Q. (2008). EigenRank : A Ranking-Oriented Approach to Collaborative Filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM* (pp. 83–90).

Lu, Q., Chen, T., Zhang, W., Yang, D., & Yu, Y. (2012). Serendipitous personalized ranking for top-N recommendation. *Proceedings - 2012 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2012*, 258–265. http://doi.org/10.1109/WI-IAT.2012.135

Meng, L., Li, J., & Sun, H. (2011). WSRank: A Collaborative Ranking Approach for Web Service Selection. In *2011 IEEE 11th International Conference on Computer and Information Technology* (pp. 103–108). Ieee. http://doi.org/10.1109/CIT.2011.30

Musa, A. B. (2012). Comparative study on classification performance between support vector machine and logistic regression. *International Journal of Machine Learning and Cybernetics*, *4*(1), 13–24. http://doi.org/10.1007/s13042-012-0068-x

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: bringing order to the web., 1–17. http://doi.org/10.1109/ICCKE.2013.6682858

Pan, W., Zhong, H., Xu, C., & Ming, Z. (2015). Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems*, *73*, 173–180. Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-thieme, L. (2009). BPR : Bayesian

Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 452–461). http://doi.org/10.1145/1772690.1772773

Said, A., Kille, B., Jain, B., & Albayrak, S. (2012). Increasing Diversity Through Furthest Neighbor-Based Recommendation. *In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, 1–4.

Sawant, S. (2013). Collaborative Filtering using Weighted BiPartite Graph Projection - A Recommendation System for Yelp. *Proceedings of the CS224W: Social and Information Network Analysis Conference*.

Shang, M. S., Fu, Y., & Chen, D. Bin. (2008). Personal recommendation using weighted bipartite graph projection. In *2008 International Conference on Apperceiving Computing and Intelligence Analysis, ICACIA 2008* (pp. 198–202). http://doi.org/10.1109/ICACIA.2008.4770004

Shang, M.-S., Zhang, Z.-K., Zhou, T., & Zhang, Y.-C. (2010). Collaborative filtering with diffusion-based similarity on tripartite graphs. *Physica A: Statistical Mechanics and Its Applications*, *389*(6), 1259–1264. http://doi.org/10.1016/j.physa.2009.11.041

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. *Recommender Systems Handbook*, 257–298. http://doi.org/10.1007/978-0-38785820-3\_8

Shi, Y., Karatzoglou, A., & Baltrunas, L. (2012). Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems. ACM* (pp. 139–146).

Shi, Y., Karatzoglou, A., Baltrunas, L., & Larson, M. (2013). xCLiMF : Optimizing Expected Reciprocal Rank for Data with Multiple Levels of Relevance. In *Proceedings of the 7th ACM conference on Recommender systems. ACM* (pp. 431– 434).

Shi, Y., Larson, M., & Hanjalic, A. (2010). List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender system* (pp. 269–272).

Shi, Y., Larson, M., & Hanjalic, A. (2013). Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation. *Information Sciences*, *229*, 29– 39.

Silva, A., & Zaki, M. (2013). ProfileRank : Finding Relevant Content and Influential Users based on Information Diffusion. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis. ACM* (pp. 1–9). http://doi.org/10.1145/2501025.2501033

Sun, Y., Han, J., Yan, X., & Yu, P. S. (2011). PathSim : Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. In *VLDB* (pp. 992–1003).

Ting, Y., Yan, C., & Xiang-wei, M. (2013). Personalized Recommendation System Based on Web Log Mining and Weighted Bipartite Graph. *2013 International Conference on Computational and Information Sciences*, 587–590. http://doi.org/10.1109/ICCIS.2013.161

Ting-Peng, L., Hung-Jen, L. A. I., & Yi-Cheng, K. U. (2006). Personalized Content Recommendation and User Satisfaction: Theoretical Synthesis and Empirical Findings. *Journal of Management Information Systems*, *23*(3), 45–70. http://doi.org/10.2753/MIS0742-1222230303

Volkovs, M. N., & Zemel, R. S. (2012). Collaborative Ranking With 17 Parameters. In *Advances in Neural Information Processing Systems* (pp. 2294–2302).

Wang, S., Sun, J., & Gao, B. J. (2014). VSRank : A Novel Framework for Ranking-Based collaborative filtering. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *5*(3), 51.

Weimer, M., & Karatzoglou, A. (2007). Maximum Margin Matrix Factorization for Collaborative Ranking. *Advances in Neural Information Processing Systems*, 1–8.

Weimer, M., Karatzoglou, A., & Smola, A. (2008). Improving maximum margin matrix factorization. *Machine Learning*, *72*(3), 263–276. http://doi.org/10.1007/s10994-0085073-7

Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., & Sun, J. (2010). Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’10* (pp. 723–731). ACM Press. http://doi.org/10.1145/1835804.1835896

Xiao, Z., Che, F., Miao, E., & Lu, M. (2014). Increasing serendipity of recommender system with ranking topic model. *Applied Mathematics and Information Sciences*, *8*(4), 2041–2053. http://doi.org/10.12785/amis/080463

Yang, C., Wei, B., Wu, J., Zhang, Y., & Zhang, L. (2009). CARES: a ranking-oriented CADAL recommender system. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries. ACM* (pp. 203–212).

Yao, W., He, J., Huang, G., Cao, J., & Zhang, Y. (2013). Personalized Recommendation on Multi-Layer context graph. In *Web Information Systems Engineering–WISE 2013* (pp. 135–148).

Yu, X., Ren, X., Sun, Y., & Gu, Q. (2014). Personalized Entity Recommendation : A Heterogeneous Information Network Approach. In *7th ACM international conference on Web search and data mining* (pp. 283–292).

Yu, X., Ren, X., Sun, Y., Sturt, B., Khandelwal, U., Gu, Q.,Han, J. (2013). Recommendation in Heterogeneous Information Networks with Implicit User Feedback. In *Proceedings of the 7th ACM conference on Recommender systems. ACM* (pp. 347–350). Zhang, M., Guo, X., & Chen, G. (2015). Prediction uncertainty in collaborative filtering : Enhancing personalized online product ranking. *Decision Support Systems*, *83*, 10–21. http://doi.org/10.1016/j.dss.2015.12.004

Zhang, Z., & Kulkarni, S. R. (2014). Detection of Shilling Attacks in Recommender Systems via Spectral Clustering. In *Information Fusion (FUSION), 2014 17th International Conference on. IEEE.* (pp. 1–8).

Zhang, Z.-K., Zhou, T., & Zhang, Y.-C. (2010). Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs. *Physica A: Statistical*

*Mechanics and Its Applications*, *389*(1), 179–186. http://doi.org/10.1016/j.physa.2009.08.036

Zhen, Y., Li, W. J., & Yeung, D. Y. (2009). TagiCoFi: Tag Informed Collaborative Filtering. *Proceedings of the Conference on Recommender Systems (RecSys)*, 69–76. http://doi.org/10.1145/1639714.1639727

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., Zhang, Y.-C., & Settings, M. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences of the United States of America*, *107*(10), 4511–5. http://doi.org/10.1073/pnas.1000488107

Zhou, T., Ren, J., Medo, M., & Zhang, Y. C. (2007). Bipartite network projection and personal recommendation. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *76*(4), 1–7. http://doi.org/10.1103/PhysRevE.76.046115

**参考文献原文**

**Graph-based Collaborative Ranking**

Bita Shams a and Saman Haratizadeh a

a University of Tehran, Faculty of New Sciences and Technologies North Kargar Street, Tehran, Iran 1439957131

**Abstract**

Data sparsity, that is a common problem in neighbor-based collaborative filtering domain, usually complicates the process of item recommendation. This problem is more serious in collaborative ranking domain, in which calculating the users’ similarities and recommending items are based on ranking data. Some graph-based approaches have been proposed to address the data sparsity problem, but they suffer from two flaws. First, they fail to correctly model the users’ priorities, and second, they can’t be used when the only available data is a set of ranking instead of rating values.

In this paper, we propose a novel graph-based approach, called GRank, that is designed for collaborative ranking domain. GRank can correctly model users’ priorities in a new tripartite graph structure, and analyze it to directly infer a recommendation list. The experimental results show a significant improvement in recommendation quality compared to the state of the art graph-based recommendation algorithms and other collaborative ranking techniques.

**Keywords:** Collaborative ranking, pairwise preferences, graph modelling, recommendation systems, personalized PageRank

** Introduction**

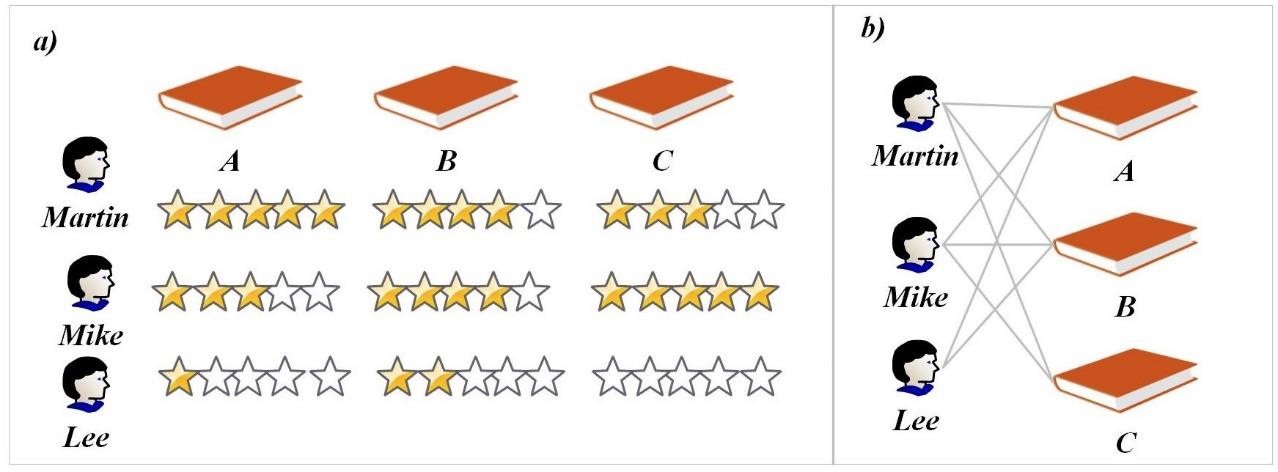
Collaborative filtering (CF) techniques are effective algorithms that help people by filtering irrelevant contents and providing personalized recommendation of useful services. These techniques seek to learn models to predict the services that a user will require in the future based on his preferences in the past.

