豆瓣电影推荐

姓名：何占魁、李冬皓、冉诗菡
学号：15307130175、15307100013、15307130424

1.任务和算法简介

1.1 社交网络挖掘简介

我们生活在大数据时代，随着社交媒体的蓬勃发展，世界各地数以亿计的网民花费大量的时间在社交媒体上以史无前例的速度分享、交流、联系、互动，这在使得互联网成为一个充满活力与朝气的领域同时也产生了海量的用户生成数据。这些海量信息由公民记者收集、组织和发布，同时又被无数的用户阅读和传播并给予即时的反馈。社交媒体使得我们可以随时随地与他人沟通和交流，并且可以站在一个全新的角度在空前的规模下观察人类的行为。社交媒体平台为我们提供了一个绝好的机会，使得我们可以从海量数据中挖掘人类行为模式，进而对人类个体进行全面的剖析和理解，这在社交媒体出现前是根本无法完成的任务。通过更好地理解个体，我们可以设计出能够更好地适应个体需求的计算系统，进而更好地为个人乃至整个社会服务。

社交媒体挖掘是从社交媒体数据中表示、分析和抽取可操作模式的过程，它是一个新兴的研究领域，其中有很多亟待解决的难题，例如大数据悖论、获取足够的样本、噪声消除谬误和评价困境等。社交媒体的存在打破了现实世界与虚拟世界之间的界限，然而遗憾的是社交媒体数据与数据挖掘中我们所熟悉的传统数据有着显著的差别。除去极大的数据规模，这些主要由社交媒体用户创建的数据噪声很大且完全无结构化，还蕴含着丰富的如好友关系、关注与被关注等社交关系。社交媒体数据的独特性需要我们开发出全新的数据挖掘技术来处理带有丰富社会关系的用户创建内容。

社交媒体挖掘以一种可计算的方式表示和度量社会媒体的虚拟世界，并且建立模型以帮助我们理解其中的交互。此外，它还提供了工具帮助我们从社交媒体平台这一虚拟世界中挖掘出有趣的模式，分析信息传播，研究影响力和同质力，提供有效的推荐以及分析社交媒体中全新的社会行为。

1.2 推荐系统简介

社交媒体中的个体每天都需要做出各种各样的决定，这些决定有可能是关于购买一款产品、购买一项服务、添加一个朋友、租一部电影等。个体常常需要面对很多选择，这些不同的选择、追求最优性以及个体有限的知识使得每个个体亟需外界的帮助。有时候人们会求助于搜索引擎来寻求推荐，然而搜索引擎反馈的结果很难符合用户的特定需求，并且这些结果是依赖查询的，只要搜索问题相同，通常检索结果也相同，而非为每一个不同的搜索用户量身定制的。

目前有很多的应用和算法来帮助个体进行简单、快速且更精准的决策，这些算法能够满足每个用户的独特需求，即具备了用户自定义推荐功能。这些算法被称为推荐算法或者推荐系统。与搜索引擎不同，推荐系统以每个用户的兴趣为基础，为用户推荐个性化选择。推荐系统通常用于产品推荐，目的是推荐那些用户感兴趣的产品。假设用户集合为U，产品集合为I，那么一个推荐算法就是学习一个函数f：

$$f : U \times I \rightarrow \mathbb{R}$$

换句话说，该算法通过学习一个函数可以为每一个用户-产品对(u, i)分配一个真实的值，用于表明用户u对于产品i的感兴趣程度，这个值也代表了用户u对于产品i的评价等级。推荐系统不仅限于产品推荐，还可以推广为推荐人力和物力，例如广告或其他内容。推荐系统也面临很多挑战性问题，如冷启动问题、数据稀疏问题、系统攻击问题、个人隐私问题，以及缺乏推荐原因的解释。

在社交网络中，网站经常使用一些经典的推荐算法来推荐项目或产品，这些技术可以分为基于内容的方法和协同过滤技术。

- 在基于内容的方法中，主要是根据内容（如项目描述）和用户资料之间的相似度进行推荐。
- 在协同过滤算法中，根据用户的用户-项目评分记录进行推荐。
    1. 协同过滤算法可以分为基于记忆和基于模型两种技术。
    2. 在基于记忆的技术中，使用用户（基于用户）或项目（基于项目）之间的相似度预测未知评分。
    3. 在基于用户的技术中，假设存在一个基础模型描述了用户如何对项目进行评分。使用矩阵分解技术可以近似地预测未知评分。

在社交媒体中，也可以根据朋友信息来做出推荐。可以仅仅使用这些朋友信息进行推荐（例如朋友推荐），也可以将其加入到经典算法中，或者作为限制条件对经典推荐技术的推荐结果进行限制。

关于推荐系统的评估，评估手段有准确率、相关性和推荐项目的排序。利用NDCG、NMAE和RMSE评估准确率、精确率和召回率；利用F度量值评估推荐的相关性；利用相关系数值评估推荐的排序。

1.3 豆瓣电影网站简介

豆瓣(douban)是一个创立于2005年的社区网站，它以书影音起家，提供关于书籍、电影、音乐等作品的信息，无论描述还是评论都由用户提供（User-generated content）。网站还提供书影音推荐、线下同城活动、小组话题交流等多种服务功能，它更像一个集品味系统（读书、电影、音乐）、表达系统（我读、我看、我听）和交流系统（同城、小组、友邻）于一体的创新网络服务，一直致力于帮助都市人群发现生活中有用的事物。

