南京林業大學

本科毕业设计(论文)

题　　目: 群智化云服务 API 推荐方法

学　　院: 经济管理学院

专　　业: 电子商务

班　　级: 1505071

学　　号: 150507107

学生姓名: 丁俊尧

指导教师: 王磊　**职称:** 副教授

二〇一九年五月二十六日

群智化云服务 API 推荐方法

摘　要

随着互联网技术和互联网行业的发展，越来越多的云服务 API 被提供出来。尤其是最近几年，各大云计算提供商建立了 API 市场，这也让云服务 API 推荐成为一种需要。但是目前的服务推荐的研究,大部分是关于QoS 预测、内容分析和用户偏好分析的内容，适用条件比较苛刻，对于目前规模还不算大的 API 市场环境来说还不适合。因此，本文希望在 API 市场达到足够规模之前，提供一种过渡性的、较为节约成本的、适宜的 API 推荐方法。本文针对 API 市场的现状，通过分析 API 推荐相对于其他推荐的不同和需要注意之处，提出了 API 推荐的重点是推荐用户需要的服务，以及使用用户行为对 API 进行推荐的想法，并论述了群智化的 API 推荐策略，并重点论述了两个主要的推荐模型：聚类与关联规则挖掘模型，以及评价推测模型。另外，本文通过已有模型和算法，在 MovieLens 数据集上进行了 API 推荐的实验。根据实验结果，本文提出了混合 API 推荐的策略，以及群智化的推荐系统效果评估方式。

关键词：服务推荐；API；群智化；云服务

Crowdsourcing Cloud Service API Recommendation Methods

ABSTRACT

More and more cloud service APIs have been provided with the development of Internet technology and Internet industry, especially in recent years, API markets based on cloud computing providers have been established, which makes cloud service API recommendation necessary. However, most current research on service recommendation focuses on QoS predication, content analysis and user preferences analysis, which needs strict application condition and is not suitable for current small API markets' environment. Therefore, this paper is intended to provide a transitional, cost-saving and suitable API recommendation method before the API market become large enough. Aim at the fact of API markets, this paper analyzes the differences between API recommendation and others, and the points for attention in API recommendation, and comes up with an idea that the aim of API recommendation is to recommend services that users need, and an idea that recommend APIs according to users' behavior. It also discusses crowdsourcing API recommendation strategy, especially two main recommendation models: clustering and association rules mining model, and ratings predication model. In addition, an experiment of API recommendation on MovieLens datasets is carried out with existing models and algorithms, and a hybrid API recommendation strategy and a crowdsourcing recommendation system performance evaluation method are presented in this paper according to the result.

Key words：Service Recommendation；API；Crowdsourcing；Cloud Service

目　录

1 绪论

1.1 研究背景和意义

1.1.1 云服务 API 及 API 市场

通常意义上的云服务 API（Application Programming Interface，应用程序编程接口）指通过网络提供某种特定服务的接口。在目前的环境下，这种接口以完全自由/有限制地免费/收费的方式提供,通常使用 HTTP/HTTPS 协议和 GET/POST 方式访问，往往遵循 RESTful 规则，需要传入特定的参数，传回字符串/JSON/XML 类型的值。

云服务 API 的一个优势就是：如果实现某个功能的 API 被提供，开发者不需要为实现它而自己写代码,只需要调用已有的 API,对传回的数据进行处理就行了。一方面,这类服务通常为高技术含量的服务（如人工智能）或法律、政策上有限制、风险的服务（如短信验证码、实名认证），开发人员如果自己写，技术上难以实现，还可能会有法律风险；将这样的服务交给有资质、能承担风险的人员或企业，让他们提供接口，让更多的人调用，才是权宜之策。另一方面，就算是技术和政策上比较方便的服务，如果经常被调用，也会让小型开发者的主机不堪重负，还不如调用有能力承载负荷的大型开发者的接口。最重要的是，使用云服务 API 能够显著降低学习成本，减轻初级开发者的学习负担，而且提供 API 也能够成为高级开发者的盈利来源。

随着互联网技术和互联网行业的发展，越来越多的云服务 API 被提供出来。由于各家服务提供商的 API 极为分散，不利于用户寻找比对，加上云计算提供商需要推广自己的产品，一些云计算提供商将多家 API 提供商提供的 API 集结起来，构建了类似于软件分发平台的 API 市场，消费者可以根据自己的需求选购需要的 API 服务，比如阿里云云市场、腾讯云市场、华为云市场。但是，这些云市场规模还处于较小的水平。

1.1.2 服务推荐

第三方 API 市场就性质而言，和诸如淘宝这样的第三方电商平台没有太大的差异。正如电商平台需要对商品进行推荐,API 市场也有这样的需求——服务推荐。

同时，由于“大众创业、万众创新”的推动和国内个人开发者的崛起，API 市场的潜在消费者规模十分庞大。因此，各家 API 提供商希望能够让平台提供推广服务，让自己的服务更容易让目标消费者发现，而 API 推荐服务也可以成为 API 市场的重要盈利手段。

1.1.3 群智化系统

群智化系统是一种使用大量的人群来解决系统拥有者的问题的系统[1]。但是，这种系统中的人群并非受雇于系统的拥有者，而是依靠自己的自愿性的贡献，或者是将自己的行为记录提供给系统。这种系统是群体智慧的重要体现，被应用于各个方面，如 Linux、Wikipedia，而且在推荐系统中也能够做到应用。

1.1.4 研究意义

传统的 Web 服务推荐以服务的可用性或者说是服务质量（QoS, Quality of Service）为主要的参考依据，但是 API 市场作为第三方平台，虽然可以通过一些方式获取到服务质量，但以此作为主要的推荐指标还是非常片面的：一些 API 虽然服务质量高，但是可能并不是用户需要的；一些 API 虽然性能不是非常好，但是用户觉得能用就行。这时候，可以借鉴普通商品的交易平台的推荐模式，基于用户对 API 的行为和评价等信息开展推荐。这就是群智化的云服务 API 推荐。

提出这种推荐方式，是考虑到目前的 API 市场发展还未成熟，而大多数已有研究为学术性研究，条件较为苛刻（如：需要足够多的 API 信息、需要特定的 API 信息格式），而且成本较高，并不适合目前尚在发展过程中，还未壮大的 API 市场。但是，用户行为的分析研究已经在很多领域有较为成熟的研究，因此可以依靠群体智慧，借此进行 API 的推荐。

相比于获取并评估服务质量，群智化推荐方法在技术上较为成熟，也较容易操作，成本也较低，而且有较强的可移植性和可扩展性，能够降低云计算提供商进驻 API 市场这个行业的技术和管理门槛，让云计算提供商不至于在 API 市场上投入大量的资金、人力物力。对于 IaaS（Infrastructure as a Service，基础设施即服务）、PaaS（Platform as a Service，平台即服务）、SaaS（Software as a Service，软件即服务）等形式的云计算服务平台，本来就拥有消费者的一些信息（如订购了什么服务、这些服务处于哪些价位、消费者的研发方向），可以在自己的 API 市场上迅速地部署、应用这样的推荐方法。这样的方法也可以作为一个过渡方法，在 API

市场还未发展壮大的时候应用，等到API市场积累一定程度的API和用户群体后，可以追加其他的推荐方法，或者是逐步替代成更加复杂但可靠性、准确性和效率更高的推荐方法。

1.2 国内外研究现状

1.2.1 Web服务的个性化推荐的目标

根据张秀伟等人（2013）的研究综述，Web服务的个性化推荐是为了解决信息超载问题而研究的[2]。随着互联网技术和互联网行业的发展，越来越多的云服务API被提供出来，能够让开发人员不需要大量的开发工作就能够完成需要的任务，也让服务更加模块化。但是，大量的API让API收录或分发平台面临服务信息过载的问题。这就让Web服务推荐显得非常重要，也让学者开始对其进行研究。

这些学者绝大部分是国内的学者，而且这样的研究绝大多数只是研究模型，成功应用案例较少。这样的情况有以下方面的原因：国内长期以来缺乏类似于ProgrammableWeb这样的第三方API收录平台，绝大多数API提供商都是自行宣传，而且它们实力雄厚，靠自己就可以很好地推广自己的API；API针对性强，而且长期以来，API数量远未达到信息超载的地步，用户在选择API方面不太需要个性化推荐；API推荐中，用户群体较为小众：对API推荐有较强烈需求的用户，大多数为大中企业，在选择API的时候，这些企业需要综合考虑多重因素，而小企业和个人开发者往往是找到有需要的功能的API就用，大部分考虑的是价格方面的问题。

1.2.2 群智化系统及需要解决的问题

A. Doan等人(2011)按照协作本质、架构、是否需要吸收用户、用户能够做的行为等维度对群智化系统进行分类，并认为群智化系统面临四个问题：

(1) 如何吸收、维持用户？

(2) 用户可以做出什么贡献？

(3) 如何组合用户的贡献，以解决目标问题？

(4) 如何评估用户和他们的贡献[1]？

对于API市场而言，用户提供他们的浏览、搜索、购买、评价等行为，系统根据这些行为信息进行API推荐。一般而言，API市场的用户来自市场对应的云计算提供商，用户的吸收、维持问题不属于推荐系统的范畴，而属于运营范畴。但是，

做到推荐系统的准确高效，虽然解决的是后两个问题，但是对用户的维持也有帮助——用户认为推荐系统能够推荐他们需要的物品，那么也就对 API 市场更加信任。这样一来，需要解决的问题就是后两个问题，而解决这些问题的论述也是本文的重要部分。

1.2.3 基于 QoS 的评估和预测的 Web 服务推荐

目前绝大多数关于 Web 服务推荐的文献都是将研究重点放在 QoS 的评估上的，某种程度上说，很多 Web 服务推荐的文献都是在讲述 QoS 的评估和预测。

唐明董等人（2018）认为传统的协同过滤方法和近年来流行的矩阵分解技术在 Web 服务推荐上各具不足，引入了一种通用的因子分解机模型到 Web 服务推荐中，并提出了质量感知 Web 服务推荐方法，并通过实验证明该方法在预测精度上优于其它协同过滤及因子分解推荐算法。同时由于该方法具有较低的时间复杂度，可以较好地解决大规模 Web 服务推荐系统的可扩展问题[3]。这个方法的实质是依靠相似的用户或服务来推测 QoS。

陆贝妮、杜育根（2019）对于 QoS 预测的现有方法存在数据稀疏和冷启动问题，提出了新的基于社区发现的 QoS 预测方法：首先通过谱聚类对用户进行社区划分，然后根据位置信息对 Web 服务聚类，最后利用改进的混合协同过滤方法预测 QoS 值。实验结果表明，该方法不仅能缓解上述问题，且相比其他预测方法有更高预测准确度[4]。

汪浩等人（2018）借助非参数统计学的 Bootstrap 技术，提出估计 Web 服务 QoS 值置信区间的方法；然后利用与当前 Web 用户相似的其他 Web 用户调用待预测 Web 服务的 QoS 历史数据，预测当前 Web 用户调用待预测 Web 服务的 QoS 值的置信区间。实验表明预测的 QoS 置信区间与估计的 QoS 置信区间的平均覆盖率超过 70%，最高达 76%[5]。

Y. Zhang 等人（2018）针对传统的 Web 服务推荐方案难以处理大量服务相关数据的问题，提出了 CA-QGS（Covering Algorithm based on Quotient space Granularity analysis on Spark），作为在大规模场景中精确推荐 Web 服务的可扩展手段。该算法首先基于用户在共同调用的 Web 服务的以往的质量体验来聚合用户和 Web 服务，再对聚合结果进行粒度分析，识别与目标用户和 Web 服务类似的用户和 Web 服务，使用协同过滤技术为目标用户预测目标 Web 范围的质量。他

们进行了大量的实验，证明该算法在推荐准确性和效率上均优于现有方法[6]。

1.2.4 基于服务描述的 Web 服务推荐

鉴于 QoS 的不确定性和难获取性，也有人在 Web 服务推荐上不仅看重 QoS，而且也兼顾其他方面，如用户的信息、API 的描述和评价、API 之间的关系。

有学者从 Web API 的描述文档入手进行服务推荐的研究。

李鸿超等人（2018）针对提高 Web 推荐的质量的问题，提出了一种融合多维信息的主题自适应 Web API 推荐方法 HDP-FM（Hierarchical Dirichlet Processes-Factorization Machines）为 Mashup 的创建推荐 Web APIs 集合。该方法以文档为语料库，利用 HDP 模型训练每个 Web API 的主题分布向量，预测每个 Mashup 的主题分布向量，通过计算出的相似度排序获取用于推荐的 Web API 集合。实验证明，该方法在准确率、召回率、F-measure 和 NDCG@N（Normalized Discounted Cumulative Gain at N）等方面具有较好的性能[7]。

M. Shi 等人（2018）则对于基于服务的原始功能性描述对服务进行选择存在的描述挖掘特征稀疏、无法训练良好模型、忽略区分特征的权重等问题，提出了文本扩展和基于深度模型的服务推荐方法，以及通过两种注意机制——功能注意机制和上下文注意机制——实现的基于长短期记忆网络（LSTM, Long Short-Term Memory）的模型。经实验验证，该方法相比于传统的 LSTM 模型，在 F-measure 上提升了 34%[8]。

1.2.5 基于用户的 Web 服务推荐

还有学者从用户角度入手进行研究。

曹步清等人（2015）提出了一种基于用户使用历史与信誉评价的 Web API 服务推荐方法，通过实验证明这种方式推荐的 Web API 用户兴趣度 DCG（Discounted Cumulative Gain）值高于 SR-Based 方法，服务信誉度 DCG 值高于 UI-Based 方法[9]。