Collaborative-filtering techniques can be categorized into two classes: rating-oriented and ranking-oriented algorithms. The goal of rating-oriented algorithms is to accurately predict a user’s ratings and then, recommend the items with the highest predicted rating for him. On the other hand, ranking-oriented approach, called collaborative ranking, seek to directly predict the rankings of items from the viewpoint of a target user, without explicitly predict the ratings. It has been shown that ranking-oriented collaborative filtering approach is sometimes more intuitive and applicable. To see why, notice that, recommendation is naturally a ranking task and what a recommendation algorithm really needs is to improve the quality of Top-k ranking not predicting the rates (N. Liu & Yang, 2008; Y Shi, Karatzoglou, & Baltrunas, 2012; Y Shi, Larson, & Hanjalic, 2010; Yue Shi, Larson, & Hanjalic, 2013). Moreover, in many applications, all we have is a set of implicit feedbacks while no rating data is available and hence, rating based methods can’t be used in such a situation. Note that despite rating and other kinds of explicit feedback, that require the user to explicitly assess the items, implicit feedback can be automatically gathered by tracking the user’s interactions with the system (e.g. click, buy, like, etc.). Ranking oriented collaborative filtering can be applied in such situations as well.

Neighbor-based collaborative filtering, one of the main classes of collaborative filtering, estimates the ranking/rating of target user based on the behavior of similar users. Despite several researches in this class of algorithms, they still are not able to precisely calculate users’ similarities. The reason can be explained by *sparsity problem* which refers to the fact that in recommender systems, users have given feedback to a small proportion of items, and consequently, they rarely have enough common items or pairwise comparisons for estimation of their true similarities/ dissimilarities (Desrosiers & Karypis, 2011). One approach to overcome this issue, is graph-based recommendation that takes advantages of heterogeneous information networks, that are information networks containing different types of nodes and edges, to refine the similarity measures (M. S. Shang, Fu, & Chen, 2008; Z.-K. Zhang, Zhou, & Zhang, 2010; Zhou, Ren, Medo, & Zhang, 2007), expand the neighborhoods, and, directly calculate the closeness of users and items(Chiluka, Andrade, & Pouwelse, 2011; Silva & Zaki, 2013; Xiang et al., 2010; Yao, He, Huang, Cao, & Zhang, 2013).

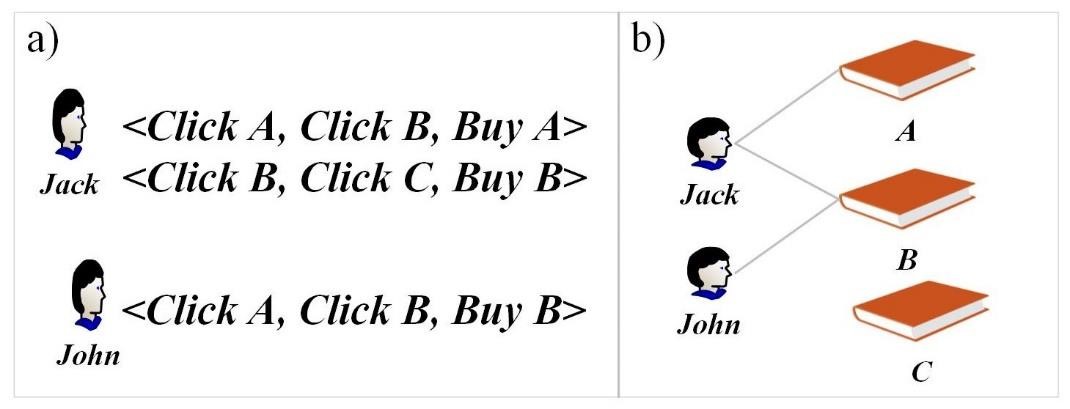
Graph-based recommendation algorithms represent the relations between users and items as a bipartite graph in which there is a weighted or unweighted link between a user and each item he has rated (Li & Chen, 2013; M. S. Shang et al., 2008; M.-S. Shang, Zhang, Zhou, & Zhang, 2010; Ting, Yan, & Xiang-wei, 2013; Xiang et al., 2010; Z.-K. Zhang et al., 2010; Zhou et al., 2007). Unfortunately, this approach is basically designed for rating/binary feedbacks and has crucial insufficiencies for ranking-oriented class of neighbor-based collaborative filtering.

The first problem is that current graph-based approaches are incompetent to capture the preference order of users. We refer to the example of Fig.1a to illustrate this shortcoming. Mike and Lee have the same preference order for item *A,* and *B*, while Mike, and Martin have completely opposite preference orders on all items. Current graph-based algorithms represent this data as Fig.1b (Sawant, 2013; M. S. Shang et al., 2008). Intuitively, under this graph modeling, most of well-known graph proximity measures (e.g. common neighbors, distance, Katz, and personalized PageRank) will suggest that Mike is much closer to Martin than Lee that is counterintuitive.



**Fig.1. An example to illustrate incompetence of current graph-based structure to capture preference data gathered in form of rating.**

The second shortcoming of current graph-based approaches have been proposed for binary implicit feedback and that they cannot capture the pairwise preference (i.e. choice context) of user that is generated by different implicit feedbacks. It is clear that the choice context is a valuable piece of information that can be used to improve the recommendation quality. To see how such information is lost when data is modeled by current graph representations, you can observe in the example of Fig.2 that John has preferred item A over B in one session and item B over C in another session, while, Jack has preferred item *B* over *A*. Current graph-based representation of implicit feedbacks, makes a link between the user and those items receiving the positive feedbacks (Chen, Wang, Huang, & Mei, 2012; Xiang et al., 2010; Z.-K. Zhang et al., 2010). Therefore, these algorithms cannot differentiate heterogeneous implicit feedbacks (i.e. buy, click). More importantly, they are not able to clarify the fact that Jack and John disagree when it comes to comparison of items *A* and *B,* as illustrated in Fig.2b.



**Fig.2. An example to illustrate incompetence of current graph-based structure to capture the choice context collected through the browsing/ purchasing history.**

This paper presents a novel framework, called GRank, that captures the preference of users using a new Tripartite Preference Graph (TPG) structure that demonstrates the relations between users, items, and pairwise preferences. GRank, also provides a new ranking algorithm, which extends personalized PageRank for top-k recommendation. To the best of our knowledge, this algorithm is the first graph-based approach that is able to capture the preference information provided by implicit feedbacks. Experimental results show higher accuracy of GRank compared to the state of the art collaborative ranking algorithms as well as available graph-based recommendation systems.

The rest of this paper is organized as follows. In Section 2, the related work on graphbased recommendation and collaborative ranking techniques are discussed; then, we present the details of GRank’s framework in Section 3. The experimental results are presented and analyzed in Section 4. In Section 5, we discuss how GRank can address some the current shortcomings of collaborative ranking and graph-based recommendation methods. Finally, in Section 6 we conclude and introduce our future works.

** Related Work**

The quality of recommendation can be analyzed from many different points of view including accuracy(Koren, Bell, & Volinsky, 2009; Weimer & Karatzoglou, 2007), coverage (Bellogin & Parapar, 2012; Cacheda, Carneiro, Fernández, & Formoso, 2011), diversity (Adomavicius & Kwon, 2012; Said, Kille, Jain, & Albayrak, 2012; Zhou et al., 2010), serendipity(Lu, Chen, Zhang, Yang, & Yu, 2012; Xiao, Che, Miao, & Lu, 2014), uncertainty (M. Zhang, Guo, & Chen, 2015), shilling attack detection(Z. Zhang & Kulkarni, 2014) and scalability (Jiang, Lu, Zhang, & Long, 2011). Although all these aspects are important factors in the success of a recommender system, the

recommendation’s accuracy is a key element with this regard and a core set of researches have been formed to achieve higher levels of recommendation accuracy. This paper lies in this category, presenting a novel graph-based framework that improves the accuracy of recommendation in the absence of contextual information location when the only available information is the preference data. Contextual information usually refers to the environmental state in which the interaction of the user and the system happens (e.g. time, location, emotion, etc.). On the other hand, the “choice context” reflects the options among which the user makes a choice.

Here, we will review the existing researches related to the main aspects of our proposed algorithm: Collaborative ranking, and Graph-based recommendation.

** . Collaborative ranking**

Collaborative ranking is a class of collaborative filtering algorithms that seeks to predict how a user will rank items. As we mentioned before, despite some similarities, collaborative ranking algorithms are differentiated from rating-oriented collaborative filtering (i.e. collaborative rating) by the fact that collaborative rating algorithms rely on the rating data and try to minimize the rating prediction error, while collaborative ranking algorithms do not depend on rating data. They can use any kind of preference data and try to minimize the rank prediction error.

There is also some similarities between two concepts of collaborative ranking and learning-to-rank problems in the information retrieval domain, as they both try to order entities of on type, e.g. documents/items, for a target entity of another type, e.g. queries/users. However these two kinds of problems are different in practice. In learningto rank problem there exist a set of explicit common features, such as terms’ frequencies among two types of entities, queries and document (Balakrishnan & Chopra, 2012; Fan & Lin, 2013; Y Shi et al., 2010; Volkovs & Zemel, 2012) while there is no such features available or used to relate user and item entities in collaborative ranking problem (Balakrishnan & Chopra, 2012; Y Shi et al., 2010; Volkovs & Zemel, 2012). Because of this important difference between the nature of the problem in these domains, different classes of algorithms have emerged for solving those problems. These approaches can be categorized into two categories: matrix factorization (MFCR) and Neighbor-based algorithms (NCR).

Matrix factorization techniques in collaborative ranking, try to learn representative latent features for an accurate prediction on ranking of items for each user. CofiRank was the first algorithm that uses matrix factorization techniques to optimize a rank-oriented metric(Weimer, Karatzoglou, & Smola, 2008; Weimer & Karatzoglou, 2007). Another technique, ListRank, estimates the Top-1 probabilities to infer a ranking for items. (Y Shi et al., 2010). URM is another model that combines ListRank and probabilistic matrix factorization in order to improve system’s accuracy in terms of both ranking and ratings (Yue Shi, Larson, et al., 2013). BoostMF is another matrix factorization approach that sequentially learns a set of weak matrix factorization models based on preference data(Chowdhury, Cai, & Luo, 2015). Bayesian personalized ranking and its variants, try to optimize area under the curve (AUC) for a Bayesian prediction model that is generated based on a set of prediction of pairwise comparisons between relevant and irrelevant items (Lerche & Jannach, 2014; Pan, Zhong, Xu, & Ming, 2015; Rendle, Freudenthaler, Gantner, & Schmidt-thieme, 2009). Recently, some approaches have been proposed that focus on correctly predicting the pairwise preferences for the items with the highest ranks (Christakopoulou & Banerjee, 2015; Dhanjal, Clémençon, & Gaudel, 2015). Climf (Y

Shi et al., 2012) and xClimf (Yue Shi, Karatzoglou, Baltrunas, & Larson, 2013) are two other algorithms that exploit matrix factorization techniques to optimize Mean Reciprocal Rank (MRR) of the recommendation list (Y Shi et al., 2012; Yue Shi, Karatzoglou, et al., 2013).