豆瓣没有编辑写手，没有特约文章，也没有六百行的首页和跳动的最新专题。豆瓣的藏书甚至没有强加给用户的“标准分类”。这里所有的内容，分类，筛选，排序都由用户产生和决定。给评论一个“有用”，它的排位会自动上升；贴“我女儿的最爱”给一本书，它会在整个网站的标签分类中出现。豆瓣相信大众的力量，多数人的判断，和数字的智慧。通过网站幕后不断完善之中的算法，有序和有益的结构会从无数特异而可爱的个性中产生。**我们希望用社交网络挖掘的方法，探究豆瓣用户的关系、结构和内在特征。**

豆瓣的发起者发现，对多数用户做选择最有效的帮助其实来自其亲友和同事。随意的一两句推荐，不但传递了他们自己真实的感受，也包含了对该用户口味的判断和随之而行的筛选。他们不会向单身汉推荐育儿大全，也不会给老妈带回赤裸特工。遗憾的是，即使把一个用户所有的亲友加起来，听过看过的仍然有限。而且，口味最类似的人却往往是陌路。**我们在下文中会探索用协同过滤和基于内容的方式，构建豆瓣电影的推荐系统。**

2.社交网络可视化

2.1 数据预处理

在分析社交网络的数据时，我们只需要选取user表格中的信息。“following_id”这一列说明了这个用户的关注情况，我们通过这一列将两个用户连接起来。

在处理数据时，我们发现，“following_id”这列使用的是用户的名字，其中包括了许多不在我们数据集中的用户。对于这些用户，我们在不同的任务中进行了不同的处理，接下来会详细说明。

2.2 数据统计信息

2.2.1 数据规模统计信息

经过数据处理，我们统计出这个数据集的一些基本信息：

其中，我们定义核心用户为拥有基本信息的用户，且其拥有至少一个与有基本信息用户好友关系的用户。

- 涉及到的总用户数：88039
- 总的好友关系数：187265
- 拥有基本信息的用户数：5000
- 核心用户数：768
- 核心用户好友关系数：951

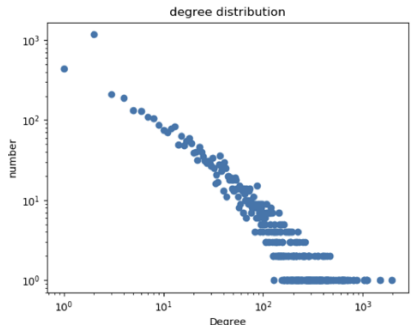
我们可以看到，我们的核心用户很少，即给定数据中，很少有用户有交互。这对于我们接下来基于社交网络的推荐系统的构建产生了很大的困难。即如果我们希望利用好友的信息，只有很少数的信息可以利用，剩余的用户会面临冷启动的问题，即没有对应的好友信息，如果我们完全基于社交网络进行推荐，这将使得我们的推荐系统可用性变的很差。

2.2.2 度分布图

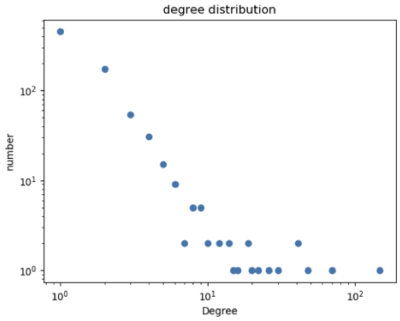
这里我们展示了这个社交网络中的度分布图，从度分布图中可以直观的看出这个社交网络的基本信息。其中X，Y坐标轴是对数尺度，这样可以看出是否符合幂律分布。