李凌飞（2015）根据云计算环境下用户和服务特点，提出了一种服务之间合作关系的个性化推荐算法，旨在推测用户功能需求，以此为根据完成推荐。该方法根据用户使用服务的记录，分析服务之间的关系，通过这些关系进行推荐；分析用户对 QoS 的不同指标的关注程度，通过服务自身的 QoS 与用户偏好进行匹配，满足用户个性化需求；根据用户的活动，缩小对目标用户推荐的服务的范围。实验表明，

该方法有效地对用户进行了划分，提高了推荐的精度和速度[10]。

黄琳（2018）针对当前 Web 服务应用在移动互联网盛行，服务规模不断扩大发展的形势下出现的 QoS 数据可信性不高，Web 服务选择效率低下、服务推荐不准确的问题，提出了基于用户反馈评价的 Web 服务信誉度度量方法。该方法主要从时效性和全面性上对信誉度进行客观的度量，通过直接信誉和间接信誉的综合评价，以及时效性权重的考虑，削弱了恶意反馈对信誉度的影响程度[11]。

王泽源（2018）认为用户满意度对云服务的生存起到决定性的作用，考虑到用户在进行云服务选择时相似用户带来的影响，以及自身对于云服务选择的态度和期望，提出了一种综合考虑用户相似性及满意度的云服务排序方法，包括两个主要工作：在计算用户相似性时，提出了基于 Jaccard 因子的增强型排序相似性度量方法，可以更准确地度量用户间的相似性；在云服务预测过程中，提出了用户满意度度量方法，可以模拟用户在云服务选择时的态度和期望。最后设计了包含上述两方面的云服务排序推荐过程，实验表明，与现有方法相比，该方法预测精度更高，且能通过参数调整各类用户的偏好，具有很强的适用性和有效性[12]。

1.2.6 一般商品或服务的推荐的研究成果

在研究基于用户的 Web 服务推荐时，上述学者使用的指标，类似于网购平台的商品推荐中所用的指标。鉴于 API 市场这一 API 分发平台的形式的出现，实际上，API 市场上的 Web 服务更像是一种在电商平台上销售的商品。因此，对于群智化的云服务 API 推荐，除了 Web 服务推荐相关的文献外，还可以参考针对于一般商品或服务的推荐的研究成果。

熊丽荣等人（2018）针对个性化评价中评分数据存在稀疏问题和用户对于 top-k 项目排序列表的需求，提出了一种基于信任的面向 top-k 排序的推荐方法 BTRank，基于 LTR（Learning-To-Rank）方法，结合用户评分以及用户信任信息来构建项目排序模型，有效地提高对所有用户的 top-k 排序列表质量。同时，考虑到用户兴趣会随着时间演变而变化，设计了时间效应模型函数用于处理用户历史评分数据。实验证明该方法效果明显优于传统的推荐算法和同类 top-k 排序推荐算法[13]。

总之，对于 API 推荐，学术界已经有大量的理论研究，云计算提供商的崛起和 API 市场的出现也让它有用武之地，研究前景还是比较广阔的。API 市场的形式和网购平台类似，让类似于网购商品推荐的群智化云服务 API 推荐能够更好地应用

于实际。

1.3 论文内容与结构

本论文主要分为四个部分：

第一章和第二章为第一部分，主要介绍 API 市场的现状和运营模式，以及服务推荐（API 推荐）的概况、常用的推荐方法、推荐效果的评估方式；

第三章为第二部分，主要论述在群智化目标下的 API 推荐策略、推荐准备过程和一些重要的模型；

第四章为第三部分，介绍了使用前面的章节中提到的算法和模型进行推荐实验的过程和结果，并简单分析了结果；

第五章为第四部分，根据第三部分的实验结果，结合各算法的优势与不足，提出了一种混合 API 推荐的策略；考虑到现有数据集无法较好地评判推荐效果，提出了群智化的推荐系统效果评估方式。

2 云服务 API 推荐相关理论基础

2.1 API 市场

2.1.1 API 市场的现状

API 市场的发展与个人开发者与中小企业（尤其是后者）的增长息息相关。

随着大数据、人工智能等技术的发展，以及“互联网+”“智能+”的提出，各行各业对信息处理的范围也不断扩大——从 IP 地址定位到实名认证、征信核验等。这也催生出了一批 API 提供商，但是提供商需要客户来获得收益，于是就像商家通过淘宝等第三方平台开店一样，它们也通过第三方 API 市场提供自己的 API 服务——这样可以借助大平台提升自己的服务的曝光率，也可以使用平台的增值服务更有效地推广自己的服务。

目前国内的 API 市场有阿里云市场、腾讯云市场、华为云市场等。实际上它们并不仅仅提供 API 的服务，还提供其他的与云计算相关的第三方服务。这些 API 市场依托于自家的云计算平台，云计算平台的客户调用这些服务会更加方便，而且受客户基数大的影响，API 市场的商家能够获得更多的曝光度、流量和收益。这些 API 市场要求商家使用自己的服务搭建 API 服务，以监测状态与计费。

API 市场是新兴事物，而且 API 的门槛较高，所以各大市场内的 API 数量并不多。截至 2019 年 5 月 18 日，搜索阿里云市场的全部 API，有 1349 条记录；搜索腾讯云市场的全部 API 服务，有 299 个；而华为云市场上，API 服务有 112 个。

2.1.2 API 市场的运营模式

我们以阿里云市场为例。阿里云市场依托于阿里云，提供自有和第三方的各项云计算相关的服务，涵盖网站建设、软件、镜像、物联网等领域，API 市场仅仅是它的一部分。

阿里云市场提供了自己的 API，也提供了由对应商家提供的第三方 API。这些 API 涵盖金融理财、人工智能、生活服务、交通地理、气象水利、企业管理、电子商务、公共事务等方面。阿里云市场在首页会对一些 API 和 API 提供商（商家）进行推荐。

在搜索界面，用户可以根据成交数、价格、评分等指标对搜索结果进行排序。

同时，也会对一些服务进行推荐。但是，目前推荐的针对性并不高。图 2.1 是以“短信”为关键字的搜索结果，推荐的 Odoo 是企业资源计划系统，RabbitMQ 是消息队列服务器的环境，与短信没有直接关系。在商品的详细页面，用户能够根据自己的需求选择不同价位的套餐，而且能够看到评价。

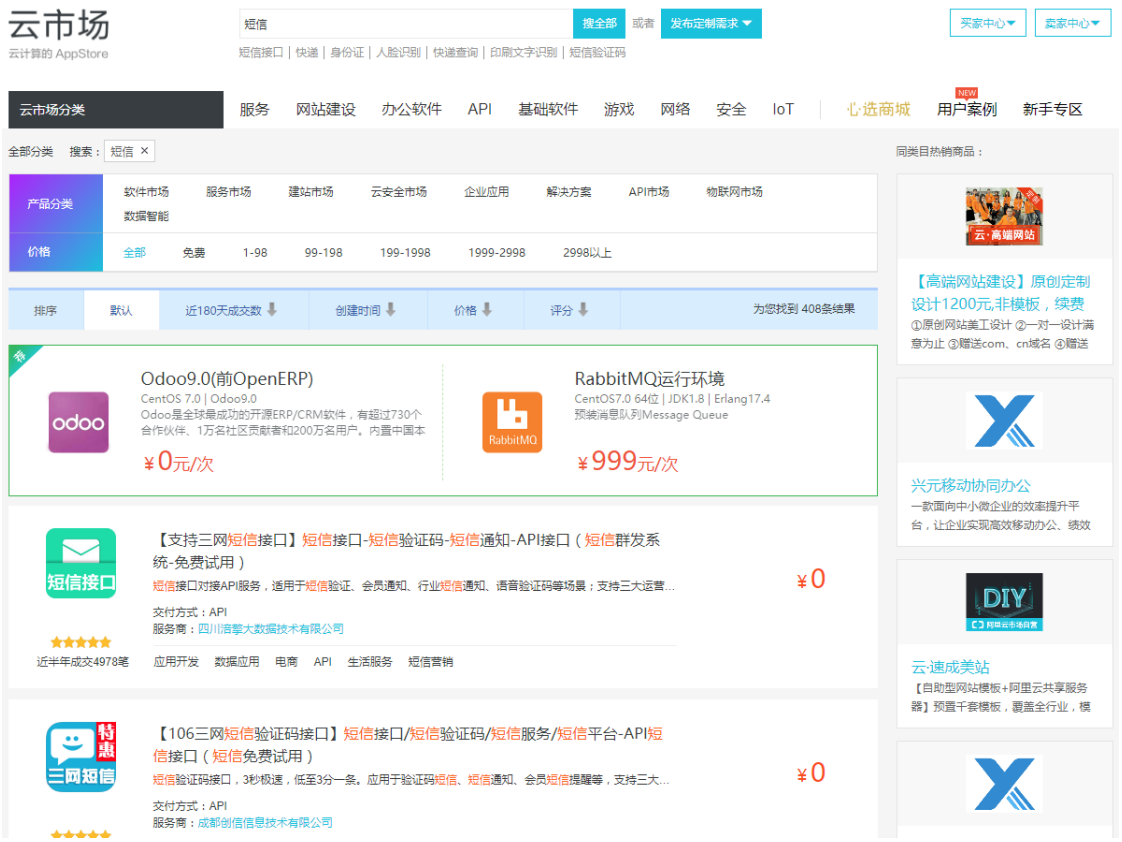

图 2.1　　阿里云市场的搜索与推荐

在 API 的详情页，有关于 API 的详细信息、套餐（包含价格和配额）、调用方式和评价，如图 2.2、图 2.3、图 2.4 和图 2.5 所示：

图 2.2　　某 API 的简介和套餐价位信息

图 2.3　　某 API 的部分调用方式信息

图 2.4　　某 API 的部分详细信息

图 2.5　　某 API 的部分评价信息

阿里云市场在商家进驻的时候会对其进行审核，并要求 API 使用阿里云的产品（如 API 网关）进行搭建。对于按次计费的 API，阿里云市场根据 API 的成功调用次数进行计数，如果 API 调用失败，则不会计入次数。对于按量计费的 API，依据用户调用 API 时消耗的资源量进行计费，商家可以自主设置。这些方式能够让阿里云对 API 的质量等信息进行监测。结合商家上架 API 时填写的详细信息和 API 的评价信息，以及用户的个人信息和浏览、搜索、购买等行为信息，可以做到对用户的群智化的云服务 API 推荐。

2.1.3 云服务 API 推荐相对于其他方面的推荐的特点

目前，推荐系统在许多场合中被应用，如商品、视频、电影、文章、新闻、应用和游戏。

用户在选购 API 的时候是有目标的，而且目的性强：比如，用户需要在项目中应用短信验证码，于是用户在 API 市场上查找、搜索关于短信验证码的 API，比较价格和质量，并完成购买。因此，与应用在电影、视频、文章等场合下的推荐系统不同，应用于云服务 API 的推荐系统，最需要关注的是“用户需要什么”，而不是单纯的“用户对什么感兴趣”，更不是单纯的“用户喜欢什么”。

一般情况下，API 的售卖方式为按请求数量或请求资源量付费，采取预付费方式，不同配额的套餐有不同的价位；用户会视自己的项目的情况，购买对应价位的套餐（试用版本的套餐除外），这使得当项目稳定的时候，用户更倾向于购买较长期的套餐。由于各商家的 API 接口使用方式不统一或不完全统一，考虑到开发难度，用户在选购某个 API 后，在配额充足的时候，不会购买其他功能类似的 API，除非该 API 质量未达到用户要求。因此，相比于电影、食品等快消品的推荐，API 的推荐需要考虑到用户已经购买的 API，尽量避免推荐与用户已购买且配额充足的 API 类似的 API。

另外，对于一个用户而言，相比于花在购买、浏览其他商品上的时间，花在购买 API 上的时间很少；同时，API 提供商和服务的数量也远小于网购平台上的商家和商品的数量；而且，就目前的情况而言，因为大多数 API 提供服务同质化、单一化，而且是供大于求的状态（API 提供商大多数都是技术雄厚的企业，能够提供高可用性和高质量的 API；绝大多数 API 也主要是企业客户购买），API 的数量的增长幅度也远小于网购平台上的商品增长幅度。因此对于 API 的推荐，可以将更多

的时间花在离线推荐上，提高推荐精度；或者是使用较简单的方式推荐，节省人力物力财力。

此外，还要注意一点：一般来说，API 市场是云计算提供商的新业务，而云计算提供商此前就积累了庞大的用户量，因此潜在用户非常广泛。换句话说，在目前的 API 市场中，用户数量远远超过物品数量。这导致用户行为数据高度稀疏。

2.2 云服务 API 推荐因素

2.2.1 QoS 及其预测值

QoS 是判断服务质量的一个指标。它包括服务的信誉度、响应时间、价格、可靠性、稳定性、完整性、可用性和安全性等指标，将这些指标作为参数进行评价[11]。用户可以根据自己的实际情况，对这些指标进行加权评估，以选择合乎自己需求的服务。

QoS 的各项参数受网络环境等因素的影响较为严重，而且不同位置的用户在同一时间对同一服务观测的 QoS 值也有所不同。因此，QoS 需要通过 API 的调用记录来计算与预测。但是，由于涉及到商业利益，大多数传统的 API 提供商并不会向用户或第三方提供这样的数据。不过，在 API 市场中，由于 API 市场对 API 的搭建方式做出了硬性要求（必须使用市场所属云计算提供商的产品进行搭建，并向市场传回数据），API 市场能够较为方便地获取服务的各项指标，以计算、预测 QoS。另外，由于 API 构建于 API 市场所属云计算提供商的产品上，再加上目前的 Web 应用发展日渐壮大，所以在 CDN、负载均衡等技术的支持下，在同一云计算提供商上的用户的位置可视为相同，这也大大降低了 QoS 的计算与预测难度。