Although the collaborative ranking methods based on matrix factorization consist of a diverse set of algorithms and methods (Weimer et al., 2008; Weimer & Karatzoglou, 2007), the approach of GRank is conceptually different from them as it does not represent the data in a latent feature space. Instead, it models the rank data in the form of a graph structure that enables it to directly estimate the closeness of users and items, based on which it can do the recommendation. So, in a sense, GRank lies in another class of recommendation algorithms called neighbor-based collaborative ranking (NCR). Although this second class of algorithms has its advantages, this approach has remained less investigated, and few successful NCR algorithms have been proposed so far.

EigenRank(N. Liu & Yang, 2008) is the most famous NCR technique that infers a total ranking based on pairwise preferences of users similar to the target user. EigenRank computes users’ similarity using Kendall correlation that takes into accounts the agreement and disagreement of users over pairwise comparisons. After estimation of similarities, EigenRank estimates a preference matrix whose elements are a weighted linear combination of neighbors’ preferences. Finally, it uses a greedy or Markov-based approach to infer a total ranking over items. To Our knowledge, all of NCR techniques follow the main approach presented by EigenRank with slight modifications. EduRank(AvSegal, Katzir, & Gal, 2014), WSRank (Meng, Li, & Sun, 2011), and Cares(Yang, Wei, Wu, Zhang, & Zhang, 2009) customized EigenRank for different applications. VSRank (Wang, Sun, & Gao, 2014) focuses to improve Kendall similarity measure via considering importance of each pairwise comparison in similarity calculation. However, this approach still suffers from the *sparsity* problem since it still relies on common pairwise comparisons for calculating similarities.

As stated earlier, GRank aims to solve the sparsity problem of neighbor-based collaborative ranking by introducing a novel graph-based approach for modeling and analyzing data. It also differs from the current neighbor-based algorithms as it does not follow the traditional three-step framework, and directly estimates the users’ preferences.

**2.2. Graph-based recommendation approaches**

Although there is no graph-based methods designed for collaborative ranking, many recent studies have been conducted in other areas of recommender systems. Here, we will briefly review those algorithms and clarify the main differences between the current work and them.

Graph-based recommendation algorithms are composed of two steps: Constructing a graph representing the data and making recommendations by analyzing the graph. These recommendation algorithms have exploited different types of graphs. However, in all of them, the main component of the graph is the relations between users and those items that have been rated by them. Therefore, the most common approach is constructing a

bipartite network where the connections are from one part of the network, users, to the other part, items. Once the bi-partite graph is constructed, several approaches can be used to rank the items using the information from the neighbors of the target user. Approaches like using common neighbors, Katz similarity, diffusion scores and personalized PageRank have been used in this domain (Huang, Li, & Chen, 2005; Z.-K. Zhang et al.,

2010)

Recent methods have extended the bi-partite network by adding some layers to it. Some researchers (Xiang et al., 2010) have considered using a session layer to take into account the long-term and the short-term preferences of the user in order to make recommendations in a particular time. Others (Yao et al., 2013) have used different types of nodes in a multi-layer structure to make context-aware recommendation through a random walk in the graph. In (Z.-K. Zhang et al., 2010) a three-layer graph is used to improve recommendation through considering the tags assigned to items by users using a diffusion-based score introduced in (Zhou et al., 2007). In some works (Lee, Park, Kahng, & Lee, 2013; Yu, Ren, Sun, & Gu, 2014) the structure of the network has been revised. They consider a star heterogeneous network, where users and items can be connected to different types of nodes. They use this graph structure to improve the model–based recommendations (Yu et al., 2014) or to make recommendation through improvement of personalized PageRank algorithms in heterogeneous networks. (Lee et al., 2013). We emphasize that none of these algorithms are designed to capture the choice context and preferences of users. Also most of them depend on the contextual information (e.g. time, content, etc.) that does not exist or is not available to the system in all applications and may be expensive to collect.

** GRank: a graph-based framework for collaborative filtering**

It has been shown that heterogeneous information networks have strong capabilities to model the relationships among different entities of recommender systems (Cong, 2009; Sun, Han, Yan, & Yu, 2011; Yu et al., 2013, 2014). In this paper, we seek to propose an effective graph approach to ranking-oriented recommender systems, called Graph-based collaborative Ranking, or GRank. In the following, we first define the problem of graphbased collaborative ranking and its purposes. Then, we present some definitions that are needed to understand the algorithm. Next, we introduce a novel heterogeneous graph structure, called tripartite preference graph (TPG) that embeds different kinds of relations among users, preferences and items in an aggregated structure. Finally, we suggest an efficient algorithm to exploit TPG in order to rank items for each target user.

** . Problem definition**

From the collaborative ranking perspective, recommender systems can be represented by the set of users 𝑈 = {𝑢1, … , 𝑢𝑀}, set of items 𝐼 = {𝑖1, … , 𝑖𝑁}and the observation set 𝑂 = {< 𝑢, 𝑖, 𝑗 >} that is the set of preferences occasionally stated by users. Generally, we define the observation 𝑜 =< u, i, j > where u ∈ U, i ∈ I and j ∈ I denoting that the user *u* has preferred i over j. For simplicity, we call the first item, the desirable item and the second one the undesirable item.

Note that pairwise comparison is a general form of ranking data, and, all kinds of preferences (e.g. rating, browsing history) can be converted to a set of pairwise comparisons using the following rules:

***Rule 1****. Let L be a rating matrix in which* 𝐿𝑢𝑖 *represents the rating of user u for item i. The preference observation set can be obtained by* 𝑂 = {< 𝑢, 𝑖, 𝑗 >|𝐿𝑢𝑖 ≠ 0, 𝐿𝑢𝑗 ≠

0 , 𝐿𝑢𝑖 > 𝐿𝑢𝑗}

***Rule 2.*** *Let L be the matrix of positive feedbacks (e.g. Like) in which a non-zero element* 𝐿𝑢𝑖 *represents that user u likes item i. Similarly, Let D be the matrix of negative feedbacks (e.g. dislike) in which a non-zero element* 𝐷𝑢𝑖 *represents the user u dislikes item i. The preference observation set can be obtained by* 𝑂 = {< 𝑢, 𝑖, 𝑗 >|𝐿𝑢𝑖 ≠ 0, 𝐷𝑢𝑗 ≠ 0 }

***Rule 3.*** *Let W be the set of sessions that are defined as* 𝑊 = {𝑤1, … 𝑤|𝑤|}*, we can create the observation set as*

𝑂 = {< 𝑢, 𝑖, 𝑗 > |∃𝑤 ∈ 𝑊 𝑢 = 𝑤. 𝑢, 𝑖 ∈ 𝑤. 𝐵, 𝑗 ∈ 𝑤. 𝐶}

*Where w.u is the user in the session w, w.B is the set of items bought in the session w and w.C is the set of items clicked but not bought in session w*

Given a preference dataset *O*, a graph-based framework will face a key question that is how to model the information available from the preference data set in the graph structure. To answer this question, we first categorize some information that an effective graph modeling of rank-oriented recommender system ought to capture:

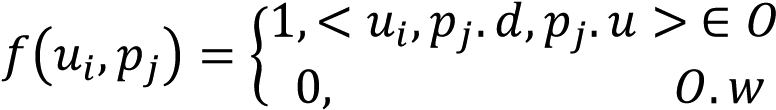
* **Users’ similarities’ in terms of priorities:** Two users of a ranking-oriented recommender system, are assumed to be similar when they have either similar opinions about certain pairwise comparisons(N. Liu & Yang, 2008; Wang et al., 2014) or prefer a particular item *A* in some comparisons even if the items over which it has been preferred are different for those two users (Meng et al., 2011). A well-organized graph model of ranking data should reflect both type of similarities among users.
* **Correlation among comparisons:** Correlated pairwise comparisons are those preferences that are similarly voted by users. These comparisons should be simply discovered by analyzing the graph representation of the data.
* **Items’ similarities:** Similar items are those that are similarly favored/disfavored by a group of similar users. An effective graph modeling should clearly reflect the closeness of these items.
* **Prediction of users’ priorities:** The ultimate goal of ranking-oriented recommender systems is to infer the total ranking of target user over unseen items and recommend the top-k items. Consequently, graph representation of these systems is responsible for the efficient and effective recommendation to the target user, and ideally, a graph representation of rank data may be used to directly predict the rankings.

** . Graph Construction**

Before we can proceed to explain how GRank constructs and exploits a graph structure based on the preference dataset, we need to define some basic concepts:

***Definition. 1****. A pairwise preference p is a tuple* < 𝑖, 𝑗 > *denoting the preference of i over j. We call i as the desirable item in p, represented by p.d, and j as the undesirable item in p represented by p.u. The pairwise preference set P is formally defined as* 𝑃 = {< 𝑖, 𝑗 > |𝑖 ∈ 𝐼, 𝑗 ∈ 𝐼, 𝑖 ≠ 𝑗}.

***Definition. 2****. A user may have some certain preference over two items or not. The agreement function* 𝑓: 𝑈 × 𝑃 → {0,1} *indicates whether the user* 𝑢𝑖 *agrees with the preference* 𝑝𝑗 *or not and is defined as:*



*Where*  𝑢𝑖 ∈ 𝑈, 𝑝𝑗 ∈ 𝑃 𝑎𝑛𝑑 *O is the observation set of preferences, as defined in section 3.1.*

***Definition.3.*** *Abstractly, each item has two sides: the desirable side and the undesirable side. We define the items’ desirability set as* 𝐼𝑑 = {𝑖𝑑|𝑖 ∈ 𝐼} where 𝑖𝑑 represents the desirable side of item *i. Also the items’ undesirability set is defined as* 𝐼𝑢 = {𝑖𝑢|𝑖 ∈ 𝐼} where 𝑖𝑢 represents the undesirable side of item *i. We also define the representative set as* = 𝐼𝑑 ∪ 𝐼𝑢 *, that will contain two elements for each item, one for each side of it.*

***Definition.4.*** *The support function* 𝑠: 𝑃 × 𝑅 → {0,1} *indicates whether a preference p supports the representative r or not. Formally, we define s as*