这张度分布图中，我们展示了所有具有基本信息的用户的度分布，我们可以很明显的看出，曲线近似一条向下的直线，这个分布符合幂律分布，是很典型的社交网络。

所有用户度分布

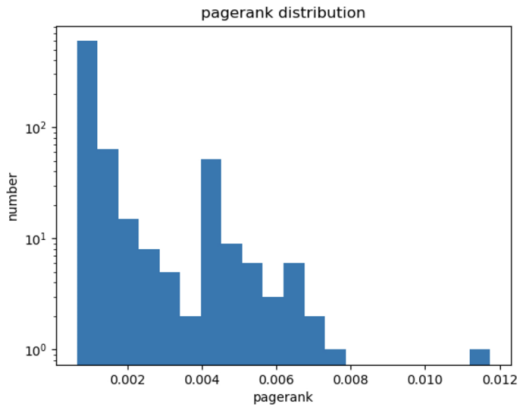


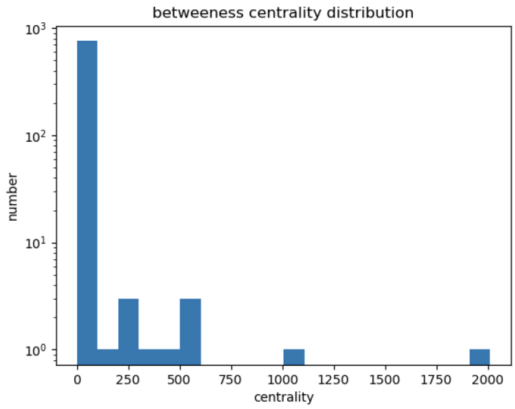
核心用户度分布

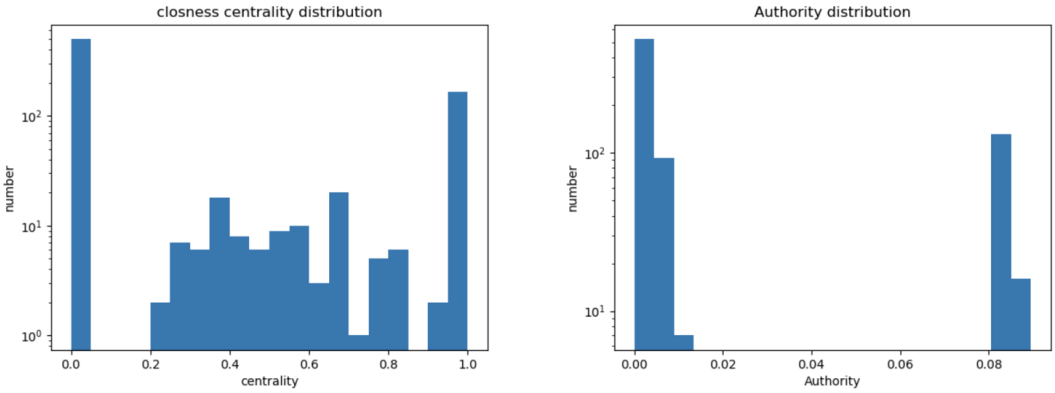


第二张图标展示了核心用户的度分布图，可以看出仍然符合幂律分布，只是因为数据量较小，得到的分布图不是很平滑。

2.2.3 其他指标统计图







我们可以从图中看出各个统计指标的分布情况。

- 左上反映了PageRank 值的分布，可以看出有两个峰比较明显，另外只有一个值的PageRank值很高，远远高出其他值。
- 第二张图反映了中间中心性，可以看出这个值的分布也是很多节点的值较小，只有一个值的中间性较大。
- 第三张图反映了接近中心性的分布，这个分布于其他的分布不同，中间值比较多。
- 第四张图反映了权威性，可以发现，实际上很多节点并没有入边或者很少。而中间不存在一些值。我们经过分析，认为产生这种现象的原因是在采集数据的时候，我们总是会从种子用户开始，进行宽度优先的爬取。因此可以认为权威值较高的点是种子节点，因为他们拥有更多的核心用户作为朋友。而其他节点因为宽度优先的爬去终止，因此他们的好友不在核心用户中。

2.3 社交网络挖掘算法

2.3.1 社群发现

为了清楚的展示豆瓣社交网络的结构与关系，我们使用了社群发现的算法，将豆瓣中的用户根据关系划分为不同的社群，从而为可视化的结果产生不同的颜色划分，能够更加直观的看出各个用户之间的关系。在实践中，我们采用了Fast Unfolding算法。

首先介绍社群发现中的基本概念：模块度公式

$$Q = \frac{1}{2m}\sum_{i,j}[A_{i,j} - \frac{k_i k_j}{2m}]\zeta(c_i, c_j)$$

其中，当且仅当$c_i = c_j$时，$\zeta(c_i, c_j) = 1$，否则为0。

社区划分的目标是使得划分后的社区内部的连接较为紧密，而在社区之间的连接较为稀疏，通过模块度的可以刻画这样的划分的优劣，模块度越大，则社区划分的效果越好 。

Fast Unfolding算法是一个迭代算法，主要包括两个阶段：

- 第一阶段称为Modularity Optimization，主要是将每个节点划分到与其邻接的节点所在的社区中，以使得模块度的值不断变大。
- 第二阶段称为Community Aggregation，主要是将第一步划分出来的社区聚合成为一个点，即根据上一步生成的社区结构重新构造网络。重复以上的过程，直到网络中的结构不再改变为止。

具体的算法过程如下所示：

1. 初始化，将每个点划分在不同的社区中。
2. 对每个节点，将每个点尝试划分到与其邻接的点所在的社区中，计算此时的模块度，判断划分前后的模块度的差值ΔQ是否为正数，若为正数，则接受本次的划分，若不为正数，则放弃本次的划分。
3. 重复以上的过程，直到不能再增大模块度为止。
4. 构造新图，新图中的每个点代表的是步骤3中划出来的每个社区，继续执行步骤2和步骤3，直到社区的结构不再改变为止。

2.3.2 力导向算法

在可视化结果中，为了将点的位置确定，如果我们仅仅是随机分配各个节点的位置，整个图就会显得很杂乱，会有很多过长的边，并且关系会变得错综复杂。因此我们需要运行力导向算法来使得每个点的距离尽可能远，并且更加重要的点在中心。

力导向算法的核心思想很简单，我们把社交网络中的每个节点认为是一个物理系统，每个节点被认为是一个带有正电荷的粒子，因此存在电磁力使得它们产生有相互远离的力，其斥力的大小随着距离的增加而减少。而每一条边被认为是一个弹簧，遵守与类似胡克定律的拉力，其拉力随着边的长度增加。开始的时候可以随机初始化节点的位置，之后因为力不平衡，我们需要对点的位置进行调节。因此，系统最终会收敛在引力和斥力相平衡的状态，这时各个节点之间的距离控制在合适的范围内，并且具有很多相互联系的边的点更加倾向于在一起，方便我们观察。