2.2.2 用户信息

对于 API 市场来说，其所属的云计算提供商拥有大量的用户数据（如这个用户拥有的产品、用户从事的行业、用户是个人开发者还是企业开发者，用户花费的价格、用户的社会关系）。因此市场可以针对用户的这些数据，评估用户的可接受价位、对服务质量的要求、需要什么 API 服务，向用户推荐可能感兴趣的 API。

2.2.3 服务描述与评价

API 本身是拥有某项功能的服务，因此提供商会向用户描述它的功能、调用方法和收费方式等信息。我们也可以通过这些描述向用户进行 API 的推荐。这需要

对获取到的这些内容进行自然语言处理（NLP, Nature Language Processing），提取关键字，判断其描述与用户需求的相似度，进而根据相似度对服务进行排序推荐。

已购买或使用 API 的用户能够对 API 的质量进行评价。如同对商品的评价，这些评价也能够作为服务推荐的很重要的指标。评价的方式有评分和文字评价，前者在数据处理上较为方便，后者通常需要进行 NLP。但是，评价的主观性较大，而且有恶意评价的可能性，需要对其进行过滤。

2.2.4 用户的行为记录

如果用户浏览 API 的详细信息，并在页面上做了一段时间的停留，或者是搜索了某类关键词，并查看了搜索记录，说明用户可能对类似的 API 感兴趣；如果用户购买了某个 API 产品，系统也可以根据其他类似用户的购买记录，推断用户可能还会买哪些 API。

类似的经验早已用于商品、内容、广告等的推荐上，由于 API 本身也是一种商品，所以也适用。不过，API 相比于其他商品，有一些方面不同，在推荐策略的制订上需要注意。

2.3 云服务 API 推荐相关算法

目前关于云服务 API 的推荐相关算法，比较传统的有基于内容的推荐方法和协同过滤。此外，还有 Apriori 算法、Word2Vec 算法、PersonalRank 算法等也可以应用于 API 推荐。以下对这些算法进行简要的介绍。

2.3.1 基于内容的推荐方法

该方法根据用户过去喜欢的项目，推荐与这些项目类似的项目。它提取用户与项目的特征，对项目内容特征与用户兴趣特征进行匹配，再推荐内容与用户感兴趣项目类似的其他项目，生成推荐列表。该方法实现较容易，可解释性强，可以对评分不足的项目进行推荐，但是它无法区分项目之间的质量，推荐项目的多样性也较差[14]。

2.3.2 协同过滤推荐方法

协同过滤算法起源于 1992 年，最初被施乐公司用于个性化定制邮件系统，直到现在还被广泛应用，是目前最流行的个性化推荐方法。群体智慧也是在 1994 年

第一次被引入协同过滤中，使用大量的人群提供的知识（如行为记录、评价）来进行推荐，做到了省时省力，而且具有较高的推荐精度[15]。

按照处理方式的不同，协同过滤分为基于邻域的协同过滤和基于模型的协同过滤。

按照处理对象的不同，基于邻域的协同过滤分为基于用户的协同过滤和基于物品的协同过滤。前者寻找与用户相似的用户，根据这些用户的选择来预测用户感兴趣的物品；后者寻找与物品类似的物品，根据选择这些物品的用户的属性来预测用户对物品的感兴趣程度。它们需要先建立用户-物品矩阵，计算用户或物品之间的相似度，通过最相似的若干邻居评分预测用户评分，生成推荐列表。它的可解释性和扩展性强，实现起来较容易，但不适用于数据稀疏的场合。它的缺点是冷启动问题（如果没有历史数据就无法分析）和新用户问题（新用户没有评分就无法进行推荐）。

基于模型的协同过滤是最主流的协同过滤类型，它基于对以往的数据学习得出的模型预测用户与项目之间的偏好关系。常用的模型有贝叶斯网络、聚类模型、潜在语义模型、矩阵分解模型、马尔可夫模型等。这种方法可以有效解决数据稀疏问题，但是可解释性差，模型训练较费时间[14]。

2.3.3 Apriori 算法

Apriori 算法是比较传统的关联规则挖掘算法。假设某事务数据库中有若干事务，每个事务为由物品组成的列表，代表同一时间购买物品的记录，定义最小支持度，对该数据库应用 Apriori 算法的过程如下：

(1) 首先，找出候选 1 项集。扫描数据库，对每个物品计数，其数目为支持度计数，或者说是绝对支持度，绝对支持度与事务之比为相对支持度；

(2) 筛选支持度大于最小支持度的物品，得到频繁 1 项集；

(3) 将频繁 1 项集中的各项两两组合，所得集合得到候选 2 项集；

(4) 扫描数据库，对候选 2 项集内各集合计数，筛选支持度大于最小支持度的集合；

(5) 重复 3、4 步（要保证集合的子集也是频繁项集），直到无法继续，由此得到所有的频繁项集。

2.3.4 Word2Vec 算法

Word2Vec 是从大量文本语料中以无监督的方式学习语义知识的一种模型，被广泛应用于自然语言处理中。它将语料中的词汇映射到空间向量上，语义接近的词汇在该空间内的距离也接近。Word2Vec 有两种模型：Skip-Gram 和 CBOW。其中，Skip-Gram 模型输入中心词，输出预测的上下文；CBOW 模型输入上下文，输出预测的中心词[15]。因此可以活用该模型，将其应用到行为分析上。将用户的行为（如搜索、浏览、购买、评价行为）序列作为语料，其“词汇”为目标物品（如购买的物品），让 Word2Vec 进行训练，可以根据用户的行为推断关联物品，进行推荐（同理，也能够推断类似物品，但不属于 API 推荐的范围）。Spotify、Airbnb 和 Yahoo 都进行了类似的研究，并取得了较好的成果[16][17]。

2.3.5 PersonalRank 算法

PersonalRank 算法是将用户与物品之间的关系（如评分、观看、购买）映射为无向二分图。首先将待推荐的用户的节点的 PR 值设为 1，其他节点的值设为 0，从用户的节点出发，在图中随机游走，以α的概率游走到出边（如果游走，在一个节点的所有出边中以相等概率随机选择一个出边游走过去），或以$(1-\alpha)$的概率回到之前的节点。每次游走时按照概率转移 PR 值，游走过程迭代若干次后，即可根据各节点的 PR 值（相对于用户节点的重要性程度）确定推荐物品。

如果单纯以二分图进行计算，时间复杂度会非常高，幸而该算法还有矩阵形式，能够一次运算完毕。将用户数为m、物品数为n的二分图映射为转移矩阵$M_{(m+n)\times(m+n)}$（映射方式如图 2.6 所示）。

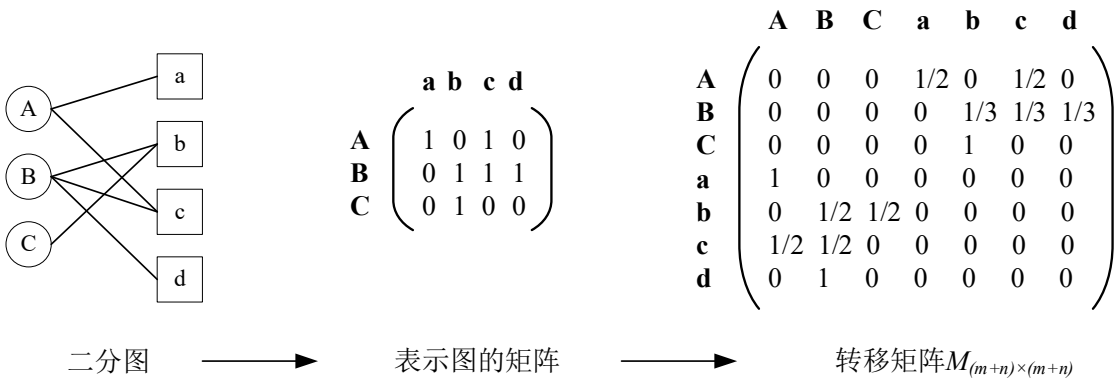

图 2.6　从二分图到转移矩阵

可得

$$r = (1-\alpha)r_0 + \alpha M^T r \tag{1}$$

其中$r_{(m+n)\times 1}$表示各节点的 PR 值，$r_{0_{(m+n)\times 1}}$表示初始 PR 值，以待推荐用户的节点值为 1，其余为 0。

上式可变形为

$$(E-\alpha M^T)r=(1-\alpha)r_0 \tag{2}$$

$$r=(E-\alpha M^T)^{-1}(1-\alpha)r_0 \tag{3}$$

其中E为单位矩阵。若$r_0=E_{(m+n)\times(m+n)}$，则可以得到所有推荐结果。另外，如果仅仅是排序，不需要$(1-\alpha)$。因此，可以得到任意一个节点到其他所有节点的供排序的 PR 值的矩阵：

$$r_{all}=(E-\alpha M^T)^{-1} \tag{4}$$

其中$r_{all_{(m+n)\times(m+n)}}$中，每一行代表对应的节点，列上的数字即为从所在行代表的节点到所在列代表的节点的供排序的 PR 值。

2.3.6 矩阵分解方法

矩阵分解算法属于基于模型的协同过滤算法[18]，其目标是寻找一种方式，能够将给定矩阵分解为若干个矩阵（通常是两个或三个），使其运算结果能够尽可能地还原原矩阵；对于推荐系统，需要在还原原矩阵的同时能够较好地填充缺失值。有很多方式进行矩阵分解，常用的有奇异值分解（SVD, Singular Value Decomposition）算法和隐语义模型。隐语义模型中，常用的算法有随机梯度下降（SGD, Stochastic Gradient Descent）和交替最小二乘法（ALS, Alternating Least Squares）。

下面以 SVD 算法为例，对矩阵分解进行讲解。

一个矩阵$A_{m\times n}$经 SVD 分解后，输出三个矩阵：$U_{m\times m}$、$\Sigma_{m\times n}$（Σ中除主对角线上的值外，其余的值全为 0）和$V_{n\times n}$：它们的关系如下：

$$A=U\Sigma V^T \tag{5}$$

其中，Σ中对角线上的值构成奇异向量。这些值从大到小排序，且减小速度很快。推荐系统中，我们不必使用全部的奇异值，而是选取最大的k个奇异值重组矩阵，使[15]

$$A_{m\times n}\approx U_{m\times k}\Sigma_{k\times k}V^T_{k\times n} \tag{6}$$

以此节约存储、计算时间，减轻服务器负担。k值可以按照奇异值占比确定，

具体方法在 4.4.6 节有介绍。

2.3.7 混合的推荐方法

实际应用中，推荐方法不会是单一的，而是将多种推荐方法进行混合，取长补短进行综合推荐。比如：对于评价较丰富的项目，使用协同过滤方法；对于评价不足或新的项目，使用基于内容的方法；两者的推荐结果按一定规则穿插呈现，不仅能让用户发现感兴趣的项目，而且能够让更多不知名（长尾）的项目被发现。

2.4 云服务 API 推荐评价标准

2.4.1 预测准确度

预测准确度是云服务 API 推荐最重要的评价标准，直接体现云服务 API 推荐的质量。在服务评分或 QoS 预测中，可以使用平均绝对误差MAE或均方根误差$RMSE$来反映预测准确度。若用户i对 APIj的 QoS 的预期值为r_{ij}，预测的 QoS 值为$\hat{r}_{ij}$，预测值数量为N，则

$$MAE = \frac{\sum_{i,j}|r_{ij} - \hat{r}_{ij}|}{N} \tag{7}$$

$$RMSE = \sqrt{\frac{\sum_{i.j}(r_{ij} - \hat{r}_{ij})^2}{N}} \tag{8}$$

MAE越低，表示推荐系统的预测精度越高。而$RMSE$加大了对预测不准的值的惩罚，对系统的评价更加严谨苛刻。这两个指标也可以用于评分的预测准确度的度量。

而对于 API 的 Top-N 推荐，使用准确率P、召回率R和 F-measure（F，为P与R的调和平均数）来反映预测准确度。若用户最终使用的 API 列表为L_t，系统推荐的 API 列表为L_r，则[2]

$$P = \frac{|L_t \cap L_r|}{L_r} \tag{9}$$

$$R = \frac{|L_t \cap L_r|}{L_t} \tag{10}$$

$$F = \frac{2PR}{P + R} \tag{11}$$

2.4.2 覆盖率

覆盖率描述了推荐系统对 API 的推荐能力，即推荐系统能够推荐出的 API 列

表$R(u)$占全部 API 集合S的比率：

$$Coverage = \frac{|R(u)|}{S} \tag{12}$$

覆盖率需要结合预测准确度使用[2]。

2.4.3 信任度

信任度指用户对推荐系统的信任程度，通常使用问卷调查法得出。推荐系统需要让用户感到其实用性，而不是广告推介平台从而使用户反感。提高信任度的方法是让推荐算法或规则透明化，提供完整的推荐原理解释[2]。

2.4.4 多样性

由于云服务 API 推荐需要让用户能够更大可能地接触到感兴趣的 API，而不是热门的 API，避免马太效应，因此需要注重推荐 API 的多样性。多样性描述了推荐的 API 之间的不相似性。若两个 API（物品）s_i与s_j之间的相似度定义为$s(i,j) \in [0,1]$，则用户u的推荐列表$R(u)$的多样性的计算公式为：

$$Diversity\big(R(u)\big) = 1 - \frac{\sum_{i,j \in R(u), i \neq j} s(i,j)}{\frac{1}{2}|R(u)|(|R(u)| - 1)} \tag{13}$$

推荐系统的整体多样性可以定义为所有用户的推荐列表多样性的平均值[2]：

$$Diversity = \frac{\sum_{u \in U} Diversity\big(R(u)\big)}{|U|} \tag{14}$$

2.5 本章小结

本章详细介绍了 API 市场的现状和运营模式，以证明使用群智化的方式进行云服务 API 推荐是可行的；指出了云服务 API 的推荐相比于其他的推荐，有什么不同之处，需要注意什么；列举了云服务 API 的推荐因素，介绍了它们在 API 推荐中的重要性；简单介绍了云服务 API 的常用的推荐方法和它们的原理、步骤；介绍了 API 推荐的评价标准，并给出了它们的意义和计算方法。