1, 𝑝. 𝑢 = 𝑖 𝑎𝑛𝑑 𝑟 = 𝑖𝑢

𝑠(𝑝, 𝑟) = {1, 𝑝. 𝑑 = 𝑖 𝑎𝑛𝑑 𝑟 = 𝑖𝑑

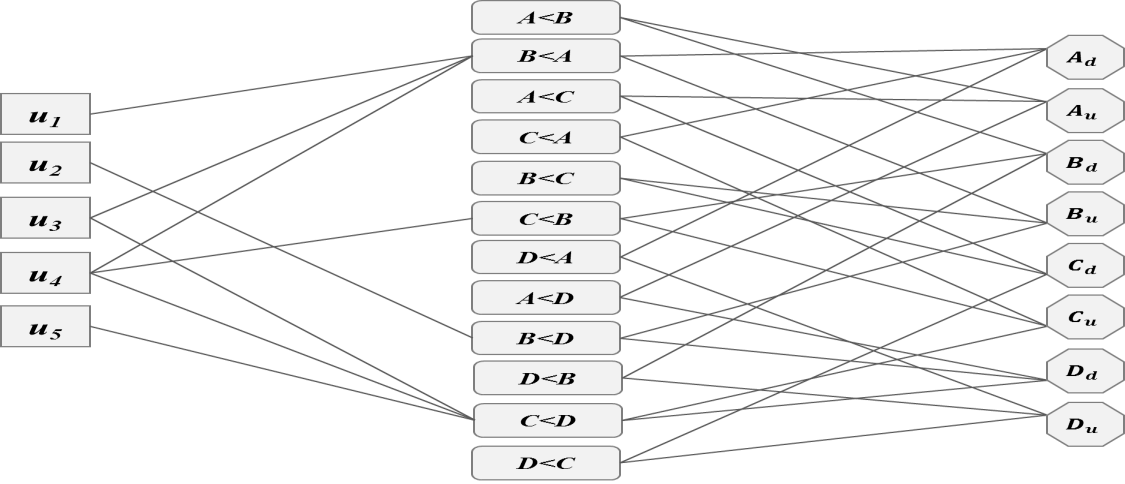
0, 𝑂. 𝑊

*Where* 𝑝 ∈ 𝑃 *and* 𝑟 ∈ 𝑅*.*

Based on these definitions and concepts we can now explain how GRank models the preference data using a structure called Tripartite Preference Graph. Formally, *Tripartite Preference Graph (TPG)* is a tripartite graph TPG(𝑈 ∪ 𝑃 ∪ 𝑅, 𝐸𝑈𝑃 ∪ 𝐸𝑃𝑅), where *U* is the set of users, *P* is the set of pairwise preferences and *R* is the set of representatives. EUP = {(𝑢, 𝑝)|𝑓(𝑢, 𝑝) = 1, 𝑢 ∈ 𝑈, 𝑝 ∈ 𝑃},is the set of edges between the nodes in *U* and *P*, and 𝐸𝑃𝑅 = {(𝑝, 𝑟)|𝑠(𝑝, 𝑟) = 1, 𝑝 ∈ 𝑃, 𝑟 ∈ 𝑅}*,* is the set of edges connecting the nodes in *P* to the nodes in *R*.

More clearly, TPG contains three layers, each containing a different type of nodes:

* **Users:** TPG’s first layer contains one node for each user.
* **Preferences:** The preference layer consist the nodes corresponding to the set of all possible pairwise preferences 𝑝 ∈ 𝑃. Foe simplicity, the corresponding node to each preference 𝑝 =< 𝑖, 𝑗 > is labeled in the form of ("𝑖 > 𝑗") that clearly demonstrates the preference of *i* over *j*.
* **Representatives:** The representative layer contains the set of both undesirable and undesirable representative of items represented by 𝑖𝑑 and 𝑖𝑢, respectively TPG also contains two types of links:
* **User-Preference links:** 𝐸𝑈𝑃 is the set of edges that connects each user *u* to his stated preference. More clearly, For each preference data < 𝑢, 𝐴, 𝐵 >∈ 𝑂, there is a link between user 𝑢 and the preference node labeled by “𝐴 > 𝐵”
* **Preference-representative links:** 𝐸𝑃𝑅 is the set of links that connects each preference to the representatives that it supports. For instance, a preference node labeled by “𝐴 > 𝐵” is connected to the nodes corresponding to desirable case of 𝐴, i.e. 𝐴𝑑 and undesirable case of B i.e. 𝐵𝑢. This links are used to model the fact that an “𝐴 > 𝐵” preference, implicitly supports item desirable side of “𝐴” and undesirable side of “𝐵”.



**Fig. 3. An example: A TPG constructed from a system containing 5 users, 4 items, and, 9 assigned pairwise preference.**

As mentioned before, TPG is a tripartite graph in which the preference layer connects to both other layers: the user layer and the item layer. Traversing TPG through different types of paths reveals different types of information in a ranking-oriented recommender systems, some examples of which are presented in Table.1.

**Table 1. The meta-paths and their semantics in TPG**

|  |  |
| --- | --- |
| **Meta-patha** | **Semantic** |
|  | Users’ similarities in terms of pairwise preference |
| 𝑼 − 𝑷 − 𝑰𝒅 − 𝑷 − 𝑼  𝑼 − 𝑷 − 𝑰𝒖 − 𝑷 − 𝑼 | Agreement of users over desirability/ undesirability of an item over different items. |
| 𝑷 − 𝑼 − 𝑷 | Correlation between pairwise comparisons |
| 𝑰𝒅 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒅  𝑰𝒖 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒖 | Direct relations between items; items that are simultaneously favored/ disfavored by users |
| 𝑰𝒖 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒅  𝑰𝒅 − 𝑷 − 𝑼 − 𝑷 − 𝑰𝒖 | Indirect relationships between items. Items that are contrarily ranked by user. |

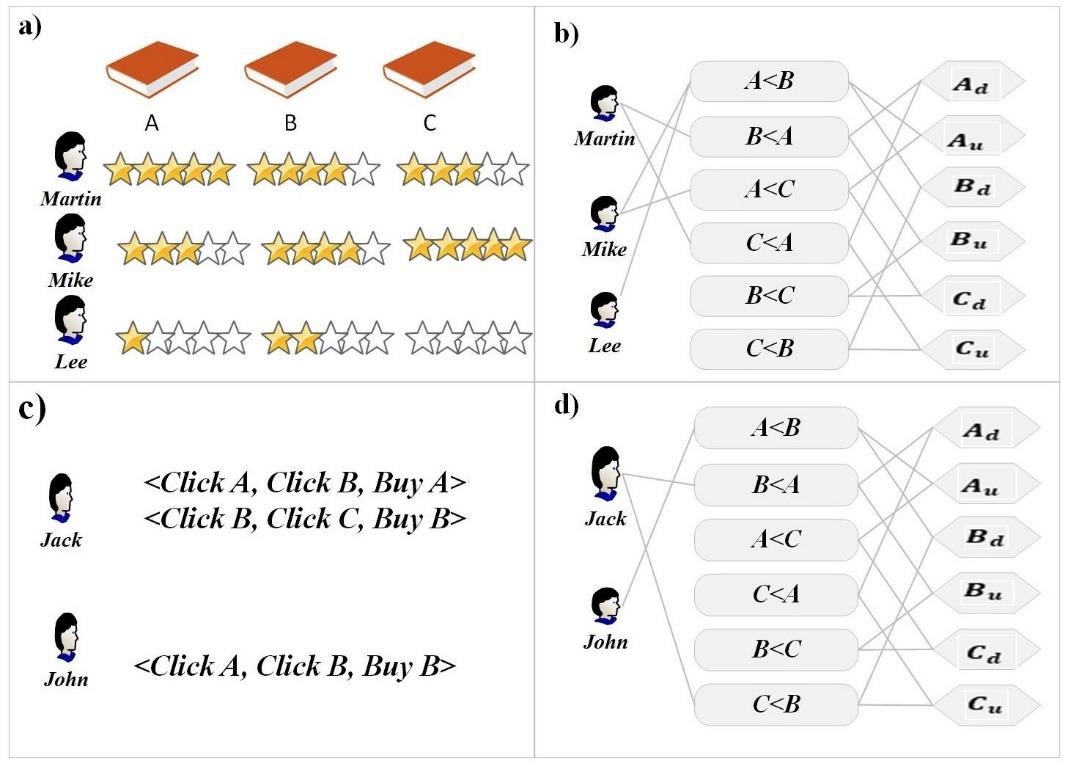
The pseudo code for constructing TPG is presented in Algorithm.1. It first generates 𝑀 nodes corresponding to 𝑀 users and 2𝑁 nodes for desirable and undesirable case of 𝑁 items. Then, it generates 𝑁(𝑁 − 1) nodes for each possible preference data. Each preference data has desirable and an undesirable sides that are represented by *p.d and p.u*, respectively. The next step, is to add links between users and preferences. It scans the preference database and for each < user, item1, item2 > triple, it adds a link between the corresponding nodes in the user and the preference layers. In more details, *getUserNode(u)* returns the corresponding node to user *u while getPreferenceNode(i,j) returns* the node *p* representing the preference of *i* over *j.* After that, a link is created between the node *u* and *p.* Finally, the algorithm scans through the preference nodes in which, a preference node p states that “p.d > p.u”. For such a node, the algorithm finds *the* corresponding representative nodes using *getDesirableNode(p.d)* and *getUnDesirableNode(p.u)* and inserts an edge between *p*  and each of those representatives.

**Algorithm 1. Construction of Tripartite Preference Graph (TPG)**

|  |
| --- |
| **Input:** Set of users U, Set of items I, Observation set of preference (O) **Output:** Tripartite graph (*G*) |
| Initialize a graph G  *//Initializing user layer*  For each item 𝑢 ∈ 𝑈  Create a node 𝑢 in user layer    *//Initializing representative layer*  For each item 𝑖 ∈ 𝐼  Create a desirable-node 𝑖𝑑 in the representative layer  Create an undesirable-node 𝑖𝑢 in representative layer    *//Initializing preference layer*  For each item 𝑖 ∈ 𝐼  For each item 𝑗 ∈ 𝐼  If (𝑖 ≠ 𝑗)  Create a node *p* in preference layer *p.d = i;*  *p.u=j;*    *// Connecting user and preference layer* For each o:(u,i,j) ∈ 𝑂 *u* = G. GetUserNode (u); *p* = G. GetPreferenceNode (i ,j); Connect node *u* to node *p*.    *//Connecting preference and item layer*  For each node v ∈ preference-layer  𝑟𝑑= G.GetDesirableNode (p.d)  𝑟𝑢= G.GetUnDesirableNode (p.u)  Make a link between *p* and 𝑟𝑑 |

Make a link between p and 𝑟𝑢.

**Example.1.** Fig.4 illustrates how TPG can reflect the preference data mentioned in Fig.1 and Fig.2. As shown in Fig.4b, TPG clearly indicates that Mike and Martin (in Fig.1a and Fig.4a) have not the same preference as they neither share any neighbors nor longer paths to each other in TPG. On the other hand, Mike and Lee share one common neighbor that denotes their agreement over comparison of A and B. The same holds for Jack and John in the example of Fig.4c where TPG representation (Fig.4d) can evidently reflect that they do not have the same opinion over the comparison of A and B.Note that current graph-based approaches are not capable to model users’ preference in these samples as mentioned in Fig.1b and Fig.2b.