2.3.3 度量重要性

在可视化过程中，我们希望节点的大小可以反应节点的重要性，从而对更重要的节点给予更多关注。我们考虑了各种度量重要性的指标，比如最简单的利用度（包括入读和出度）来表示节点的重要性，之后还有中间中心性（betweenness centrality），接近中心性（closeness centrality）以及PageRank值。在可视化中，因为这些值的取值范围和方差不同，我进行了归一化处理并且根据显示效果，对其进行单调的非线性变化，如指数变化，对数变化。

最终，我们将各种度量中心度的算法都进行了展示，可以从不同角度分析社交网络。

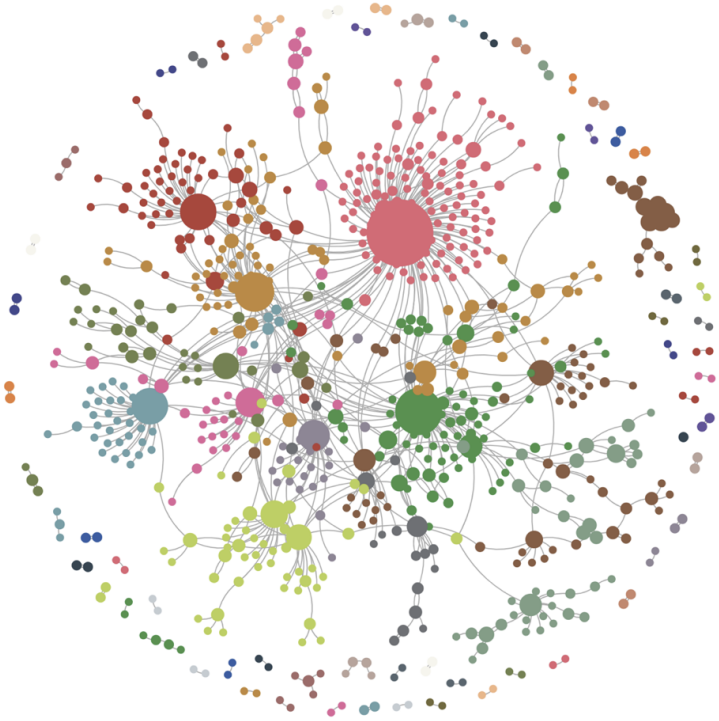
2.4 社交网络可视化工具

我们使用PyEcharts作为数据可视化工具，PyEcharts提供Echarts的非官方python接口，方便用户使用。Echarts是百度提供的免费数据可视化工具，拥有极佳的视觉效果以及交互体验。并且生成的图表为html格式，可以跨平台浏览，方便快捷。

在我们可视化的结果中，我们画出了所有的核心用户以及他们之间的关注关系，并且将鼠标移到改节点上，可以高亮显示出该节点及其邻居节点；将鼠标放在边上，可以找出连接的两个节点，并且可以显示出边的方向。再加上我们可以指定不同的颜色分类以及不同的大小，我们可以很直观的对这个网络有一个直观的认知，比如我们可以直观地找出最有影响力的用户。

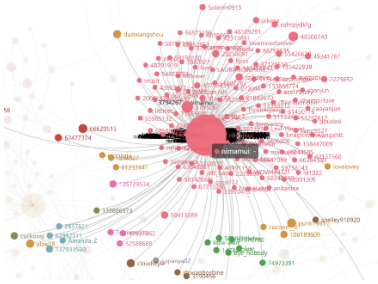
2.5 网络可视化展示说明

- 首先，我们可以看到这样的可视化效果。看到这个可视化最直观的印象为中上部分巨大的粉色节点以及其子节点。当我们将鼠标移上去时，即可显示这个节点的名字。（下图一）
- 其次，我们可以显著的观察到不同颜色的划分，这个是社群发现算法的作用，将不同的社群给与不同的颜色。



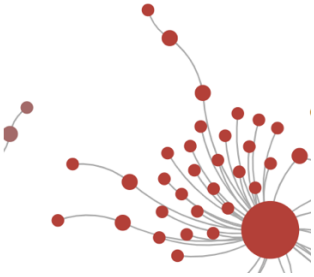
- 之后，我们可以观测到许多散落在周围的，于中间不联通的节点，他们是一些种子节点，但是由于其子代没有与大图联通，因此孤立，仅仅与少数节点相连。他们在图的周围，因为引导力算法倾向于排斥他们，而他么没有向心的力牵引。（下图二）
- 另外，我们观察到了相互关注的节点占比不是很多，很多节点依靠单边相邻，因此我们认为豆瓣中熟人不多，因为熟人更倾向于相互关注，而陌生人之间往往会产生不平等的关注关系，比如新手关注大V。（下图三）
- 我们希望找到一些三元闭包，但是图中没有很多，我们总结有两个原因，第一个可能是图的大小有限，第二个是网友之间社交压力比较小，关注关系不会传递很快。（下图四）

图一



图二

图三



图四

---

3. 推荐系统模型

3.1 数据预处理

3.1.1 相关信息选取

给定user和moive两张表之后，我们首先筛选出我们认为对于推荐系统有潜在帮助的相关信息：

- user表信息选取：**用户编号u，**以及其对**电影i的打分y_{ui}**。
- movie表信息选取：**电影编号i**，**电影简介文本$Summary_i$**，**电影平均得分$\overline{y}_i$**，**导演编号d**和演员编号a**。