3 群智化云服务 API 推荐方法

3.1 假设与推荐策略

3.1.1 假设

为了制定推荐方法和构建推荐系统，根据实际情况，首先对用户行为做出以下假设：

(1) 用户不会选择与已购买服务高度相似（尤其是功能相同）的服务，除非已购买的服务快到期或配额快使用完，或对已购买的服务不满意；

(2) 如果用户此前购买了一些服务，则用户会购买与之匹配的服务；

(3) 用户对服务的选择与类似用户或同类型用户对服务的选择相近。

此外，为明确目标，假设 API 市场在以下情境中：

(1) 由云计算提供商驱动，强制要求 API 提供商使用云计算提供商的产品进行 API 的搭建，以获取调用信息；

(2) 和云计算提供商的服务集成度高，云计算提供商的用户能够很好地知晓市场的存在，如果已购买云计算提供商的服务，用户会优先选择同一云计算提供商驱动的 API 市场；

(3) 推荐系统尽量群智化，即以用户的行为为主要的参考要素。

3.1.2 推荐策略

对于推荐，有两种情况：未搜索情况下的推荐和搜索情况下的推荐。

本文中，搜索情况下的推荐仅考虑用户搜索相关 API 的情景，而不考虑用户给出明确需求（如需求文档）的情景——当用户给出明确需求的时候的服务推荐方法已有很多。在搜索情况下，推荐系统应当首先根据搜索条件的相关性过滤 API，为 API 排序，赋予权重；再使用这个列表进行个性化推荐。简而言之，在搜索的情况下，应当以搜索条件的相关性为主。

未搜索情况下的推荐就是用户在访问市场的时候，并没有进行搜索，就像许多电商平台会在首页推荐产品一样，用户并没有直接进行输入。此前的研究很多都是在用户提出明确需求的情况下进行推荐的，对这种用户没有明确提出需求的情况鲜有研究。而随着个人开发者的增长，这样的情景会越来越常见。本文主要论述的

就是这种情况下的推荐——搜索情况下的 API 推荐在过滤 API、排序之后，基本与未搜索情况下的 API 推荐相同。

云服务 API 推荐，重点在于猜测“用户需要什么”。如果以群智化的方式进行推荐的话，有以下思路可以实现这个目的：

一是推荐配套（关联）服务。用户在 API 市场购买服务，一般而言是为了开发、运营项目，势必要购买配套的服务（包括 API）。这时，可以通过所有用户或类似用户的购买记录，挖掘关联规则，判断用户需要什么服务，并进行推荐。可以通过 Apriori、Word2Vec 等方式实现。

二是推荐类似用户购买的服务。实现这个方式的方法很多，如协同过滤、PersonalRank。

三是推荐用户现在感兴趣的服务。这方面可以通过用户对服务的浏览（但未购买）记录、搜索记录等等来分析，将这些项目进行聚类，根据聚类中心为用户推荐更多的类似服务。

上面的思路并不是孤立的，而是可以结合使用的，如挖掘类似用户同时购买的服务。也可以结合使用其他的推荐方式以加快推荐速度、提高精度。

以上解决的是“提供需要的服务”的问题，除此之外，还要保证服务是高质量的。

一个有效的方式是测定 QoS 值，根据 QoS 值为候选服务进行排序，选择符合用户需求的服务进行推荐。云计算提供商驱动的 API 市场中，云计算提供商能够比较容易地获取各 API 的调用记录，并评估、预测某些 QoS 值，这一方面已经有非常多的研究。但是，QoS 可以通过某些方式人为地提高（如伪装请求、错误时返回 HTTP 200 这样的“正常”状态），而且如果仅仅依靠云计算提供商自动记录分析的话，某些 QoS 值无法很好地评估、量化（如安全性）。因此，不能仅仅依靠 QoS 值对服务进行排序。

还有一种方式就是通过用户的评价信息判定 API 的质量。评价包括显性评价和隐性评价。显性评价包括评分、评论、其他用户对评论的表态；不同的 API 市场包括的显性评价形式可能不一样。隐性评价可以从用户的购买、使用行为分析得出，如：用户购买某项服务后，如果在配额充足时便购买其他功能高度相似的服务，那么可以判定用户对之前购买的服务不满意。对于显性评价，除评分和其他用户的表态外，需要考虑时间效应——由于 API 的更新迭代，新的 API 可能与旧的 API 在

QoS 等表现上有所不同，因此越新的评价参考价值越大（这时需要防止用户通过恶意重新编辑评价来增大权重）。此外，还要考虑到评价造假等问题，这方面也已经有大量的研究——一个简单的方式是参照用户调用 API 的记录，分析评价与调用记录的对应性（如：如果多次成功调用，但用户的评价表现出“没法调用”，那么可以证明该评论是虚假评价）。对于评论，可以对其进行 NLP，挖掘关键词，判断信息量，确定其参考价值——通常来说，信息量越大，评论的参考价值就越大，权重也应该设置地越大。另外，评价不一定是一维的，还可能是多维的（如：费用、时延、返回结果——这些在一些文献中被认为是 QoS 的指标，但这里这些指标偏主观，由用户主动提供，而不像上文中所述的 QoS 指标一样能够自动记录），这时候可以通过用户的行为来推断用户关注的方面，进而推荐某些维度评价较好的 API。但是，购买某项服务的用户在用户群体中是极少数的，进行评价的用户就更少了，这使得评价矩阵会非常稀疏。对这样的问题，可以通过协同过滤、矩阵分解等方式，预测用户的评分，根据评分推荐用户大概率满意的服务。

同时，由于 API 调用方式不同，如果用户更换 API 或者是为多个功能高度相似的 API 写调用方法，会增大开发难度和复杂性。因此，用户不会去选择与已购买、满意且未到期的 API 相似的 API。因此，在推荐过程中，需要排除已购买的服务和与这类服务高度相似的服务，使推荐更有意义。

此外，还要考虑冷启动问题。对于 API 的冷启动，可以通过新 API 根据内容优先推荐、专家评定等方式解决。因为 API 的特殊性，用户通常会信任资历较老的 API，而对于新 API 常常持观望态度。所以最好使用专家评定的方式解决 API 的冷启动问题。对于 API 市场而言，实现这样的方式的途径比较多，如算法大赛，可以吸引初创企业和学生参与，获得曝光率和流量；API 市场的运营商也可以借此机会对外界推广自己的服务（如 API 网关、云服务器、云数据库），实现双赢；这样的方式也可以增加用户对 API 市场的信任度。对于用户的冷启动问题，可以根据用户注册时或回访时提供的信息，通过为用户推荐同类型用户购买的服务来解决。

至此，推荐系统的较为概括性的结构图如图 3.1 所示：

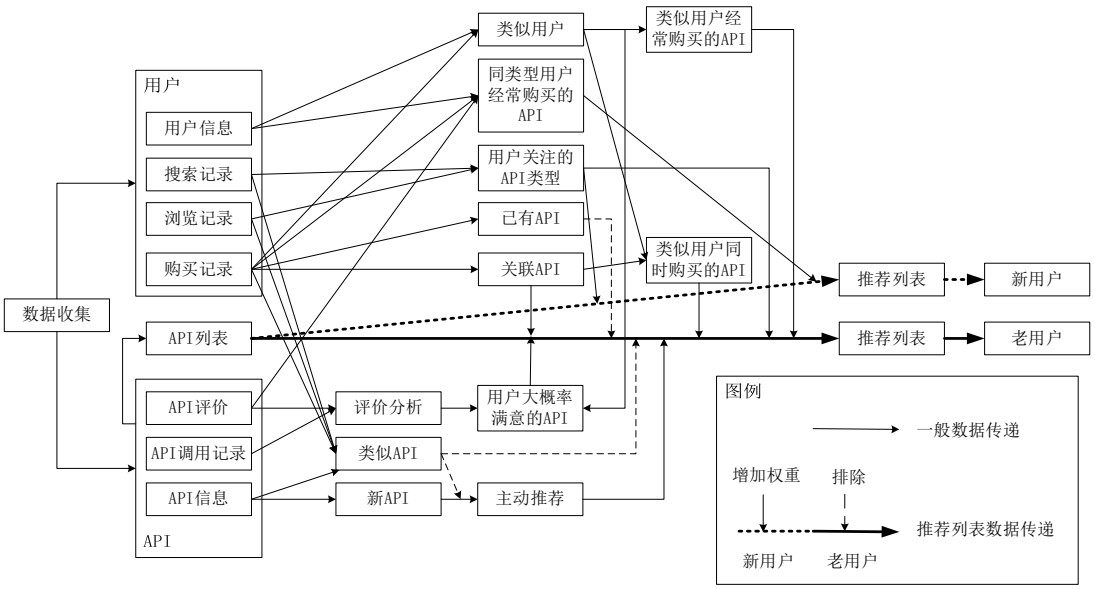

图 3.1　　推荐系统的较为概括性的结构图

不过，由于平台制度等原因，图 3.1 中的信息不一定能够完全收集到，可以适当做取舍；也可以进行扩增，以增加推荐的质量。

解决了上面的问题，就相当于解决了 1.2.2 节中提到的群智化系统面临的问题中的后两个问题。

3.2　数据准备

3.2.1　日常记录的数据

对于云计算提供商驱动的 API 市场，如果希望进行服务推荐，就要对用户和服务数据进行充分的获取和存储，以备接下来的模型建立、推荐，也可以用于其他方面的分析。需要存储的数据包括但不限于：

(1) 服务的名称、内容、关键词等信息；

(2) 服务的评价信息，包括哪位用户在什么时候对服务进行评价，评分多少，文字内容的评价又是什么；

(3) 用户的类型、领域等信息；

(4) 用户的服务购买记录和使用情况，包括服务的购买时间、金额、数量等信息。这里的“服务”，不仅仅是 API，还包括其他云服务，如服务器、域名；

(5) 用户的搜索记录和服务信息页面的访问记录，包括访问时间、访问时长等信息；

(6) 服务的调用记录，包括用户、调用时间、响应时长、状态码等信息。

由于不同 API 市场的具体情况不同，本章中不会介绍具体的数据获取方式。

3.2.2 数据初步处理

设用户列表为U，通过搜索、浏览与服务购买等记录，生成以下矩阵：

(1) 用户近期搜索矩阵U_{search}

矩阵的每一行代表用户，每一列代表服务。值根据搜索时间、搜索频率等信息生成，介于 0 和 1 之间；值越大，表示搜索与该服务相关的内容越频繁，或者是时间距离现在越接近。

(2) 用户近期浏览矩阵U_{browse}

矩阵的每一行代表用户，每一列代表服务。值根据浏览时间、浏览频率等信息生成，介于 0 和 1 之间；值越大，表示浏览该服务页面越频繁，或者是时间距离现在越接近。

(3) 用户近期购买矩阵U_{buy}

矩阵的每一行代表用户，每一列代表服务。值根据购买时间、购买量、购买价格等信息生成，介于 0 和 1 之间；值越大，表示购买该服务越多、花费金额越高、时间距离现在越接近。

(4) 用户服务使用量矩阵U_{usage}

矩阵的每一行代表用户，每一列代表服务。数值介于 0 与 1 之间，为购买服务已使用的百分数除以 100。当值为 1 时说明服务已用尽或没有购买过该服务。

(5) 用户相似度矩阵$U_{similar}$

行列为用户，值为他们之间的相似度，介于 0 与 1 之间。相似度根据用户所在领域、用户类型和上面的矩阵得出。可以得出对于用户$m,n \in U$，设他们之间的相似度为$u_{similar(mn)}$，有：$u_{similar(mn)} = u_{similar(nm)}$与$u_{similar(mm)} = 1$。

设 API 列表为S，生成以下矩阵：

(1) API 评价矩阵S_{rate}和用户评价权重列表W_{rate}

对于这两个矩阵，每一行代表 API，每一列代表用户，值介于 0 与 1，在

S_{rate}中表示总体评价，在W_{rate}中表示 API 中每一位用户评价的权重。

评价的来源有评分和文字评价。如果有文字评价，可以通过 NLP、与调用记录综合分析等方式对评价内容进行分析，判断评价的信息量、真实性和参考价值。也可以根据用户对其他 API 的评价数量和质量，确定用户的评价的参考价值。以上构成评价的权重。

(2) API 相似度矩阵$S_{similar}$

行列为 API，值为它们之间的相似度，介于 0 与 1。相似度根据 API 的名称、类别和关键词等方面得出。

确定上面一些矩阵各项的值时，可以参考一些模型对值进行处理，比如熊丽荣等人提出的资源衰减的时间效应模型，模型如下：

$$h(\Delta t,\lambda) = e^{-\lambda \Delta t} \tag{15}$$

其中h为学习后经Δt时长后剩余记忆比例，λ表示遗忘速率，不同人的遗忘速率不同，根据λ值不同可以分为念旧型和多变型两种人群[13]。应用到矩阵值的处理时，越新的浏览记录、搜索记录、购买记录等，权重越大。

除此之外，矩阵具体的生成方法在本章并不做太多陈述，因为不同的系统能够获取到的数据及其重要性不一样，需要具体问题具体分析。而且，上面的矩阵并非都是必需项目，而且如果可以分析更多元素，也可以生成更多的矩阵。

3.3 推荐模型

对于 API 推荐，其中的一些任务（如评价分析、类似 API 过滤、关联 API 推测）可以作为模型对待。这里重点论述两种模型——聚类与关联规则挖掘模型，以及评价推测模型。这两种模型构成群智化云服务 API 推荐方法的重要部分。