**Fig.4. TPG representation of examples in illustrated in Fig.1, and, Fig.2, respectively.**

** . Top-k recommendation using TPG graph**

It is a common assumption in recommender systems that users are interested in items that are preferred by their neighbors or are similar to their favorite items. As mentioned before, TPG provides an affluent platform to determine the users’ similarity and items’ relations. GRank exploits the closeness of users to the desirable/undesirable representatives to estimate how much the target user likes/dislikes a specific item. In other words, given a target user *u* and a TPGdescribing the observation set, GRank defines a function 𝐺𝑅: 𝑈 × 𝐼 → ℝ for predicting the goodness of each unseen item i for each user *u based on the closeness of the node u to the* desirable and undesirable representative nodes of item i. Then GRank recommends to u the items with the highest *GR* values calculated for u.

The desirability/undesirability of an item *i* for user *u* can be estimated based on two general types of paths: Desirability and Undesirability paths. Desirability paths are in form of < 𝑢, 𝑣1, 𝑣2, … , 𝑣𝑚, 𝑖𝑑 > and show the closeness of a target user *u* to the desirable case of an item *i*. Similarly, undesirability paths are in form of < 𝑢, 𝑣1, 𝑣2, … , 𝑣𝑚, 𝑖𝑢 > and depict the closeness of the target user *u* and the undesirable case of item *i*.

Intuitively, there are a large number of desirability/undesirability paths between the target user *u* and an item *i.* Therefore, some form of proximity measure is required to compare the number of desirability/undesirability paths between the target user and items.

Recently, some proximity measures have been proposed for analyzing heterogeneous networks. Unfortunately, these methods are heavily dependent to the definition of meta paths and require to get the importance weight of different meta-paths as an input parameter to calculate the proximity among nodes(Lee et al., 2013; Sun et al., 2011). On the other hand, a general approach for measuring nodes’ proximity in a network, is personalized PageRank, or PPR (Page, Brin, Motwani, & Winograd, 1999) that is acknowledged as one of the most effective measures that ranks nodes based on their reachability from a certain set of nodes in a network. It gives high scores to items that are closer to the target user regarding a wide range of graph properties such as distance or number of paths between them (Lee et al., 2013). GRank defines a measure based on PPR to calculate proximities, but before introducing the measure, first we briefly review the concept of PPR.

Formally, the personalized PageRank of a node indicates the probability that a random walker, with a given skewed restarting distribution, will jump to that node. PPR can be considered as a Markov process with restart, and, is defined by Eq.1

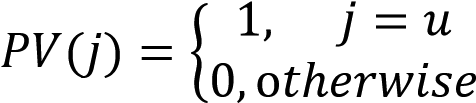
PPR(t) = α · T · PPR(t − 1) + (1 − α) · PV (1)

where PPR(t) denotes the rank vector at the 𝑡 − 𝑡ℎ iteration, 𝑇 is the transition matrix,

𝛼 is the damping factor and “𝑃𝑉” is the user-specific personalized vector. In most of applications 𝛼 is set to 0.85.

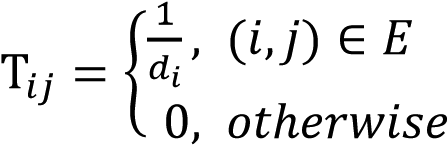
To calculate closeness of some graph nodes, such as item nodes, to a particular node, like the target user’s node, personalized PageRank needs to define a personalized vector

𝑃𝑉 as in Eq.2

 (2)

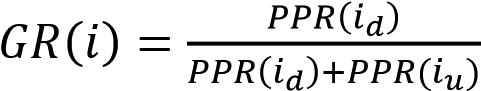
where 𝑢 is the target node.

Given a graph G(V, E) with the set of all nodes *V* and the set of all edges *E*, each element of the transition matrix is obtained from Eq.3

 (3)

Where 𝑑𝑖 is the degree of the 𝑖 − 𝑡ℎ node.

As aforementioned, PPR can be used to find the closeness of each node to the target user’s node in TPG. More clearly, personalized PageRank of the target users in TPG, estimates the probability that a random walker, starting from the target user, will follow a path to the desirable and undesirable representative of each item. GRank defines the goodness of an item *i* for the target user based on PPR of the desirable and undesirable cases of *i*, as in Eq. 4

 (4)

One can expect that 𝐺𝑅(𝑖) gives top scores to those items for which personalized PageRank of their desirable case is much higher than that of their undesirable case. To make a recommendation to the target user, items are sorted according to their GR values, and the top-k items are suggested.

Algorithm 2 summarizes the GRank’s approach for top-k recommendation. GRank requires to calculate the personalized PageRank of nodes for each target user *u.* Forthis purpose, it first defines the transition matrix of TPG and the personalization vector for u, using Eq.3 and Eq.2, respectively. Then, GRank randomly initializes the PPR values and then normalizes them to their summation. Next, it updates PPR value using Eq.1 and iterates until convergence. After that, GR values are calculated for each item using Eq.4. Finally, the items are sorted based on their GR values and the top-k items are recommended to the target user *u*

**Algorithm 2. Top-K recommendation on TPG**

|  |
| --- |
| **Input:** Tripartite graph (*G*), Target user *u, number of recommended items K, set of Items I* **Output:** The best *k items.* |
|  |
| Initialize transition matrix *T* through Eq.3  Initialize personalized vector *PV* through Eq.2  Randomly initialize  t=1  Repeat until convergence    For each item  Calculate GR value for items through Eq.4  Put items in descending order of their GR values in list L  Return as the recommendation list the first k items in L |

** . Computational Complexity**

GRank is composed of two phases: Graph construction and recommendation task. Assume that the  is the number of users, is the number of items andis the number of total pairwise preference assigned by all users. Clearly to total number of possible pairwise comparison would be .

TPG contains  vertices and  edges. The time

complexity of constructing the graph would depend on the implementation approach. If we use adjacency lists, then it would have a time complexity of 

*.* . Additionally, we often know that  where c is a small

constant (e.g. 2.48 for MovieLense100K). So, the time needed for graph construction phase is in O(.

The time complexity of recommendation task is equal to that of personalized

PageRank (PPR) computation. Computational complexity of personalized PageRank is  where , is the number of graphs’ edges and is the number of iterations needed before personalized PageRank converges. In TPG, we have edges, and as we mentioned, we expect that , for a small constant *c*. So the time complexity of a recommendation in GRank, is expected to be . Note that  is a small number for sparse graphs such as TPG. In our experiments, it does not exceed 20

Note that recommendation by GRank has a better computational complexity than

EigenRank, the most acknowledged memory-based CR. EigenRank’s computational

complexity is in the order of  where  is for calculating the

similarity between the target user and all other users,  is for estimating the preference matrix, and,  for inferring the total ranking.

** Experimental settings and results**

We have conducted a series of experiments for evaluating GRank algorithm. Here, we will first give a detailed description of the experimental protocol. Then, we will analyze the ranking quality and scalability of GRank.

** . Experimental setting**

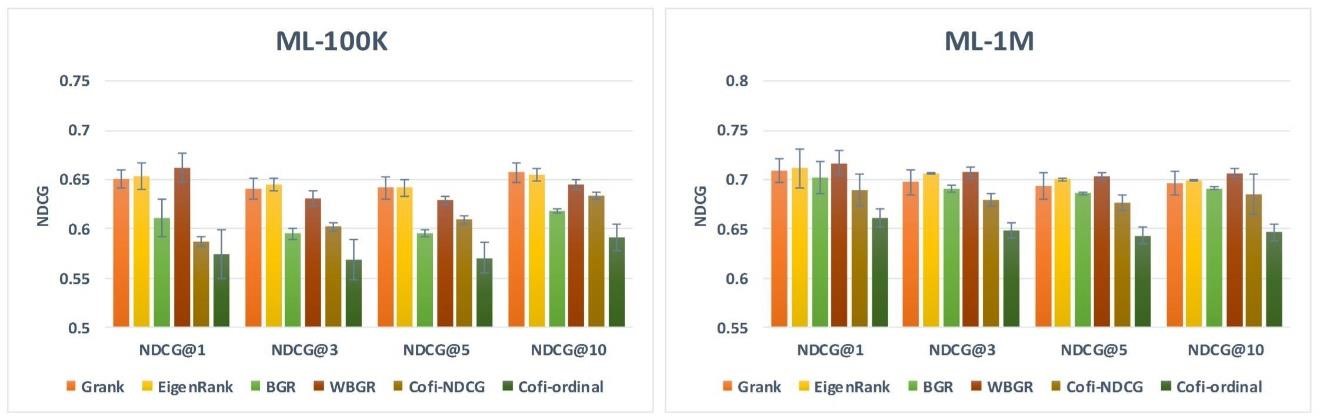
**4.1.1 Dataset**

We conducted all experiments on two publicly available datasets that are widely used in related work (Fan & Lin, 2013; Rendle et al., 2009; Yue Shi, Larson, et al., 2013; Volkovs & Zemel, 2012; Wang et al., 2014): Both data sets have been generated by Movielens group , but contains different number of users, items, and, ratings. The first data set, Movielens-100K, consists of 100,000 ratings (scale 1-5) assigned by 943 users to a collection of 1,682 movies. The second dataset, Movielens-1M, is composed of 3,952 movies rated by 6, 040 users. There are one million ratings in this dataset. Since GRank is designed for using pairwise preference data, we converted the rating information into a set of pairwise comparisons using Rule 1: we created a preference instance of data <u, item#1, item#2) if *item#1* has been rated higher than *item#2* by user u.

**4.1.2 Evaluation methodology**

In our experiments we followed a standard protocol widely used in related work

(Balakrishnan & Chopra, 2012; Fan & Lin, 2013; J. Liu, Wu, Xiong, & Liu, 2014; Rendle et al., 2009; Y Shi et al., 2012; Volkovs & Zemel, 2012). We analyzed the effectiveness of our algorithm under different conditions of user profiles, regarding to the number of user’s ratings; for each user, a fixed number 𝑇 of ratings was randomly sampled and placed in the training set, and the remaining ratings went to the test set. Our experiments involve 𝑇 = 20, 30, 40, 50 items. For each 𝑇, we make sure that we can compare algorithms on at least 10 rated items per user in the test set. Therefore, the users with respectively less than 30, 40, 50, and, 60 items are dropped from both train and test sets. We generated 5 variants of both data sets via random sampling and the average performance on all variants of the test set is reported.