值得说明的是：在数据集douban_1k中我们另外选取了movie表格中的类型信息 $Type_i$，但在新的数据集douban_5k中并没有包含这个信息，因此我们使用简介文本、导演和演员等信息探索电影类别的其他表示方式。选取这些信息的原因会在3.4章节中详述。

3.1.2 数据规范化

我们对数据进行了以下的整理，为模型的训练提供基础：

- **线性变换，使电影平均分$\overline{y}_i \in [1,5]$**： 我们知道豆瓣电影中每个用户的评分y_{ui}是五星制（如“《纯洁心灵·逐梦演艺圈》一星评分”事件），而豆瓣呈现的电影平均分$\overline{y}_i$是十分制（如《霸王别姬》评分高达9.5分）。为能使y_i和$\overline{y}_i$之间具有可比性，我们将$\overline{y}_i$的分值线性变换到$[1,5]$的区间内。
- **索引化，为特征提取做准备**：
  将所有的电影、用户、导演、演员赋予一一对应的索引号，为之后的One-Hot向量、以及深度学习的Embedding矩阵的Lookup操作做准备。

3.1.3 数据集划分

- 我们把用户没有评分过的电影当做负样本，分值为0，其他分值和用户评分一致。
- 对每个用户我们随机抽取4个正样本(如果该用户评分数大于4个)放入测试集中，在测试集中再针对每个用户选取46个负样本。
- 我们对于所剩样本，进行随机负采样，每个用户选取2个负样本，每个Epoch之后选取新的负样本，以达到让模型“看”到更多负样本实例。

# Train Set Size	# Test Set Size	Negative Sampling Ratio
915720	24000	1 User : 2 Negative Sample

3.2 统计信息

3.2.1 基本统计信息

#USER	#MOVIE	#DIRECTOR	#ACTOR
5000	41785	22824	89619

3.2.2 交互统计信息

- 用户评分的电影平均数量为**184.29**部，用户最多有**500**次评分行为，最少评分次数为**15**次。
- 每部电影平均收到评分**2257.41**次，其中评分最多的电影有**193101**次评分，最少有**0**次评分。

3.3 推荐系统算法

针对豆瓣电影推荐任务，我们通过上述分析，总结出该任务的三个特点：

- **数据稀疏性：** user和movie 的交互数据只有 921540条，仅仅在user × movie的大小为928925000可能交互空间内占据0.44%，占比极小，数据极具稀疏性，这要求我们要考虑能有效处理稀疏数据的推荐系统模型，这为我们的模型设计起到了指导性作用。

- **特征多样性**：我们除了拥有user和movie之间的**交互历史**，也拥有大量的其他的**结构化信息**，如**电影评分、用户打分历史**等。除此之外，我们还掌握了大量的**非结构化信息**，如**评论文本、电影简介、用户之间的社交关系**。上述的结构化和非结构化的数据信息，除了激发了我们去探索传统特征和新兴深度特征的勇气，也提醒我们：为了能够兼容各种形式的特征，我们需要一个极具**可拓展性**的模型。
- **预测相对性**：通常对于机器学习任务，我们可以将其定义为**回归**问题或者**分类**问题。不过具体到**推荐系统**任务，我们可以发现它有这样的两个特点：
    1. 预测的具体分数不重要，重要的是：预测结果的**相对顺序**应该尽可能与真实情况相似。这里举出一个例子：

        > *任务描述：Caroline心目中电影A打分为5分，电影B打分为4分。模型在不知用户真实打分的前提下，为Caroline推荐A或B中的一部电影。*
        > *模型f_1结果：$f_1(U_{Caroline}, I_A) = 4.5$, 且 $f_1(U_{Caroline}, I_B) = 4.7$。 模型$f_2$结果：$f_2(U_{Caroline}, I_A) = 3.5$, 且 $f_2(U_{Caroline}, I_B) = 3.0$。*
        > *模型分析：从回归和分类任务的角度分析，模型f_1的性能远远好于f_2。但实际上模型f_2推荐的那一部电影，才是Caroline最想看的电影。*

        因此，这告诉我们用回归和分类的视角定义任务，可能无法真实地训练和评估有效的模型，与具体分值相比，预测结果的排序更重要。
    2. 召回率并不重要，重要的是准确率。因为对于每个用户来说，他们可能感兴趣的电影浩如烟海、目不暇接，因此高召回率对于用户来说没有什么意义，用户无法一次性接受那么多信息。所以，我们应当对排序的预测结果做一个最大数K的截断。

综上，考虑到**数据稀疏性、特征多样性**和**预测相对性**，我们在广泛阅读论文与代码，进行模型理论分析和比较之后，选取了**自动分解机算法(Factorization Machines)**，为进行进一步探索，我们选取了采用深度学习算法的**神经自动分解机算法(Neural Factorization Machines)**。