3.3.1 聚类与关联规则挖掘模型

在时间层面上对某组用户的购买记录进行聚类分析，找到所有包含用户同时购买的服务的组合。然后对这些组合进行关联规则挖掘，找到所有关联规则。然后根据特定用户近期的购买记录，套用关联规则，生成用户可能需要的 API 的列表。

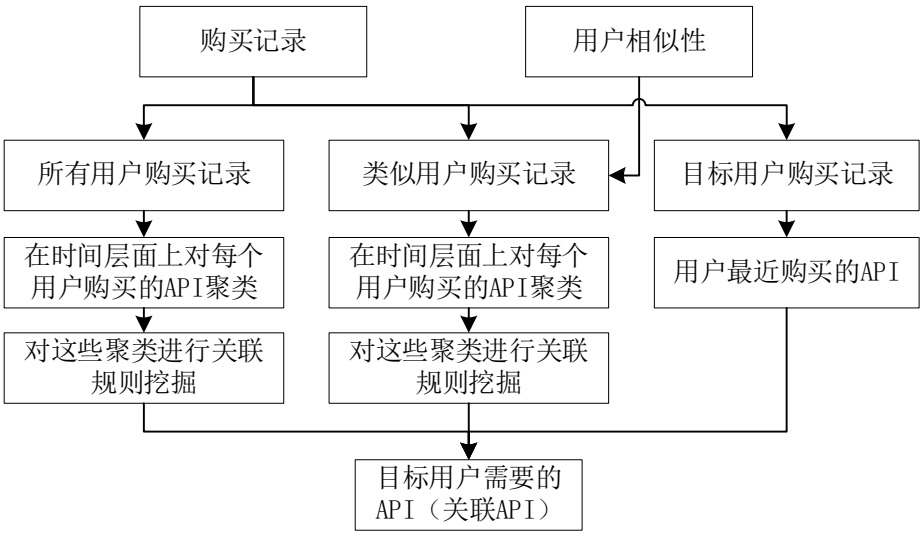

图 3.2　关联规则的挖掘方法

上面提到的“某组用户”，可以是全体用户，也可以是与特定用户类似的用户，或者是两者都有。如果两者都有的话，与特定用户类似的用户的关联规则应该占更大的权重。

传统的关联规则挖掘，场景类似于超市，事务数据库中的一个事务相当于顾客单次的购买记录，其中包含多个项。但是在 API 市场中，由于项目的工期不确定而且项目的细节可能有所改变，用户不一定会一次性购买项目所需的所有服务，而是分开购买或按需购买，这就导致 API 市场中用户的单次购买记录内项数极少（往往只有一项），无法进行关联规则挖掘。所以，寻找关联服务，首先需要通过购买记录进行物品在购买行为上的聚类。这里的聚类分析是在单维指标（时间）上的分析，而且簇的数量不定，所以可以使用 Mean Shift 算法解决。Mean Shift 算法相比于 k-means 等等的聚类算法，其一大优势是：Mean Shift 不需要定义要分成的簇的个数[18]。这种特性非常适合进行非一并购买的购买记录的聚类。因为一个项目的工期可能会长达一个月，所以在确定该算法的模型参数时，尽量能够让 30 天内的记录聚在一个簇内。

对于关联规则的挖掘，传统的 Apriori 算法在支持度较低的时候，会花费大量的时间和空间；但在商品种类和购买记录繁多的时候，某种组合的支持度通常也会很低，需要确保低支持度才能挖掘出更多的规则，以及筛选出更多高置信度的强关联规则。这时候可以采用 Word2Vec 算法来代替。Word2Vec 算法本来是用于自然语言处理的算法，输入给定的分词后文本，计算每个词在高维空间中的特征向量，以

此来计算词语的关联度[16]。除此之外，它还可以进行上下文与中心词的预测。根据这种特性，可以将用户同时购买的服务序列作为一段文本输入，把项目编号看作“词语”，交给 Word2Vec 算法来处理，从而获取到可能的关联 API。

此外，根据黄昕等人（2019）提到的方案，通过基于物品的协同过滤算法，如果将“物品相似度”定义为一个与“两个物品同时被购买的概率”相关的指标，也能做到类似的效果。而且，在此基础上，也可以引入评价处理和对热门物品降权的措施[16]。

这个模型适用于购买次数较多的用户，对于新用户完全无法使用，而对于购买次数少的用户，找不到符合关联规则的物品的可能性比较大。

3.3.2 评价推测模型

对于显性评价，由于每个用户的评分标准不同（有的人打分较为宽松，倾向于给物品打高分；有的人要求非常苛刻，倾向于给物品打低分），所以在推测评价前，可以将每个用户的评分减去这个用户的评分的平均值，便于更好地进行计算。在这种情况下，大于等于 0 的评分被视为较正面的评价，低于 0 的评分被视为较负面的评价。

对于 API 来说，用户的评价相当于用户对 API 的满足程度，或者说是实际水平与期望水平之差。对于不同类别的用户，其满足程度也不一样：个人开发者开发个人小项目时，所需 API 可能能用就行；而企业用户需要处理大量的并发请求，所需 API 在响应速度上会有很高的要求。这样一来，有些人满意的 API，有些人不满意。在处理评价时，应当重点考虑类似用户的评价。

对于评价的推测，主要有协同过滤与矩阵分解两种方法（实际上矩阵分解属于基于模型的协同过滤）。下面重点介绍协同过滤方法。

因为 API 市场中用户数量远远超过服务数量，所以 API 推荐不适用基于用户的协同过滤，而由于 API 的数据信息较为稳定，两两之间的相似度变化不大，所以适用基于物品的协同过滤[16]。

但是，传统的基于物品的协同过滤是计算物品之间的相似度，将相似度作为权重，根据用户已经评价的物品对为评价物品进行评价的预测，进而推荐与购买物品相似的物品。但是对于 API 这样的服务而言，如果推荐相似服务，会导致推荐大量功能与用户已经购买的服务相同的服务，不符合要求。因此，对于购买记录，可以

参考黄昕等人提到的方案，采用将两件物品同时购买的情况作为物品相似度的方法，也就是物品之间在购买上的相似度。

首先，计算物品两两之间同时被购买的次数。使用之前的聚类作为“同时购买”的依据，生成物品两两之间同时被购买的次数的矩阵。然后，根据这个矩阵计算物品之间在购买上的相似度w_{ij}：

$$w_{ij} = \frac{|N_i \cap N_j|}{\sqrt{|N_i||N_j|}} \tag{16}$$

其中，$|N_i|$为购买物品i的用户数，$|N_j|$为购买物品j的用户数，$|N_i \cap N_j|$为同时购买物品i、j的用户数。

然后根据相似度计算可能的评分，计算方式为：对于所有用户u已购买物品集合$N(u)$中的物品j，将用户对它们的评分r_{uj}乘上这些物品与该物品的相似度w_{ji}，结果之和为预测分数[16]：

$$\hat{r}_{ui} = \sum_{j \in N(u)} w_{ji}\, r_{uj} \tag{17}$$

设用户数量为m，物品数量为n，物品间在购买上的相似度组成矩阵$W_{n\times n}$，用户对物品的评分矩阵为$S_{rate_{n\times m}}$，则可将评分矩阵中每个用户的评分减去每个用户评分的均值，将缺失值用 0 补足后，记为S'_{rate}，让两个矩阵相乘，所得结果为预测评分矩阵：

$$S_{predict} = W S'_{rate} \tag{18}$$

如果待预测物品与用户买过的多个物品在购买上的相似度较大（也就是说该物品更经常与这些物品一并购买），则这样的效应会叠加。因此，该评分只用作排名，并不对应于用户在现有评分框架下对物品的评分，但是由于参照了用户对已购买的购买上类似的物品的评分，这里面的评分也能够代表用户对该物品的可能的满意程度。另外，这种特性能够让与用户购买过的多个物品之间都存在关联的物品的权重更高，从而更容易被推荐，因此该方法在某种程度上也做到了对关联物品的挖掘和推荐。

不过，对于浏览、搜索记录，可以使用物品间属性的相似度进行协同过滤，因为这种场景下，用户希望找到具有某种属性的物品，而此前搜索的关键词和浏览的

物品，往往体现了这种属性。

3.4 本章小结

本章简单介绍了群智化云服务 API 的推荐方法在用户行为、市场环境方面的基本假设，同时简单论述了群智化的 API 推荐策略。本章还列举了群智化 API 推荐方法需要准备的数据，论述了两个重要推荐模型——聚类与关联规则挖掘模型、评价推测模型的构建思路，以及可以应用的算法。

4 实验验证

4.1 引言

由于生态、功能、用户数量等方面的差异，不同的 API 市场使用的 API 推荐方法在细节上无法做到一致。本章基于已有的数据集，对前面提到的算法测试推荐效果，验证推荐方法的可行性和性能。

以下如涉及到小数，取小数点后五位。

相关代码请参见 GitHub 仓库：

https://github.com/DingJunyao/service_recommendation

4.2 实验环境

4.2.1 数据集

目前来说，由于针对于服务推荐的数据集少之又少，因此无法很好地完成实验。一个比较有名的数据集是 WS-DREAM，被广泛用于服务推荐算法的测试，但是 WS-DREAM 是 QoS 相关的数据集，用在以用户行为为主的群智化的服务推荐并不适合。不过我们可以使用其他的数据集进行类似的实验。

理论上说，在缺乏 API 购买、评价记录的情况下，数据集最好可以类似于耐用品的浏览、搜索、购买、评价记录，而且应当有用户的较为详细的信息。但是，撰写本文时，并没有找到这样的数据集，所以只能使用较为接近的 MovieLens 进行替代。

MovieLens 数据集首次发布于 1998 年，描述了人们对电影表达的偏好。数据集包括电影的信息和用户评价的信息，同时也分为多种版本供不同规模的研究使用[19]。

以 2016 年 10 月发布的 20m 版本的数据集为例，数据集包括以下的 csv 文件：

(1) movies.csv：记录电影的名称与分类；

(2) tags.csv：记录某个用户在某个时间为某个电影添加的标签；

(3) ratings.csv：记录某个用户在某个时间为某个电影进行的评分；

(4) links.csv：记录电影在 IMDb 和 TMDb 的链接序号；

(5) genome-tags.csv：罗列了一系列描述电影的标签（与用户添加的标签没

有关系），供 genome-scores.csv 使用；

(6) genome-scores.csv：描述了各个电影与各个标签的相关性（0-1），背后由用户进行评价[20]。

但是，不同规格的数据集，其含有的文件和意义也不一样。以 2003 年 2 月发布的 1m 版本的数据集为例，包含以下的 dat 文件：

(1) users.dat：用户信息，包括性别、年龄、职业；

(2) movies.dat：电影信息，包含信息与 20m 版本的相同；

(3) ratings.dat：评分信息，包含信息与 20m 版本的相同。

而且，这些 dat 文件以两个冒号“::”为各列的分隔，且没有标题行，与 csv 文件还是有不同的；20m 版本的用户 ID 是随机分配的，因此无法与 1m 版本的用户 ID 在真实环境下一一对应。鉴于实验考虑，本实验将对用户的数据集进行预处理，以 1m 版本的数据集为主，将 20m 版本的 genome-tags.csv 与 genome-scores.csv 整合进来，最终生成符合实验要求的数据集。预处理的方式在实验方法中描述。

4.2.2 运行环境

本实验使用 Python 语言编写，运行环境如下：

(1) CPU：AMD A10-7400P Radeon R6, 10 Compute Cores 4C+6G 2.50Ghz

(2) RAM：8.00GB（6.94GB 可用）

(3) 操作系统：Windows 10 专业版 1903

(4) CPython 解释器：Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

(5) 使用 Anaconda 集成环境，安装有 Cython，以便更高效地使用某些模块（如 gensim）

4.3 实验准备

4.3.1 数据集预处理

现有 MovieLens 数据集的 1m 版本和 20m 版本。首先加载所有数据集。由于 1m 版本和 20m 版本的电影数量不同，所以将 20m 数据集中 genome-scores 表中电影 ID 超过 1m 版本中 movies 表中的最大电影 ID 的电影删除。然后将电影的分类映射为矩阵，附加到 1m 版本的 movies 表中。最后统一使用 CSV 格式输出整合之后的数据集。后续实验均只使用此数据集。

数据集的各项文件的各列代表含义如下：

表 4.1　movies.csv 的各列代表含义

列	含义
MovieID	电影 ID
Title	电影名称（包含上映年份）
Action，Adventure，Animation，Children's，Comedy，Crime，Documentary，Drama，Fantasy，Film-Noir，Horror，Musical，Mystery，Romance，Sci-Fi，Thriller，War，Western	电影是否属于该类别，属于为 1，不属于为 0

表 4.2　users.csv 的各列代表含义

列	含义
UserID	用户 ID
Gender	性别，女为 F，男为 M
Age	年龄
Occupation	职业

表 4.3　ratings.csv 的各列代表含义

列	含义
UserID	用户 ID
MovieID	电影 ID
Rating	评分
Timestamp	时间戳（从 UTC 时间 1970-01-01 00:00:00 开始的秒数）

表 4.4　tags.csv 的各列代表含义

列	含义
TagID	标签 ID
Tag	标签名称

表 4.5　tag_scores.csv 的各列代表含义

列	含义
MovieID	电影 ID
TagID	标签 ID
Relevance	电影与标签的相关系数，在[0,1]区间

由于后来发现在测试某些算法时，计算机性能不足以使用这样的数据集完成实验，因此写程序代码输出缩减的实验数据，随机取用户和电影 ID 的一部分进行实验。这里取电影的 10%、用户的 20%进行实验。完整的数据集有 3883 部电影、6041 位用户和 1000209 个评分，缩减后的数据集有 388 部电影、1208 位用户和