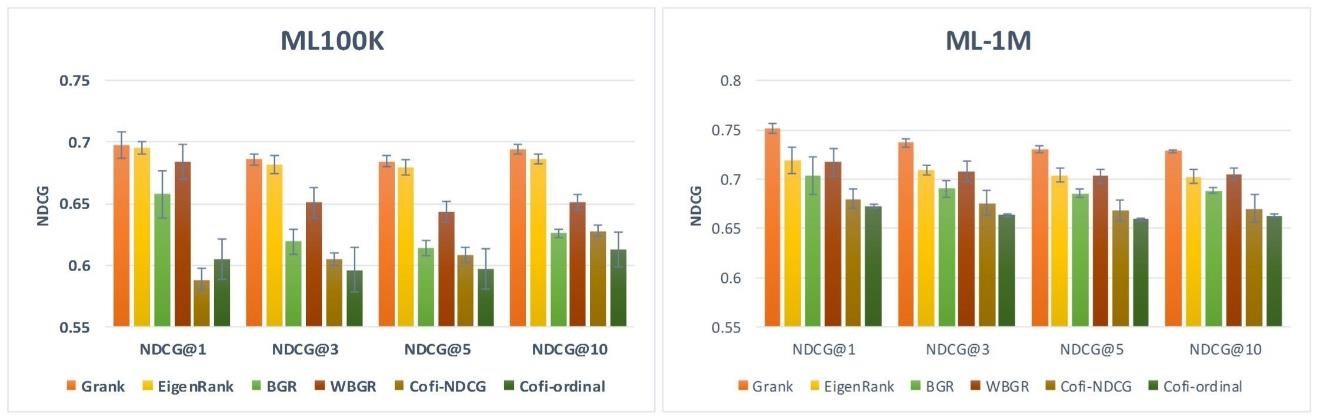


**Fig. 5. Performance comparison of algorithms in terms of NDCG where T=20.**

**4.1.3 Baseline algorithms**

Since GRank makes a connection between neighbor-based collaborative ranking and graph-based recommendation, we have compared its performance against state of the art approaches of these two classes of methods. For further analysis, we also compare our algorithm to Cofi-Rank, the state of the art matrix factorization technique that is able to do recommendation based on pairwise preference data. These algorithms are briefly described in the following:

* **CofiRank:** CofiRank (Weimer & Karatzoglou, 2007) is one of the state of the art MFCR techniques that extracts latent representations in order to optimize a structured loss function. CofiRank has several extensions. In our experiments, we used CofiRank-Ordinal and CofiRank-NDCG as our baseline algorithms. (Weimer et al., 2008) . CofiRank-Ordinal minimizes the number of discordant preferences in the predicted ranking list while CofiRank-NDCG triers to maximize the NDCG. We used the publicly available code for CofiRank and adopted the optimal values for its parameters as suggested in (Weimer et al., 2008).
* **EigenRank:** EigenRank(N. Liu & Yang, 2008) is another famous algorithm in the family of NCR techniques. We have implemented the random-walk version of EigenRank using neighborhood sizes of 100 and 𝜀 = 0.85, that have been reported to be the best parameter values for the algorithm (N. Liu & Yang, 2008).
* **Graph-based recommendation:** Also, we compare GRank with a graph-based recommendation algorithm that exploits a bi-partite graph structure to model the user-item interactions (such as Fig.1). Then, a random-walk with restart is used to rank items in the bi-partite graph(Chiluka et al., 2011). We have implemented two versions of this algorithm, abbreviated by BGR and WBGR, which links the users to those items rated by him with un-weighted and weighted links, respectively. In the weighted version, the weight of the edge between user *u* and item *i* is set equal to the rating of *u* to *i.*

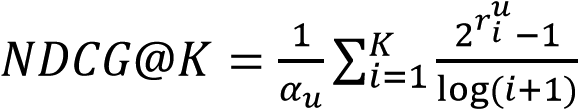


**Fig. 6. Performance comparison of algorithms in terms of NDCG where T=30**  **. Results**

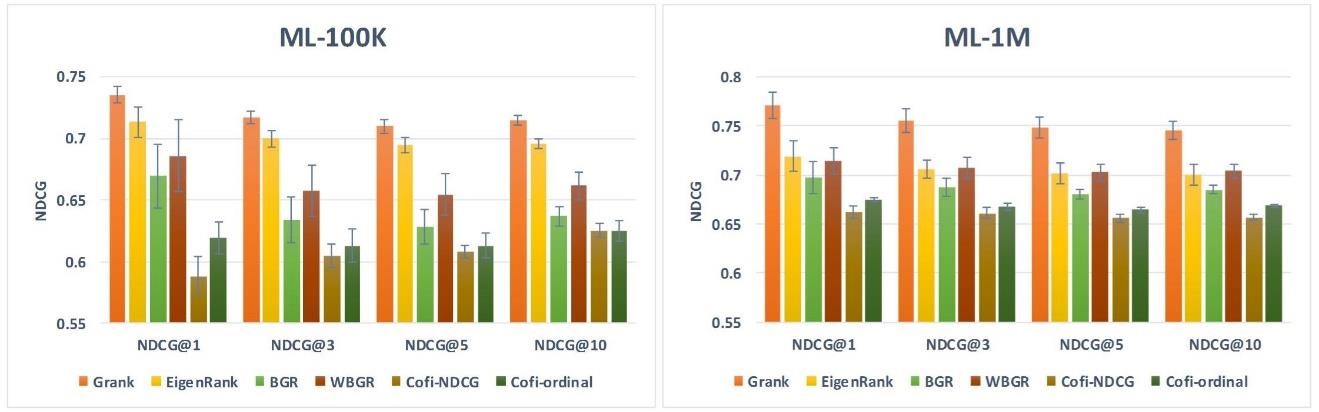
**4.2.1 Accuracy**

Following the standard evaluation strategy applied to recommender systems, we assessed the recommendation performance of the models by comparing the quality of their top-k suggestions.

Normalized Discounted Cumulative Gain (NDCG) is an evaluation metric that is widely used for assessment of CR techniques. The definition of NDCG at the top-K suggestions for a user 𝑢 can be given as:

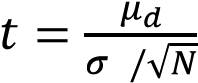
 (5)

Where 𝐾 is the length of the recommendation list,  is the rating given by user *u* to the *i-th* item in the recommendation list, and 𝛼𝑢 is the normalization factor to ensure that NDCG of the ideal recommendation for 𝑢 is equal to 1. In this paper, we report the recommendation performance by NDCG@3, NDCG@5 and NDCG@10, averaged across all users.

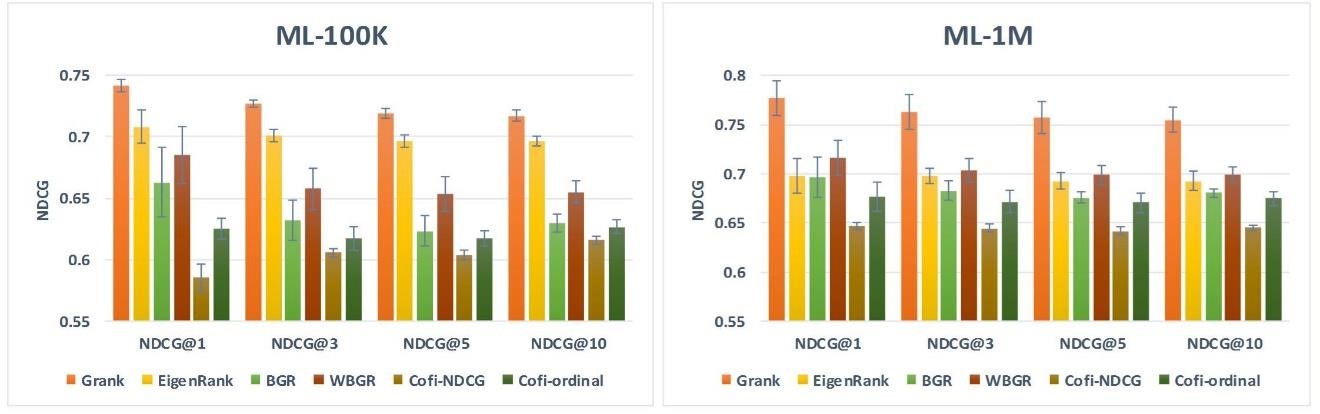


**Fig 7. Performance comparison of algorithms in terms of NDCG where T=40.**

The performance of the algorithms on ML-100k and ML-1M are shown in Fig.5 Fig.6, Fig.7, and, Fig.8. In most of experiments, GRank outperforms other algorithms. The results are reported based on a set of experiments on sufficiently large samples of data, and different algorithms are tested on common sets of samples. So, to test if the differences between the performance of GRank and other algorithms are significant, we can conduct a set of paired t-tests on the results. (Fouss, Pirotte, Renders, & Saerens, 2007; Guo, Zhang, & Yorke-Smith, 2015; Kim & Ahn, 2008; Shani & Gunawardana, 2011; Ting-Peng, Hung-Jen, & Yi-Cheng, 2006; Zhen et al., 2009).

Given {𝑥1, … , 𝑥𝑁} be the set of samples derived from the dataset, let 𝜇𝑑 and 𝜎𝑑 as the average and variance of differences 𝑑𝑖 = 𝑎𝑖 − 𝑏𝑖 where 𝑎𝑖 and 𝑏𝑖 denote the performance of two approaches *A* and *B* on the *i-th* sample. To determine whether A significantly outperforms *B* or not, we consider the null hypothesis is 𝜇𝑑 = 0 whereas the alternative hypothesis is 𝜇𝑑 ≠ 0. The null hypothesis is rejected in favor of the alternative hypothesis if the p-value, obtained by the t-statistic , is below than the

𝑑 significance threshold (e.g. 0.01). Table 2 shows the p-values indicating the significant outperformance of GRank w.r.t other algorithms.



**Fig 8. Performance comparison of algorithms in terms of NDCG where T=50.**

The experimental results can be summarized as below:

* GRank significantly outperforms all algorithms in the majority of evaluation conditions. Yet, EigenRank and WBGR show an improvement of less than 1% in ML-100k and Ml-1M where T=20.
* The performance of GRank is up to 6% and 8% better than WBGR and BGR, respectively. This result reemphasizes the importance of capturing users’ preference and their choice context for recommendation. It is worth noting that this feature will be more renowned while increasing T. In case of small number of training data, users rarely have common items that indicate their similarity.
* GRank improves EigenRank about 1%-4% in ML-100K, while, its performance is up to 8% better than EigenRank in ML-1M. This results can be explained by the natural problem of Kendall correlation measure with sparse data. At fixed *T*, the number of items in ML-1M is two times more than ML-100K, and consequently is two times sparser, Therefore, EigenRank faces more difficulties to effectively calculate similarities in ML-1M and so wrong users may be picked by EigenRank as neighbors of a target user. On the other hand, GRank that does not rely on direct similarity calculation among users, can handle such sparse data sets quite well.