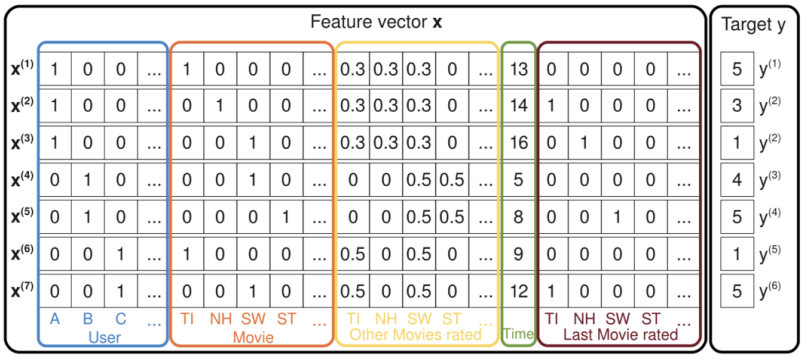
3.3.1 自动分解机算法 [Rendle, 2014]

自动分解机算法(FM)是2010年由Rendle提出的一种通用的方法，用特征工程模仿大多数的矩阵分解。它的核心公式如下：

$$\hat{y}(X) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i+1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

这里X是输入的特征向量，x_i是特征向量X的第i个元素取值，特征长度为n。$\hat{y}(X)$是给定特征X时的预测值，其余w_i和v_i均为通过优化算法更新的模型参数。自动分解机算法优势与我们的需求非常吻合：

- **处理稀疏数据**：模型分解机考虑到特征之间的交互项x_i和x_j，并通过矩阵分解进行建模，将稀疏数据稠密化为维度极低的向量v_i。既能有效地提取系数信息，又只需要线性规模的参数。
- **特征多样性**：作为通用算法，自动分解机对于输入的向量X并没有具体的内容要求，我们可以通过任意个数值型或者One-Hot的向量拼接，方便有效地拓展特征，论文中的这张图能够充分表明自动分解机算法的特征拓展特点，可以看出特征横向拼接的便利性：



- 运行高效性：作者通过矩阵展开和同项合并，将原公式的二次项进行了如下变换：

$$\sum_{i+1}^{n}\sum_{j=i+1}^{n}\langle v_i, v_j\rangle x_i x_j = \frac{1}{2}\sum_{f=1}^{k}((\sum_{i=1}^{n} v_{i,f}x_i)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2)$$

算法的时间复杂度从$O(n^2)$变为$O(n)$，算法的运行效率和其他同规模的算法相比非常高效。

考虑到以上特点与我们的任务需求的吻合性，自动分解机成为了我们的主要模型，我们将通过实验证明它的性能，以及通过学习神经自动分解机算法[]，探究深度学习自动分解机对我们的任务的帮助。

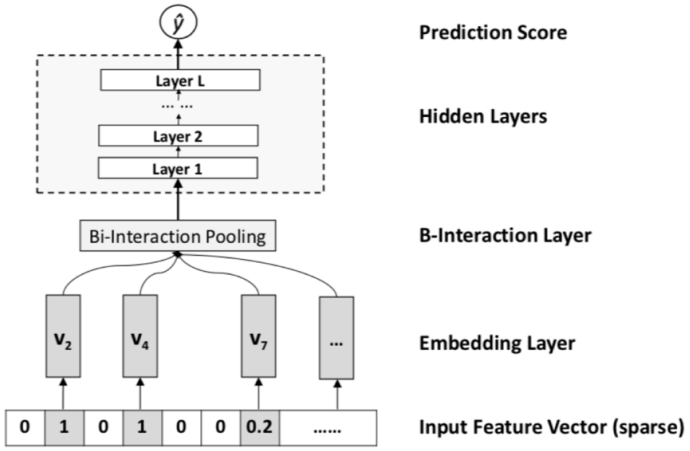
3.3.2 神经自动分解机算法 [Xiangnan He, 2017]

自动分解机算法(FM)是2017年由Xiangnan He团队提出的一种深度学习方法，是对于自动分解机工作的拓展。它的核心思想是用神经网络对交互项x_i, x_j进行非线性建模，在不过多增加参数规模的情况下提升模型性能，其核心公式为：

$$f_{BI}(V_X) = \frac{1}{2}[(\sum_{i-1}^{n} x_i v_i)^2 - \sum_{i=1}^{n}(x_i v_i)^2]$$

$$\hat{y}(X) = w_o + \sum_{i=1}^{n} w_i x_i + h^T \sigma_L(W_L(\dots\sigma_1(W_1 f_{BI}(V_X) + b_1)\dots) + b_L)$$

其模型结构的示意图为：



其优点是对于交互项的非线性提取性能更好、参数规模与自动分解机类似，但也因为拟合能力太强导致过拟合的现象出现，可能需要更多的防止过拟合的技巧。

3.3.3 其他算法：

- **SVM:** 我们受到自动分解机论文启发，选取SVM回归算法，作为我们的算法基准之一，用来体现自动分解机算法在处理稀疏数据上的有效性。
- **Item Popularity:** 我们通过基于流行度(这里定义社交网络为item的平均得分)实现**Item**

Popularity算法，这是推荐系统的常用基准算法，具体内容不在此赘述，用这个模型作为我们排序性能的参照。

3.4 推荐系统特征选取

3.4.1 传统特征

我们首先考虑结构化信息，进行如下处理变成特征：

- **结构化特征一**：user的One-hot信息，Moive的One-Hot信息和user的评分历史信息。
- **结构化特征二**：user 的One-Hot信息和 moive的One-Hot信息。
- **结构化特征三**：结构化特征二的基础上，加上该moive的平均得分信息。
- **结构化特征四**：结构化特征三的基础上，加上user的关注id信息。
- **结构化特征五**：结构化特征四的基础上，加上user的关注id的评分信息。