20275 个评分。一般情况下使用缩减后的数据集进行实验，但是因为一些算法对数据量敏感，有时候也会使用完整的数据集进行实验。由于上面的过程是随机的，所以每次执行后，得到的结果都不一样。本实验只取一次结果的数据集进行测试。

4.3.2 矩阵准备

本实验中，需要生成物品（电影）相似度矩阵与用户相似度矩阵。而且，由于 MovieLens 中对行为仅有评分记录，所以对于用户行为，我们不生成浏览、搜索、购买之类的矩阵，而是仅生成评分矩阵与评分时间矩阵。

准备矩阵之前，为计算、比较方便，需要对数值进行标准化。本章中，标准化的方法是使用区间缩放法，将数值按以下方式映射到[0,1]区间中，最大值为 1，最小值为 0（如果最大值和最小值不同），若数值有感情倾向，数值越大越好：

$$y' = \begin{cases} \dfrac{y - y_{min}}{y_{max} - y_{min}}, & y_{max} - y_{min} \neq 0，且若y有感情倾向，y越大越好； \\ & 默认使用该式 \\ \dfrac{y_{max} - y}{y_{max} - y_{min}}, & y_{max} - y_{min} \neq 0，且若y有感情倾向，y越小越好 \\ 1, & y_{max} - y_{min} = 0 \end{cases} \tag{19}$$

其中y代表某组数据中的一项，y'是标准化后结果，y_{min}和y_{max}分别是这组数据的最小值和最大值。本实验对用户的年龄分组、职业分组采取了这种方法进行处理。对于性别，映射到 0、1。同时需要清除无关列——邮编，因为不同地区的邮编格式不一样，且有类似，在不提供更多信息的情况下无法比较。

之后计算电影间的相似度。在 API 推荐系统中，该操作等效为计算 API 之间的相似度。这里使用余弦相似性作为两两之间的相似度指标。首先读取并整合各分量。分量由电影类别和电影相对于各标签的相关系数组成，每个电影的量化属性组成 1146 维向量。经检查整合结果，发现一些电影没有测量过相对于标签的相关系数，这时候对应的值为空。为了测量这些电影与其他电影的相似度，需要对空缺值进行填充。这里的策略是：找到与这些电影类别相同的电影，填充这些电影对应相关系数的均值；如果没有其他类别相同的电影，则使用该电影类别值为 0 的类别值也为 0（即：同样不属于一些类别）的电影的对应相关系数的均值；如果还是没有，则填充全部电影的对应相关系数的均值。这样便可以较为精确地填充全部的空缺值。接下来计算余弦相似性，所得结果组成矩阵 s_similar，即为$S_{similar}$。Python 中，第三方模块 scikit-learn 提供了余弦相似性的计算方法，输入一个矩阵，就

能够非常快速地计算每一行两两之间的余弦相似性[21]。

为了进行训练与测试，需要对用户评分生成训练集和测试集，比例为 8:2，按照评分时间划分。对于每一个用户来说，较旧的 80%的评分划分到训练集，较新的 20%的评分划分到测试集。由此生成评分方面的矩阵，包括评分矩阵 s_rate（等效为S_{rate}）、评分时间矩阵 rate_time，以及新旧评分表、矩阵及评分时间矩阵 ratings_new、ratings_old、s_rate_new、s_rate_old、rate_time_new 和 rate_time_old。

接下来计算用户之间的相似度。这里使用用户的量化、标准化后的性别、年龄和职业作为向量的分量，计算两两之间的余弦相似性。等效为$U_{similar}$。

至此生成所有需要的矩阵。导出它们至 csv 文件。对于缩减后的矩阵，如果不包括数据集的导入导出，上面的准备用时 53.14550s；对于完整的矩阵，用时 696.28980s。接下来，如无特殊说明，无论是完整的数据集还是缩减后的数据集，使用评分数据的时候，为了测试准确度、召回度和 F-measure，使用的都是旧数据，也就是训练集。新数据（测试集）仅作为测试上面所述的指标用。

4.4 推荐算法

由于 MovieLens 中仅有用户对电影的评分，而没有购买、浏览、搜索调用的记录，因此可供参考的数据非常有限。这种情况下推荐系统的结构图如图 4.1 所示：

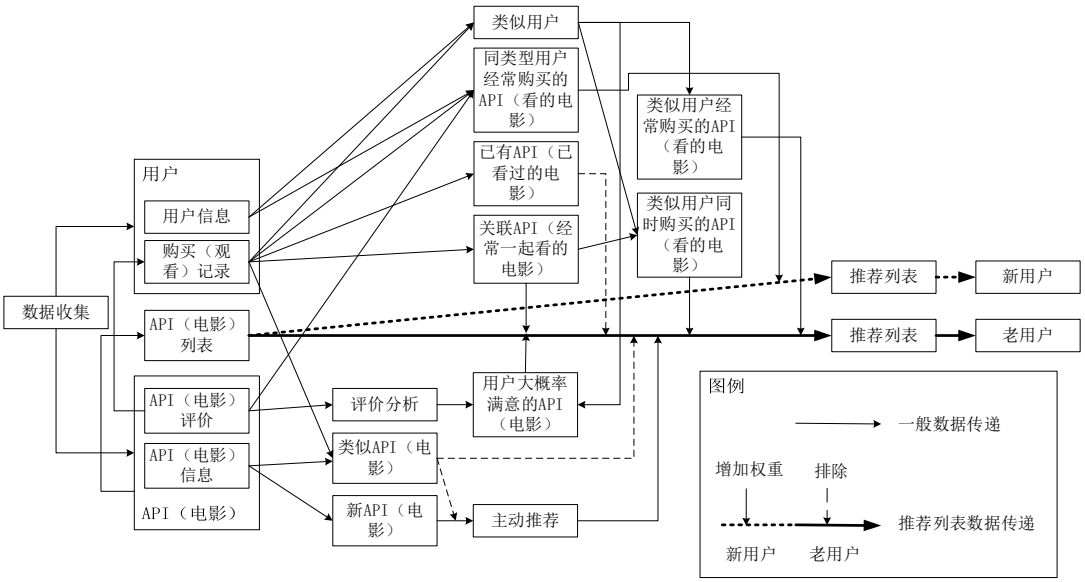

图 4.1　基于 MovieLens 数据集的推荐系统结构图

由于篇幅有限，本实验着重进行关联API（电影）和用户大概率满意的API（电影）的分析与推测。以下是本实验中用到的算法:

(1) Mean Shift 聚类算法；

(2) Apriori 算法；

(3) Word2Vec 算法；

(4) 基于物品的协同过滤；

(5) PersonalRank 算法；

(6) SVD 矩阵分解算法。

除了这些算法之外，还有两个对照组：

(1) 仅推荐热门内容；

(2) 随机推荐内容。

4.4.1 Mean Shift 聚类算法

对于 API 推荐，从用户的购买记录无法直接得知用户购买的次数，因此可以使用 Mean Shift 算法对购买记录内的物品（API）进行聚类分析，以聚集很可能同时购买的物品。该算法也是关联规则挖掘的基础。

在 Python 中，第三方模块 scikit-learn 提供了 Mean Shift 的模型，可以通过它对购买记录（这里以评分记录取代）快速进行聚类[21]。本实验调用它进行了聚类分析，最终生成的 cluster 列表为聚类结果，它的每一个子列表中将评价时间接近的电影 ID 聚合在一起，输出为 pickle 文件，可供其他算法使用。

该模型有一个参数——带宽（bandwidth），其大小决定了聚类的距离（也就是说多远距离内的元素能够被聚集在一起）。经测试，在带宽值为 25 的时候能够很好地聚集 30 天内的记录。

对缩减后的数据集进行聚类，用时 18.35300s，获得 1186 条同时购买的序列；对完整的数据集进行聚类，用时 987.93306s，获得 7573 条同时购买的序列。

4.4.2 Apriori 算法

Python 中的第三方模块 mlxtend 提供了较为集成的 Apriori 算法的使用方式[22]。可以使用 4.4.1 节的聚类后的列表进行关联规则挖掘。这里取最近的 20%的评分记录进行推荐。

但是经实验发现，Apriori 算法有以下的缺点：在用户和购买记录非常多的情

况下，挖掘关联规则时，需要将支持度调低，才能挖掘到更多置信度较高的规则；但是，这种情况下，会花费大量的时间。当最小支持度（指相对支持度，下同）设为 0.04 时，在缩减后的数据集上建立模型用时 0.21440s；当最小支持度设为 0.02 时，在缩减后的数据集上建立模型用时 20.08944s。而且，就算关联规则挖掘完成，为用户生成推荐列表时，所需时间也明显长于其他方法，而且关联规则越多，所需时间也越多。另外，关联规则的存储需要占用大量的内存。上述的情况随最小支持度降低越来越严重：在缩减后的数据集上进行实验时，在最小支持度设为 0.02 时，Python 进程占用内存在 600MB-750MB 之间波动；而最小支持度设为 0.01 后，在模型没有建立完毕的时候，Python 进程的占用内存就已经上升至 4GB 以上，致使程序无法继续执行。

即便能够运行，一些行为序列不包含关联规则内的任意项，所以返回空值，而且这种情况在 Apriori 算法中出现的概率比较大。作为推荐系统，不建议将其单独使用，而是与其他模型一起使用。

4.4.3 Word2Vec 算法

在 Python 中，第三方模块 gensim 提供了 Word2Vec 的模型，可以创建模型后使用聚类结果训练它[23]。在使用之前提到的聚类结果之前，需要将子列表中的每一项由整数类型转换为字符串类型，直接使用会报错。这里仍然使用 4.4.1 节中聚类后的列表进行训练。建议在安装该模块前安装 Cython 模块，以便充分发挥它的性能。

而且，该模型的一个优势是：模型可以导出，需要使用时可以导入，而且可以追加训练，适合于记录更新频繁的场合。

由于是根据向量而不是固定的规则来判断，Word2Vec 相比于 Apriori，生成的候选列表更多，返回空列表的概率也更低，而且运行速度也明显更快（尤其在安装 Cython 后）。但是，它受数据集的大小的影响非常严重，只适用于大数据集的情况下，在小数据集上表现不佳。同样，因为仍然有返回空推荐列表的情况，所以还是需要与其他模型一起使用。

使用该模型时，需要对参数进行调试。在完整的数据集上，对 min_count（进行训练的最低词汇频数）进行调参，在[0,100]区间上，分度值为 5 的情况下，模型的各项指标变化如图 4.2 所示（推荐列表项数为 10）：

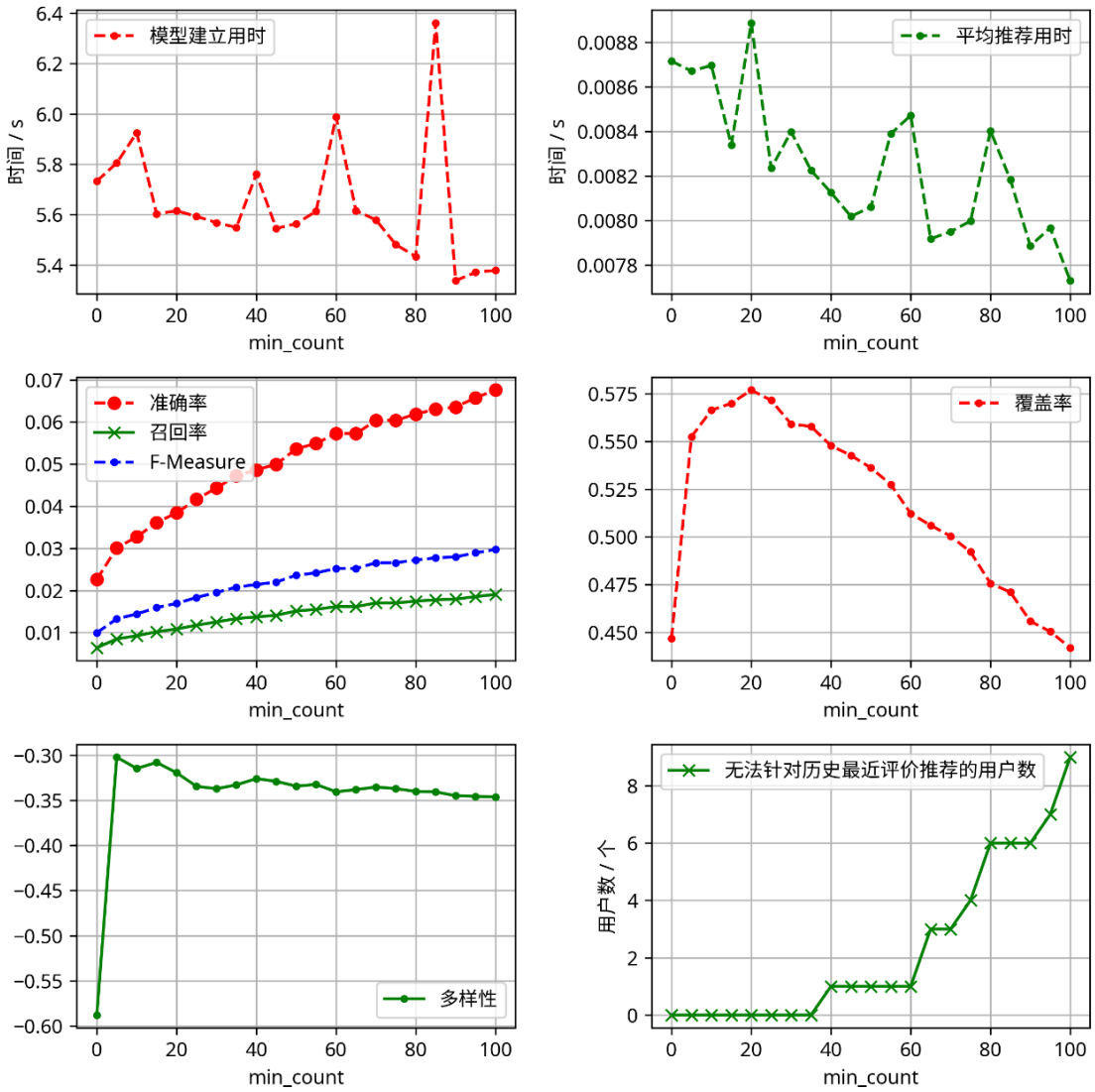

图 4.2　各指标与 min_count 值的关系

可以发现，当 min_count 变化时，各指标变化的趋势也有所不同：准确度相关的三个指标逐渐增加，而覆盖率在 min_count=20 时达到最高，随后下降；多样性在 min_count=5 时达到最高，随后基本上也呈下降趋势；而随着 min_count 的增加，一些用户可能无法获得推荐列表。在实际应用中，应该适当取舍，选择最合适的值。在完整数据集上，本实验选择 min_count=35 时的指标情况作为接下来的比较。