**Table 2. P-values obtained for the paired t-test under each evaluation condition. P<0.01 indicates the significant out performance of GRank w.r.t other algorithm**

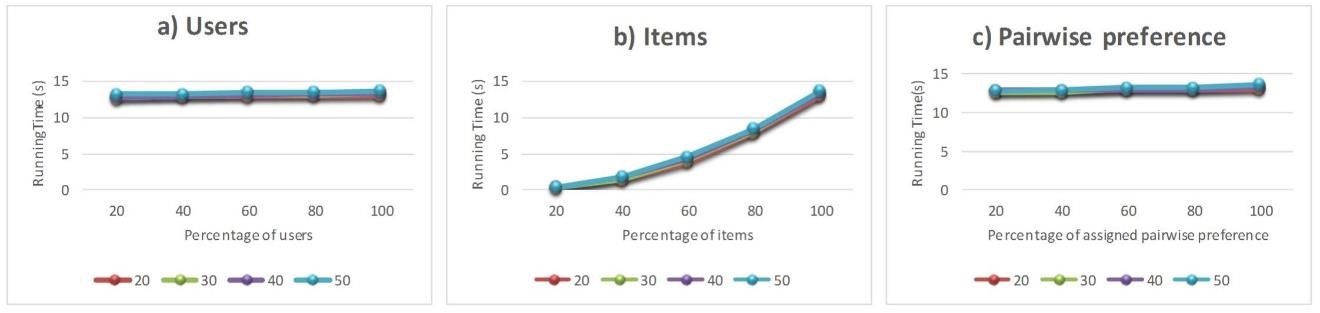
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **ML-100K** | | | | | **ML-1m** | | |  |  |
| Ta | Kb | Eigen Rank | BGR | WBGR | Cofi-  NDCG | Cofi-  Ordinal | Eigen Rank | BGR | WBGR | Cofi-  NDCG | Cofi-  Ordinal |
| 20 | 1 | 0.5396 | 0.038 | 0.3653 | 0.0003 | 0.0020 | 0.8310 | 0.7465 | 0.5417 | 0.0959 | 0.0006 |
| 30 | 1 | 0.6952 | 0.0040 | 0.0214 | 0.0002 | 0.0012 | 0.0100 | 0.0095 | 0.0071 | 0.0035 | 0.0000 |
| 40 | 1 | 0.0131 | 0.0040 | 0.0143 | 0.0002 | 0.0001 | 0.0002 | 0.0000 | 0.0003 | 0.0005 | 0.0007 |
| 50 | 1 | 0.0112 | 0.0040 | 0.0072 | 0.0000 | 0.0000 | 0.0051 | 0.0013 | 0.0015 | 0.001 | 0.0009 |
| 20 | 3 | 0.2577 | 0.0020 | 0.3033 | 0.0044 | 0.0005 | 0.2033 | 0.5991 | 0.3539 | 0.0481 | 0.0001 |
| 30 | 3 | 0.2908 | 0.0003 | 0.0041 | 0.0000 | 0.0007 | 0.0008 | 0.0007 | 0.0064 | 0.0073 | 0.0000 |
| 40 | 3 | 0.0069 | 0.0010 | 0.0064 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0001 | 0.0003 | 0.0007 |
| 50 | 3 | 0.0014 | 0.0002 | 0.0009 | 0.0000 | 0.0000 | 0.0015 | 0.0006 | 0.0007 | 0.0008 | 0.0010 |
| 20 | 5 | 0.9047 | 0.0003 | 0.1455 | 0.0099 | 0.0001 | 0.2910 | 0.5021 | 0.2865 | 0.0321 | 0.0003 |
| 30 | 5 | 0.1842 | 0.0000 | 0.0014 | 0.0000 | 0.0004 | 0.0021 | 0.0002 | 0.002 | 0.0039 | 0.0000 |
| 40 | 5 | 0.0015 | 0.0007 | 0.0051 | 0.0000 | 0.0000 | 0.0000 | 0.0004 | 0.0002 | 0.0003 | 0.0007 |
| 50 | 5 | 0.0060 | 0.0000 | 0.0005 | 0.0000 | 0.0000 | 0.0007 | 0.001 | 0.0008 | 0.0005 | 0.0009 |
| 20 | 10 | 0.3772 | 0.0009 | 0.1446 | 0.0139 | 0.0000 | 0.5587 | 0.6757 | 0.2618 | 0.2020 | 0.0002 |
| 30 | 10 | 0.0070 | 0.0000 | 0.0007 | 0.0000 | 0.0002 | 0.0063 | 0.0000 | 0.0039 | 0.0069 | 0.0000 |
| **40** | 10 | 0.0005 | 0.0000 | 0.0010 | 0.0000 | 0.0000 | 0.0000 | 0.0003 | 0.0000 | 0.0003 | 0.0005 |
| **50** | 10 | 0.0046 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0006 | 0.0004 | 0.0004 | 0.0005 |

**4.2.2 Scalability**

The final experiment investigates the scalability of GRank that, as mentioned in section 3.4, depends on three factors: number of users (𝑀), number of items (𝑁), and number of assigned preferences (𝑆). To investigate the scalability of GRank under each evaluation condition, that is the number of ratings available for each user in the training set (See Section 4.1.2), we measured the running time by altering one factor while fixing the other two parameters. For example, to evaluate the effect of the number of users on the running time of the algorithm, we fix the number of items and preferences in the training dataset, while varying the number of users by randomly selecting from 20% to 100% of all available users. Then, we compute the average running time for each recommendation. Similar steps have been followed for evaluating the effect of the number of pairwise preferences and the number of items on the running time. The results are presented in Fig.9.

As shown in Figure. 9, the computational complexity is almost constant while varying 𝑀 and 𝑆, but, there is a quadratic rise of the running time when the number of items increases.

Note that GRank’s computational complexity is quadratic to the number of items since it considers all the possible pairwise comparisons. However, the real number of pairwise comparisons is much less than 𝑁2 in many practical applications, because items usually form clusters and the comparisons often happen within those clusters. This means that in practice, we may be able to prune TPG by omitting those comparisons that have not been done by any user.



**Fig. 9. Scalability analysis of GRank in terms of the number of users, items, and pairwise preferences assigned by the users**

** Discussion**

GRank framework was introduced to resolve the sparsity problem of NCR techniques for similarity calculation. In the following, we briefly discuss how GRank is accomplished to resolve the issues.

* Using TPG, GRank implicitly aggregates different kinds of users’ similarities: One type of user similarities is calculated based on their common comparisons. This type of similarity is reflected by the paths following < 𝑈 − 𝑃 − 𝑈 > that connect two users through a pairwise comparison’s node. Additionally, two users are assumed to be similar if they both have preferred a particular item A in some comparisons, even if the items over which it has been preferred are different for those two users. The same holds for situations in which two users prefer different items over some particular item A. These types of similarity can be discovered through tracking the

paths in form of < 𝑈 − 𝑃 − 𝐼𝑑 − 𝑃 − 𝑈 > and < 𝑈 − 𝑃 − 𝐼𝑢 − 𝑃 − 𝑈 >,

respectively. Combination and replication of these meta-paths define many different relations among entities and GRank captures and aggregates them in its rank calculation process.

* Following the meta-paths < 𝑃 − 𝑈 − 𝑃 > in TPG, GRank implicitly finds correlated preferences that are the preferences of similar users. Correlate preference are highly connected through paths following < 𝑃 − 𝑈 − 𝑃 >. consequently, unknown preferences are iteratively estimated by propagating the known preferences of the target user to those unknown preferences that are correlated to them. This information can be used to calculate users’ similarity even in case of no common pairwise comparisons. We refer to an example to clarify the concept: As illustrated in Fig.3, 𝑢1, 𝑢2, and 𝑢5 , each one, has one link to the preference node *B<A, B<D*, and *C<D*. Therefore, the similarity between each pair of them will be zero according to the Kendall correlation. However, TPG reflects that {𝐴 > 𝐵} is highly correlated to the preference node{𝐶 < 𝐷} while it has no relations with B<D. Therefore, 𝑢1 is more similar to 𝑢5 as a consequence of following the paths passing from{𝐴 > 𝐵} and {𝐶 > 𝐷}
* Taking advantages of TPG and PageRank algorithm, GRank directly estimates the users’ ranking over unseen items. Personalized PageRank computation in TPG enables GRank to aggregate the ranking information obtained from different forms of meta-paths < 𝑢, 𝑣1, … , 𝑣𝑚, 𝑖𝑑 > and < 𝑢, 𝑣1, … , 𝑣𝑚, 𝑖𝑢 > for fast and accurate prediction of users’ ranking. Note that this approach differs from the typical 3-step framework (calculation of similarity, generation of preference matrix, and, inference of total ranking) used in all neighbor-based approaches.

** Conclusion**

In this paper, we studied how a graph-based framework can be designed and exploited to address the shortcomings of current neighbor-based collaborative ranking algorithms. For this purpose, we suggested that modeling the preference data as a new tri-partite graph structure and then exploring it can help us to capture the different kinds of relations existing in a ranking preference dataset (e.g. users’ similarities, items’ similarities, etc.). We also proposed a random-walk approach to make recommendation based on the proposed structure. Experimental results showed significant improvement of the suggested framework, GRank over other state-of-the-art graph-based and neighbor-based collaborative ranking methods. It seems that the graph based approach of GRank can be beneficial both in sparse and dense data sets. In dense data sets, it can form the neighborhoods more precisely, by exploring different paths that exist among entities. In sparse data sets, that users rarely have common pairwise comparisons and direct neighborhoods are usually very small, it can still traverse the edges to find farther neighbors and use their information as well for recommendation. The proposed graph structure has been mainly used here for finding closeness between users and items, but it can also be used for other purposes like finding clusters of similar users and similar items, and also discovering correlated preferences which are some of essential concepts in the field of recommendation systems.

**References**

Adomavicius, G., & Kwon, Y. (2012). Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. *IEEE Transactions On Knowledge And Data Engineering*, *24*(5), 896–911.

AvSegal, I., Katzir, Z., & Gal, K. (2014). EduRank : A Collaborative Filtering Approach to Personalization in E-learning. In *Educational Data Mining 2014* (pp. 68–75).

Balakrishnan, S., & Chopra, S. (2012). Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining. ACM* (pp. 143–152).

Bellogin, A., & Parapar, J. (2012). Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *Proceedings of the sixth ACM conference on Recommender systems* (pp. 213–216).