3.4.2 Doc2Vec [Le, 2014]信息

对于电影的简介，我们采用了Doc2vec模型进行处理，将文本信息变为向量，从而加入到我们的模型当中，作为对电影内容的概括描述。

实践中，我们采取了网络上面训练好的模型，其使用中文wiki上的网页作为语料库，经过较长时间的训练，已经有了很好的效果，下面利用不同的文本内容，计算对应的向量，利用**余弦相似度**进行相似程度的度量。期望相似的文本之间的相似度更高，从而体现模型的有效性。

我们使用了《爱乐之城》，《电锯惊魂》，《小丑回魂》这三部电影的简介信息来转换向量。其中，《爱乐之城》是一部爱情片，其余两部为恐怖片。通过比较他们之间的相似度来证明模型的有效性。

结果如下：

- 《电锯惊魂》《小丑回魂》：0.13657017591961818
- 《爱乐之城》《电锯惊魂》：-0.024096036870962415

我们可以看出，对于不同类型的电影，相似程度不同，证明我们的Doc2vec模型较为有效。

3.4.3 DeepWalk [Perozzi, 2014] 信息

对于电影、导演、演员的关系网络，我们采用了 DeepWalk算法进行表征学习，DeepWalk 算法由两个主要部分组成：**随机游走生成器**和**更新过程**，用SkipGram式的更新方式生成结点的表征向量。

在DeepWalk中，我们给定一个电影-导演-演员的异构网络图$G = (V, E)G = (V, E)$，通过训练，学习出$X_E \in \mathbb{R}^{|V| \times d}$ 的表示，其中d是一个极小的隐空间维度，在我们的模型中选取为128。

在我们的异构网络中只选取电影结点向量结果，再作为Ebedding初始化我们模型中的电影部分。

3.5 推荐系统结果

3.5.1 主要工具:

- **Tensorflow:** 我们使用Tensorflow框架在神经自动分解机作者Xiangnan He代码的基础上，将隐式反馈改成显式反馈。并添加我们的评价指标，由此实现**自动分解机**和**神经自动分解机模型**。
- **LibSVM:** 通过调用LibSVM模型库，得出我们的数据在**SVM**回归算法上的结果。
- **Numpy&Sklearn:** 我们通过Sklearn和基本的Numpy操作实现**Item Popularity**算法。

3.5.2 主要指标:

- RSME：均方根误差指标，在此处用来评价模型的拟合能力，其评价公式为：

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}}$$

- NDCG：归一化折损累积效益指标，此处用来评价模型的排序性能，是我们任务的重点，其公式为：

$$NDCG_k = \frac{DCG_k}{\max\{DCG_k\}}$$
$$DCG_k = \sum_{i=1}^{p} \frac{2^y - 1}{log_2(i+1)}$$

- 注：由于我们是采用的5分值的预测，许多用于二分类的有力指标如HR和MAP在此处不予考虑，NDCG指标可以看做MAP的改进版本。

3.5.3 模型探究结果：

- 结构化特征实验结果：

特征(上文已详述)	NDCG@FM	RMSE@FM
结构化特征一	31.98%	1.86
结构化特征二	**70.05%**	**1.03**
结构化特征三	17.2%	1.94
电影简介文本特征	67.83%	1.22
电影-导演-演员关系网络特征	67.91%	1.36

  - 结构化特征的详细说明在特征选取章节中说明，我们通过实验发现，结构化特征最有效的是`user`和`moive`的`One-Hot`信息，其他的特征信息我们实验结果均非常不理想，因篇幅原因不详列。
  - 我们使用Doc2Vec、DeepWalk特征向量初始化的模型，获得的比较优秀的效果(67.83%, 67.91%)。而结构化特征二的效果相对最好，我们分析认为：结构化特征二的信息仅为`user`和该`movie`，考虑到用户评分一般很少给出一星和两星，所有是否交互的信息影响非常大，在这个角度下来说，其他信息的增加可能会带来更多的干扰。
  - 我们接下来提到的实验特征均为结构化特征二。

- 模型对稀疏数据的处理能力与**SVM**对比：

Metric	**FM**	**NeuFM**	**SVM**
RSME (train)	0.45	0.56	2.62

  - 在这里我们重点关注FM、NeuFM、SVM模型对于稀疏数据的拟合性能，因为我们的测试集数据为排序性能设计，进行了大量的负采样。而对于SVM模型，我们研究发现其对于稀疏数据的拟合能力非常差，所有预测分值都在1左右徘徊，用训练集的均方根误差(RSME)已经能够明显地显示出它的拟合性能之差。
  - 因此，我们的实验数据可以表明，针对于稀疏的豆瓣电影数据t，SVM的拟合性能显著弱于自动分解机(FM)和神经分解机(NeuFM)。

- 模型的排序性能与**Item Popularity**对比：

Metric	**FM**	**NeuFM**	**ItemPop**
NDCG@10	**70.05%**	41.61%	18.25%
RSME (train)	**0.45**	0.56	0.84