4.4.4　基于物品的协同过滤

在本实验中，使用 3.3.2 节提到的方法进行协同过滤，使用评价行为代替购买行为。经测试，该算法在完整的数据集上运行时，模型建立时间会非常长，因此只

适合离线推荐。但是，从缩减后的数据集上测试得到的结果看，它的准确度相关指标优于其他算法。

4.4.5 PersonalRank 算法

本实验在应用 PersonalRank 算法时，将评分矩阵中每个用户的评分减去对应用户的评分均值，仅使用平均化评分矩阵中大于等于 0 的评分代表的关联。这样做的目的是尽量消除个人因素对评价的影响，而且尽可能推荐评价较高的物品。

该算法的一个优势是覆盖率远高于其他推荐方式。另外，它是可扩展的，可以使用几乎相同的方法去判断类似用户、判断用户关联等等，为更精准的个性化推荐做准备。另外，它倾向于寻找与用户联系更近的物品，这一点让推荐的物品具有较高的关联度。它的一个明显的缺点是运行时间长，但一旦建立好模型后，能够迅速返回推荐结果。

4.4.6 SVD 矩阵分解算法

Python 的第三方模块 Numpy 中提供了简单的 SVD 矩阵分解的方法，分解速度相对来说也很快。在矩阵分解之前，需要填充矩阵中的缺失值，但经实验发现，如果评分区间为[1,5]，填充缺失值为 0，则矩阵分解效果不佳——缺失值通常会被接近于 0 的数值代替。因此，在执行该算法之前，首先将矩阵中每一位用户的评分减去这位用户的平均评分。这样一来，0 就成为了平均分，在输出结果的处理上也会变得方便。同时，这样做也能够尽量消除个人因素对评价的影响。

在应用 SVD 矩阵分解算法时，k的选择方式可以根据奇异值占比来决定：矩阵的能量值按照奇异值的平方和计算，而前k个奇异值的平方和与其之比就是前k个奇异值占比（能量占比）。如图 4.3 所示，一般来说，当$k \ll m$时，奇异值占比就能够达到很大，这使得 SVD 算法能够大大降低线上存储、计算的压力[15]。

理论上讲，奇异值占比越大越好，能够更好地还原矩阵；但是考虑到矩阵分解的目的是为了推测缺失值，而且要考虑性能问题，所以也不是越大越好。图 4.4 和图 4.5 展示了在缩减和完整的数据集上，各项指标与奇异值占比之间的关系：

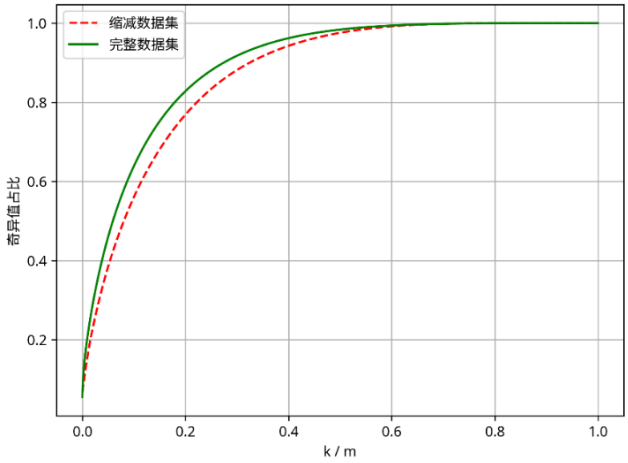

图 4.3　k值与奇异值占比的关系（图中横轴为k与m之比）

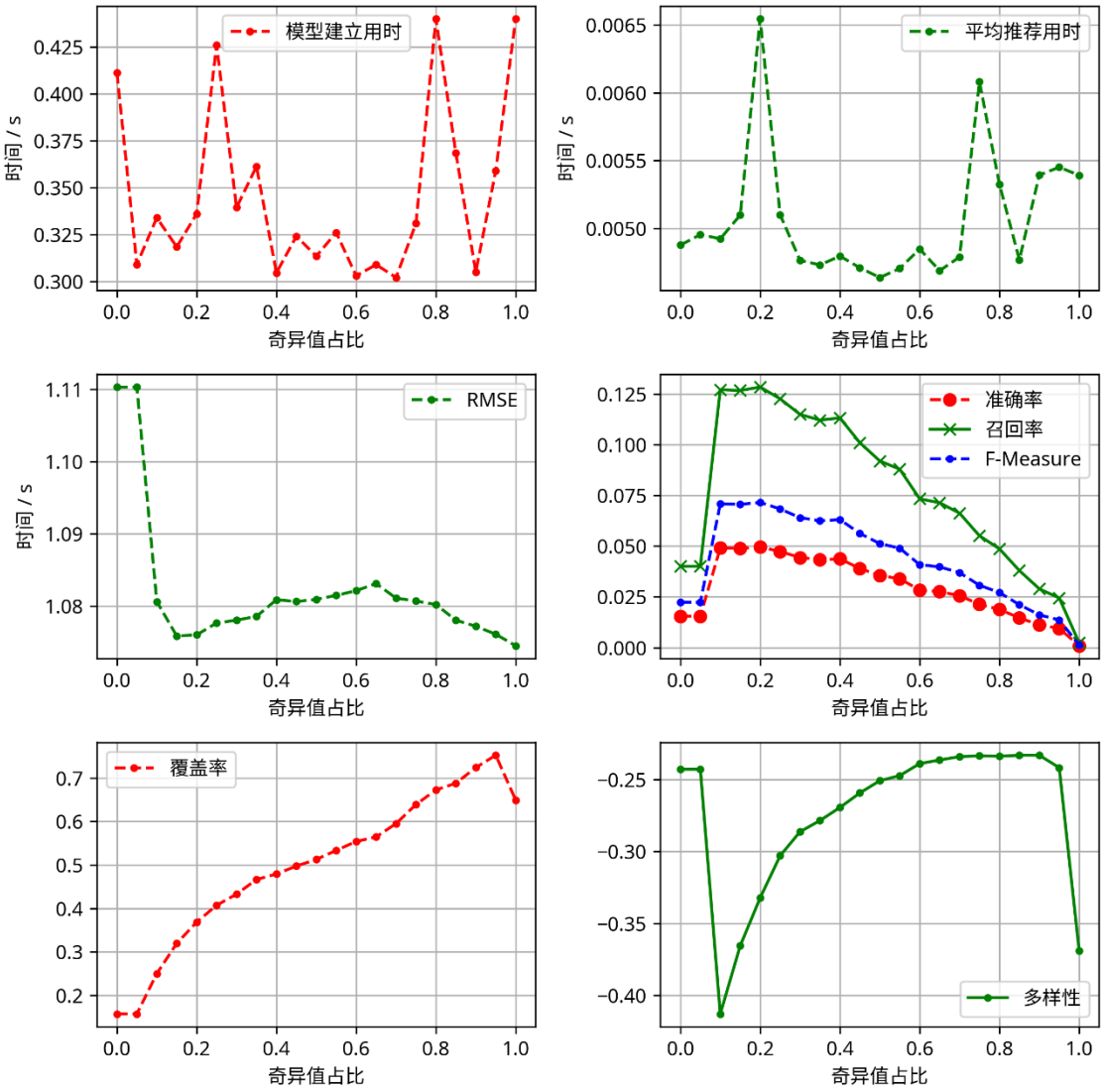

图 4.4　缩减数据集上各指标与奇异值占比的关系

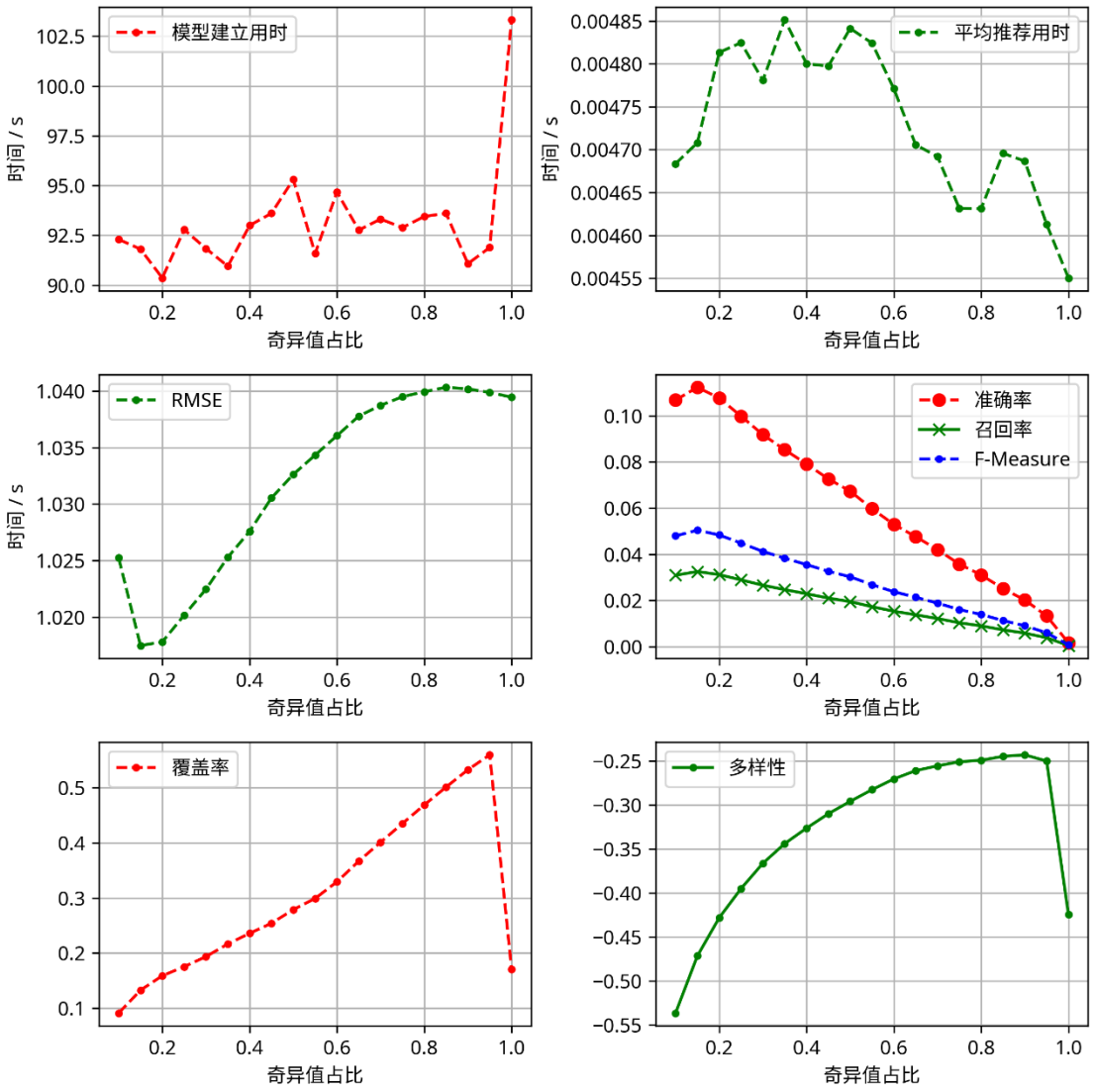

图 4.5　完整数据集上各指标与奇异值占比的关系

可以发现，无论是缩减的数据集还是完整的数据集，在奇异值占比为 0.15 左右的时候，准确性相关指标最优。但是，如果看覆盖率和多样性，奇异值占比在 0.9~0.95 的时候最优。考虑到电影与 API 在用户行为上具有一定的差异，本实验应该着重看覆盖率和多样性。

4.5　实验结果

规定推荐列表内的项目数量为 10，在缩减和完整的数据集上的实验数据如下：

表 4.6　各算法在缩减的数据集上的各项指标

指标	仅推荐热门内容（对照）	随机推荐（对照）	Apriori	Word2Vec	Item-CF-buy	Personal-Rank	SVD
建立模型用时/s	0.02089	0	22.09340	0.18400	214.27518	19.90273	0.34997
为单个用户返回推荐列表的平均用时/s	无	0.00057	0.57042	0.00455	0.00285	0.00174	0.00457
RMSE	无	无	无	无	无	无	1.08018
准确率	0.05157	0.01010	0.04396	0.00308	0.04678	0.02467	0.01879
召回率	0.13355	0.02615	0.08296	0.00643	0.08060	0.06388	0.04866
F-Measure	0.07441	0.01457	0.05747	0.00416	0.05920	0.03559	0.02711
覆盖率	0.02577	1	0.25515	0.02577	0.75258	0.87113	0.67268
多样性	-0.48005	-0.27699	-0.28284	-0.21073	-0.37091	-0.31439	-0.23375