Cacheda, F., Carneiro, V., Fernández, D., & Formoso, V. (2011). Comparison of collaborative filtering algorithms. *ACM Transactions on the Web*, *5*(1), 1–33. http://doi.org/10.1145/1921591.1921593

Chen, B., Wang, J., Huang, Q., & Mei, T. (2012). Personalized video recommendation through tripartite graph propagation. *ACM Transactions on Multimedia Computing,*

*Communications and Applications*, *10*(4), 1133.

http://doi.org/10.1145/2393347.2396401

Chiluka, N., Andrade, N., & Pouwelse, J. (2011). A Link Prediction Approach to Recommendations in Large-Scale User-Generated Content Systems. In *Advances in information retrieval* (pp. 189–200).

Chowdhury, N., Cai, X., & Luo, C. (2015). BoostMF : Boosted Matrix Factorisation. In

*Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)* (pp. 3–

18). http://doi.org/10.1007/978-3-319-23525-7

Christakopoulou, K., & Banerjee, A. (2015). Collaborative Ranking with a Push at the Top. In *WWW 2015: Proceedings of the 24th International Conference on World Wide Web*

(pp. 205–215). http://doi.org/10.1145/2736277.2741678

Cong, J. (2009). RankClus : Integrating Clustering with Ranking for Heterogeneous Information Network Analysis. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM,* (pp. 565–576).

Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook* (pp. 107–144.).

Dhanjal, C., Clémençon, S., & Gaudel, R. (2015). Collaborative filtering with localised ranking. In *Proceedings of Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI’15)* (Vol. 4, pp. 2554–2560).

Fan, C., & Lin, Z. (2013). Collaborative Ranking with Ranking-Based Neighborhood. *Web Technologies and Applications*, 770–781.

Fouss, F., Pirotte, A., Renders, J. M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *19*(3), 355–369. http://doi.org/10.1109/TKDE.2007.46

Guo, G., Zhang, J., & Yorke-Smith, N. (2015). Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *Knowledge-Based Systems*, *74*, 14–27. http://doi.org/10.1016/j.knosys.2014.10.016

Huang, Z., Li, X., & Chen, H. (2005). Link Prediction Approach to Collaborative Filtering. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries* (pp. 141–142).

Jiang, J., Lu, J., Zhang, G., & Long, G. (2011). Scaling-Up Item-Based Collaborative Filtering Recommendation Algorithm Based on Hadoop. *2011 IEEE World Congress on Services*, 490–497. http://doi.org/10.1109/SERVICES.2011.66

Kim, K. jae, & Ahn, H. (2008). A recommender system using GA K-means clustering in an online shopping market. *Expert Systems with Applications*, *34*(2), 1200–1209. http://doi.org/10.1016/j.eswa.2006.12.025

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, *42*(8), 30–37.

Lee, S., Park, S., Kahng, M., & Lee, S. (2013). PathRank : Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems. *Expert Systems with Applications*, *40*, 684–697.

Lerche, L., & Jannach, D. (2014). Using Graded Implicit Feedback for Bayesian Personalized Ranking. In *Proceedings of the 8th ACM Conference on Recommender systems. ACM* (pp. 353–356).

Li, X., & Chen, H. (2013). Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, *54*(2), 880–890. http://doi.org/10.1016/j.dss.2012.09.019

Liu, J., Wu, C., Xiong, Y., & Liu, W. (2014). List-wise probabilistic matrix factorization for recommendation. *Information Sciences*, *278*, 434–447.

Liu, N., & Yang, Q. (2008). EigenRank : A Ranking-Oriented Approach to Collaborative Filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM* (pp. 83–90).

Lu, Q., Chen, T., Zhang, W., Yang, D., & Yu, Y. (2012). Serendipitous personalized ranking for top-N recommendation. *Proceedings - 2012 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2012*, 258–265. http://doi.org/10.1109/WI-IAT.2012.135

Meng, L., Li, J., & Sun, H. (2011). WSRank: A Collaborative Ranking Approach for Web Service Selection. In *2011 IEEE 11th International Conference on Computer and Information Technology* (pp. 103–108). Ieee. http://doi.org/10.1109/CIT.2011.30

Musa, A. B. (2012). Comparative study on classification performance between support vector machine and logistic regression. *International Journal of Machine Learning and Cybernetics*, *4*(1), 13–24. http://doi.org/10.1007/s13042-012-0068-x

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: bringing order to the web., 1–17. http://doi.org/10.1109/ICCKE.2013.6682858

Pan, W., Zhong, H., Xu, C., & Ming, Z. (2015). Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems*, *73*, 173–180.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-thieme, L. (2009). BPR : Bayesian

Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 452–461). http://doi.org/10.1145/1772690.1772773

Said, A., Kille, B., Jain, B., & Albayrak, S. (2012). Increasing Diversity Through Furthest Neighbor-Based Recommendation. *In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, 1–4.

Sawant, S. (2013). Collaborative Filtering using Weighted BiPartite Graph Projection - A Recommendation System for Yelp. *Proceedings of the CS224W: Social and Information Network Analysis Conference*.

Shang, M. S., Fu, Y., & Chen, D. Bin. (2008). Personal recommendation using weighted bipartite graph projection. In *2008 International Conference on Apperceiving Computing and Intelligence Analysis, ICACIA 2008* (pp. 198–202). http://doi.org/10.1109/ICACIA.2008.4770004

Shang, M.-S., Zhang, Z.-K., Zhou, T., & Zhang, Y.-C. (2010). Collaborative filtering with diffusion-based similarity on tripartite graphs. *Physica A: Statistical Mechanics and Its Applications*, *389*(6), 1259–1264. http://doi.org/10.1016/j.physa.2009.11.041

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. *Recommender Systems Handbook*, 257–298. http://doi.org/10.1007/978-0-38785820-3\_8

Shi, Y., Karatzoglou, A., & Baltrunas, L. (2012). Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems. ACM* (pp. 139–146).

Shi, Y., Karatzoglou, A., Baltrunas, L., & Larson, M. (2013). xCLiMF : Optimizing Expected Reciprocal Rank for Data with Multiple Levels of Relevance. In *Proceedings of the 7th ACM conference on Recommender systems. ACM* (pp. 431– 434).

Shi, Y., Larson, M., & Hanjalic, A. (2010). List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender system* (pp. 269–272).

Shi, Y., Larson, M., & Hanjalic, A. (2013). Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation. *Information Sciences*, *229*, 29– 39.

Silva, A., & Zaki, M. (2013). ProfileRank : Finding Relevant Content and Influential Users based on Information Diffusion. In *Proceedings of the 7th Workshop on Social*

*Network Mining and Analysis. ACM* (pp. 1–9).

http://doi.org/10.1145/2501025.2501033

Sun, Y., Han, J., Yan, X., & Yu, P. S. (2011). PathSim : Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. In *VLDB* (pp. 992–1003).

Ting, Y., Yan, C., & Xiang-wei, M. (2013). Personalized Recommendation System Based on Web Log Mining and Weighted Bipartite Graph. *2013 International Conference on Computational and Information Sciences*, 587–590.

http://doi.org/10.1109/ICCIS.2013.161

Ting-Peng, L., Hung-Jen, L. A. I., & Yi-Cheng, K. U. (2006). Personalized Content Recommendation and User Satisfaction: Theoretical Synthesis and Empirical Findings. *Journal of Management Information Systems*, *23*(3), 45–70. http://doi.org/10.2753/MIS0742-1222230303

Volkovs, M. N., & Zemel, R. S. (2012). Collaborative Ranking With 17 Parameters. In *Advances in Neural Information Processing Systems* (pp. 2294–2302).

Wang, S., Sun, J., & Gao, B. J. (2014). VSRank : A Novel Framework for Ranking-Based collaborative filtering. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *5*(3), 51.

Weimer, M., & Karatzoglou, A. (2007). Maximum Margin Matrix Factorization for Collaborative Ranking. *Advances in Neural Information Processing Systems*, 1–8.

Weimer, M., Karatzoglou, A., & Smola, A. (2008). Improving maximum margin matrix factorization. *Machine Learning*, *72*(3), 263–276. http://doi.org/10.1007/s10994-0085073-7

Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., & Sun, J. (2010). Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’10* (pp. 723–731). ACM Press.

http://doi.org/10.1145/1835804.1835896

Xiao, Z., Che, F., Miao, E., & Lu, M. (2014). Increasing serendipity of recommender system with ranking topic model. *Applied Mathematics and Information Sciences*, *8*(4), 2041–2053. http://doi.org/10.12785/amis/080463

Yang, C., Wei, B., Wu, J., Zhang, Y., & Zhang, L. (2009). CARES: a ranking-oriented CADAL recommender system. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries. ACM* (pp. 203–212).

Yao, W., He, J., Huang, G., Cao, J., & Zhang, Y. (2013). Personalized Recommendation on Multi-Layer context graph. In *Web Information Systems Engineering–WISE 2013* (pp. 135–148).

Yu, X., Ren, X., Sun, Y., & Gu, Q. (2014). Personalized Entity Recommendation : A Heterogeneous Information Network Approach. In *7th ACM international conference on Web search and data mining* (pp. 283–292).

Yu, X., Ren, X., Sun, Y., Sturt, B., Khandelwal, U., Gu, Q.,Han, J. (2013). Recommendation in Heterogeneous Information Networks with Implicit User Feedback. In *Proceedings of the 7th ACM conference on Recommender systems. ACM* (pp. 347–350). Zhang, M., Guo, X., & Chen, G. (2015). Prediction uncertainty in collaborative filtering :

Enhancing personalized online product ranking. *Decision Support Systems*, *83*, 10–

21. http://doi.org/10.1016/j.dss.2015.12.004

Zhang, Z., & Kulkarni, S. R. (2014). Detection of Shilling Attacks in Recommender Systems via Spectral Clustering. In *Information Fusion (FUSION), 2014 17th International Conference on. IEEE.* (pp. 1–8).

Zhang, Z.-K., Zhou, T., & Zhang, Y.-C. (2010). Personalized recommendation via integrated diffusion on user–item–tag tripartite graphs. *Physica A: Statistical*

*Mechanics and Its Applications*, *389*(1), 179–186. http://doi.org/10.1016/j.physa.2009.08.036

Zhen, Y., Li, W. J., & Yeung, D. Y. (2009). TagiCoFi: Tag Informed Collaborative Filtering. *Proceedings of the Conference on Recommender Systems (RecSys)*, 69–76.

http://doi.org/10.1145/1639714.1639727

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., Zhang, Y.-C., & Settings, M. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences of the United States of America*, *107*(10), 4511–5. http://doi.org/10.1073/pnas.1000488107

Zhou, T., Ren, J., Medo, M., & Zhang, Y. C. (2007). Bipartite network projection and personal recommendation. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *76*(4), 1–7. http://doi.org/10.1103/PhysRevE.76.046115