  - 在这里我们将Item Popularity作为基准，通过RSME指标可以看出，该模型对于数据的拟

合能力已经比较不错，因此如果只考虑拟合指标，我们会认为Item Popularity算法的性能已经足够优秀。

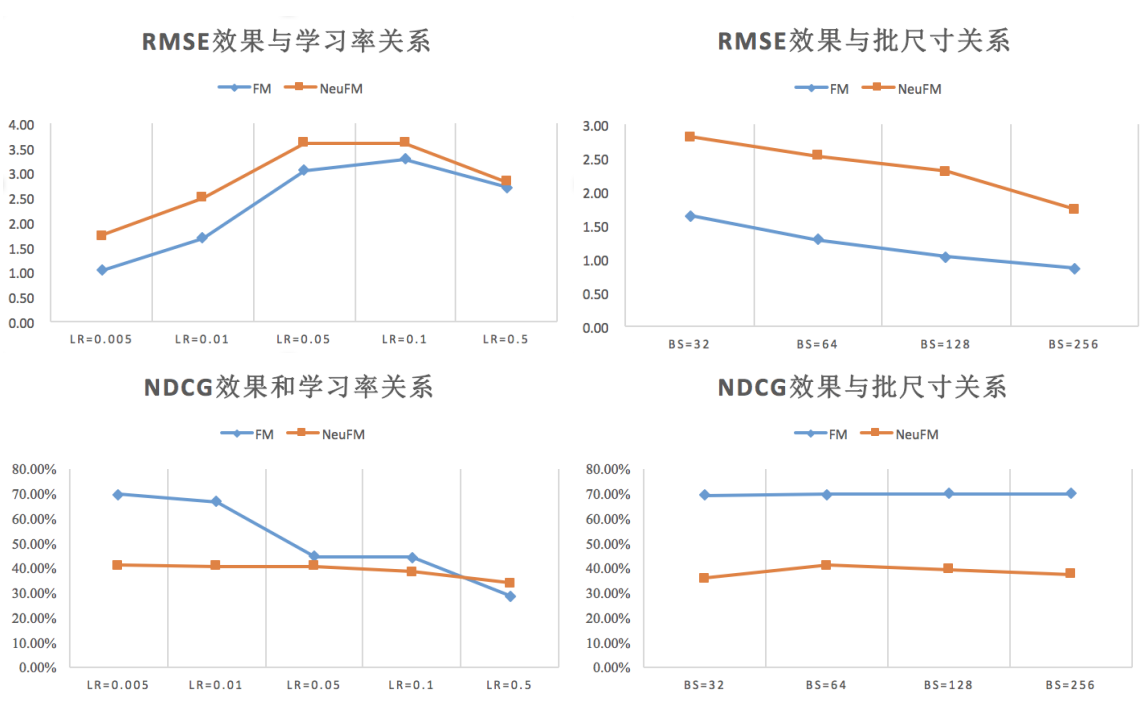
- 但通过NDCG的结果，我们可以看出Item Popularity在排序的性能上远远落后于FM和NeuFM。我们可以通过生活经验猜测，因为大众流行度在一定情况下能够代表用户个体的喜好，因此Item Popularity有一定的准确率，但建模是简单粗暴的。而FM和NeuFM模型通过调整user和item等特征的Embedding，能够让相似的用户有更近的关系和更好的预测结果，可以看做对ItemPop的“大众流行度”的细化，从而能够更加有效地预测用户喜好。

- 自动分解机和神经自动分解机对比：

模型	NDCG@10	NDCG@10 （train）
FM	**70.05%**	84.71%
NeuFM	41.61%	**96.95%**

  - 我们的多次实验结果显示FM的NDCG性能在70%左右波动，而NeuFM模型的NDCG性能在41%左右徘徊，对于豆瓣电影推荐任务而言，我们训练的FM效果显著优于NeuFM。
  - 我们对于NeuFM的实验结果显示，NeuFM在训练集上能够很快得到极高的NDCG值，但其对于测试集的性能显著差于FM，我们分析这是由于NeuFM强大的拟合能力导致过拟合。我们调整过隐含层尺寸、批归一化，也尝试过随机失活等抑制过拟合的方法，最后仍然没有将NeuFM的NDCG性能显著提升，非常遗憾。

- 模型的超参数分析：
  - 我们对模型的学习率、批尺寸等参数进行了Grid Search，其结果显示批尺寸对于模型性能影响不是很大，而NeuFM的学习率适应范围广于FM，FM在学习率0.01~0.08之间时，性能较佳。
  - 因此我们最终选取的模型参数为Batch Size=64, Learning Rate=0.05。



4. 未来工作展望

4.1 豆瓣社交网络挖掘

- 希望使用前端工具写出对于社交网络的交互式界面，方便检索社区的直观信息。
- 希望能够使用爬虫获取更多用户数据，挖掘更大范围的用户信息。

4.2 豆瓣推荐系统

- 我们的损失函数是简单的Square Loss，希望有条件能够尝试更多的针对排序算法的损失函数。
- 继续挖掘文本内容的信息，获取更多的有效的特征。

5.参考文献

1. **Rendle, Steffen. "Factorization machines." Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 2010.**
2. **He, Xiangnan, and Tat-Seng Chua. "Neural factorization machines for sparse predictive analytics." Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2017**
3. **Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." International Conference on Machine Learning. 2014.**
4. **Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.**