表 4.7　各算法在完整的数据集上的各项指标

指标	仅推荐热门内容（对照）	随机推荐（对照）	Apriori	Word2Vec	PersonalRank	SVD
建立模型用时/s	0.95100	0.00100	111.27036	5.54903	1203.65566	92.51479
为单个用户返回推荐列表的平均用时/s	无	0.00680	0.76243	0.00823	0.00247	0.00484
RMSE	无	无	无	无	无	1.03993
准确率	0.04610	0.00846	0.06037	0.04720	0.04298	0.03104
召回率	0.01335	0.00245	0.01400	0.01333	0.01245	0.00900
F-Measure	0.02071	0.00380	0.02273	0.02079	0.01930	0.01394
覆盖率	0.00258	1	0.03811	0.55782	0.81701	0.46871
多样性	-0.51735	-0.28120	-0.41265	-0.33283	-0.36140	-0.24908

其中：

(1) 建立模型时间不包括 4.3.2 节中提到的矩阵的生成时间与数据的载入时间。在建立模型前，所有所需数据已经被载入到内存中。

(2) Apriori 算法中，在缩减后的数据集上的最小支持度为 0.02，在完整的数据集上的最小支持度为 0.09；关联规则的最小置信度为 0.2。限于硬件条件和模型建立耗时太长，无法进行更低的最小支持度的情况下的实验。

(3) Word2Vec 在缩减后的数据集上使用默认参数；在完整的数据集上设定 min_count 为 35。更多参数下的情况可以参阅图 4.2。

(4) SVD 算法中，取奇异值占比为 80%的奇异值。更多参数下的情况可以参阅图 4.4 和图 4.5。

(5) Item-CF-buy 是指将物品之间在购买上的相似度作为传统的协同过滤的“物品相似度”来处理的协同过滤，其方法在 3.3.2 节中有详细介绍。由于在实验环境下因模型建立时间太长无法测试，本实验曾尝试在阿里云的 ECS（弹性云主机）上在完整的数据集上运行该算法，但在建立模型的时候发生内存错误，而且此时建立模型用时已经超过 24h。该 ECS 的运行环境如下：

1) CPU：Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz
2) RAM：2.00GB
3) 操作系统：CentOS Linux release 7.6.1810 (Core)（Linux 3.10.0-957.1.3.el7.x86_64）
4) CPython 解释器：Python 3.6.6 (default, Mar 29 2019, 00:03:27) [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux

可以看出，PersonalRank、Word2Vec 在完整的数据集上的覆盖率较好。PersonalRank 能够发掘更多的长尾内容，SVD 能够提供更加多样性的结果，而 Apriori 算法只能推荐极为有限的数据。如果看缩减后的数据集上的结果，基于物品的协同过滤算法和 Apriori 算法在准确性上的表现更出色，然后是 PersonalRank。Word2Vec 算法在完整数据集上的准确性仅次于 Apriori 算法，但它的训练需要大量的数据做支撑，在缩减后的数据集上表现较差。就运行速度而言，Wod2Vec 和 SVD 在模型建立用时上是最少的，基于物品的协同过滤和 Apriori 则耗费大量的时间。而且，Apriori 在模型建立完成后，为用户进行推荐时依然会耗费大量的时间，并不适合服务推荐。

同时，推荐列表的项数也对推荐系统的各项指标有影响：在保证推荐效果的情况下，越多的项数意味着越大的命中可能性，但如果项数太多，就失去了推荐系统的意义了。

另外一方面，如果仅推荐热门内容，不管是缩减后的数据集还是完整的数据集，准确性指标都比较高。这和电影的特性有关：热门的电影会有更多的人扎堆观看、

评价。同时，因为MovieLens是电影的数据集，电影和API还是有不同的：电影之间不存在短期内关联消费的情况，除非是一个系列的电影；用户观看电影很大程度上不是因为需要，而是因为喜好。因此，在这样的数据集上判定的准确性相关指标的意义不大。但使用 MovieLens 是不得已而为之，如果有更类似于 API 市场环境下的数据集，那么应该使用这样的数据集进行实验。

4.6 本章小结

本章在 MovieLens 数据集上对群智化 API 推荐中可用的一些常用算法的推荐效果进行了实验，并分析了实验结果。这些算法包括：Mean Shift、Apriori、Word2Vec、基于物品的协同过滤、PersonalRank、SVD。其中，Mean Shift 用来聚类评分记录(类比API市场中的浏览、搜索、购买、评价记录)，供Apriori、Word2Vec、基于物品的协同过滤这些算法使用。

5 推荐策略建议与展望

5.1 推荐策略建议

虽然在 API 推荐中可以使用的推荐方法很多，但是各种推荐方法各有利弊，而且没有一种方法能够充分利用数据的全部信息，每种方法都有其局限性[15]，因此推荐时一般不会使用一种方法进行推荐，而是需要进行混合推荐，即综合运用多种推荐方法，取长补短，以实现推荐可用性的最大化。根据上一章的实验结果，综合考虑，可以综合使用基于物品的协同过滤、PersonalRank、SVD 和 Word2Vec 进行推荐。

Word2Vec、PersonalRank 比较重关联，可以作为推荐列表的初步过滤或加权；SVD 重评分，可以将得到的初步推荐列表中的 API 进行用户评价推测，以此进行推荐。而基于物品的协同过滤则兼顾两方面，推荐效果也不错，但是模型建立时间太长，所以可以同时使用两套推荐系统，交叉混合推荐。这样可以兼顾推荐的多样性和准确性。

基于物品的协同过滤、PersonalRank、SVD 适合离线推荐，建议模型更新频率由低到高依次为基于物品的协同过滤、PersonalRank 和 SVD；Word2Vec 适合在线推荐，能够根据新的用户行为迅速训练并应用。

对于购买、评价记录不足的 API 的冷启动，可以按某个比例（如 1~5%）对它们随机推荐，均匀地放到推荐列表中，以增加曝光度。最好根据 API 在功能上的相似度，推荐功能上与推荐列表内 API 类似的 API。对于用户的冷启动问题，可以根据用户注册时或回访时提供的信息，为用户推荐同类型用户购买的服务。

不同推荐算法得到的推荐结果在排除已购买的物品和与已购买的物品高度相似的物品后，按一定比例混合在给用户的最终推荐列表中，要保证用户在若干次翻页后不会遇到重复的物品，而且尽量分布均匀，不会让用户明显感觉到 API 越来越不相关、越来越差。

图 5.1 展示了这样的推荐系统的推荐方法示意图。其中，虚线箭头为排除项目，虚线包裹的方框表示实验中没有涉及到的数据，但在实际的 API 推荐系统中，如果有这些数据，尽量将这些数据也应用于对应的算法。图中没有包含用户的冷启动问题的解决方案：

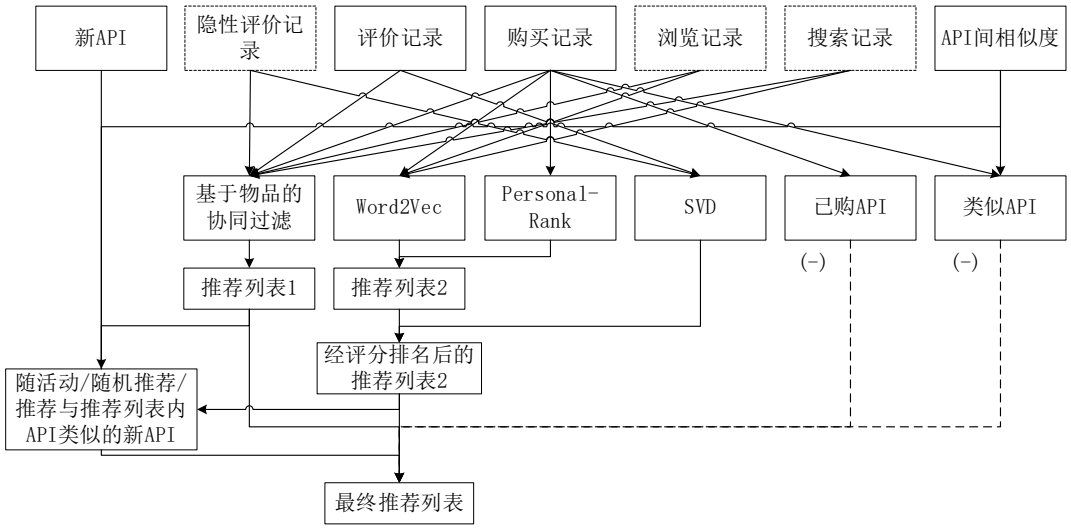

图 5.1　　推荐方法示意图

由于目前的数据集对 API 的推荐效果无法很好地进行判断，就算上面进行了实验，也无法很好地选出最合适的 API 推荐方法。为了鉴别推荐方式的优劣，可以在上线应用时对推荐列表中隐性标注推荐来源，如果用户点击、收藏、购买了服务，则说明该推荐方式有效，在后台为该推荐系统计数，分析推荐系统的准确性和收益与成本之比。这样，也可以通过群体智慧，选出最合适的推荐系统，做到推荐的群智化。

5.2　不足与展望

由于种种原因，本文有以下的遗憾之处：

(1) 由于计算机硬件问题，为完成实验，不得不将数据集缩减。对于推荐系统的实验，如果条件允许，最好选用比较大的数据集。同时，也无法对 5.1 节中提出的推荐策略进行实际的调试与测试，在实际应用中需要具体问题具体分析，对各算法的参数和推荐项数、更新频率等进行调整，以获得最佳的结果。

(2) 由于自己的数学和计算机水平有限，不擅长算法和代码的优化，加上工具简陋，一些算法的实验结果与文献内的结果差别较大。

(3) 准备不充分，不知道有第三方库提供了更为快速简便的功能。比如计算余弦相似性，最开始是自己写算法，一个值一个值进行计算，仅仅是计算完整数据集中项目之间的余弦相似性时，就用了大约 10h。而使用 scikit-

learn 提供的方法，完成整个矩阵准备过程仅用时 696.28980s。因此，在知道这样的方法之前，浪费了大量的时间。

(4) 由于缺乏云服务 API 推荐的相关数据集，本文使用了 MovieLens 数据集进行实验。但是 MovieLens 仅有用户对电影的评分，而无购买、浏览、搜索记录，更没有类似于 API 的调用记录的数据，加上电影和 API 的性质不同（如：人们会去观看与自己喜欢的电影类型高度类似的电影，而不会去购买与已购买 API 高度类似的 API），因此各算法计算出的推荐效果不是很理想。比如：为了做准确度相关的测试，本文的策略是分离出每个用户最近的 20%的评价记录，使用剩余的 80%进行模型训练，再用训练后的模型进行推荐，判断与用户接下来的评分/选择的相似程度。但是电影的选择和商品的选择不一样——除非是一个系列的电影，否则电影之间不存在类似于“关联购买”这样的情况。如果有 API 市场公开这方面的数据集的话，将大大减轻推荐系统开发人员的负担。但是，由于可能涉及到商业机密，这个想法很难实现。

(5) 在写论文的时候希望能够通过 MovieLens 提供的 IMDb 链接序号，爬取对应电影的信息，根据描述信息判断对应电影之间的相似度，但由于计算机性能和时间有限，未能进行。如果有人进行云服务 API 推荐，可以尝试类似的方法。但是，比较现实的情况是，很多商家在描述上使用图片进行表达，这就涉及到 OCR 处理等等的内容了。

实际上，API 并不是各大云计算提供商的云市场内销售的主要服务，云市场内的服务主要都是一整套解决方案（如镜像、建站服务、运维管理服务等。但是，由于它们都是服务，而且和 API 的性质差不多——用户希望能够推荐他们需要的服务，因此本文论述的推荐策略同样适用于它们。

结　论

长期以来，对于云服务 API 的推荐，众多研究都集中于 QoS 的预测、内容的处理和用户的偏好和信任度上。但是，这些推荐方法大多数为学术性研究，条件较为苛刻，而且成本较高，不适合刚刚发展的 API 市场。为此，本文希望将推荐系统中已广泛应用的群智化思想引入推荐方法中，提出一种过渡性的、较为节约成本的、适用于当前 API 市场规模的群智化 API 推荐方法。

本文分析了 API 市场的现状，同时指出了云服务 API 的推荐相比于其他的推荐，有什么不同之处，需要注意什么：用户选购 API 具有较强的目的性，因此需要将推荐的重点放在“用户需要什么”，给用户推荐他们可能需要的 API；一般而言，用户不会去购买与已经购买且配额充足的 API 高度类似的 API，在推荐的时候需要避免推荐这样的 API；由于 API 推荐使用频率较低，服务增长速度慢，对于 API 的推荐，可以将更多时间放在离线推荐上，或者是使用较为简单的方式进行推荐，节约成本；API 市场的用户数量远远超过物品（API）数量，潜在用户非常广泛，但同时用户行为数据高度稀疏。

基于上面的论断，本文论述了云服务 API 推荐的推荐策略：推荐配套（关联）服务、推荐类似用户购买的服务、推荐用户现在感兴趣的服务。与此同时，要保证服务的高质量，可以通过 QoS 值和用户评价（包括显性的和隐性的）进行判定，推荐用户大概率满意的服务。据此，本文描述了两种模型：聚类与关联规则挖掘模型，以及评价推测模型。

实现这些模型的算法很多。如果推荐关联服务，首先需要对用户的购买记录进行聚类，推测出一并购买的服务，从而对它们进行关联规则挖掘。聚类算法中，Mean Shift 算法能够不事先规定簇的个数就能较好地实现聚类。关联规则挖掘，除了 Apriori 算法外，还可以通过 Word2Vec、PersonalRank、改进后的基于物品的协同过滤算法等进行近似替代。实现评价推测，可以使用基于物品的协同过滤和矩阵分解等算法。

接下来，本文在 MovieLens 数据集上对这些算法进行了实验。实验发现，Apriori 算法推荐的数据极其有限；Word2Vec 和 SVD 在模型建立时间用时极少；在覆盖率上，PersonalRank 和 Word2Vec 表现出色，而且 PersonalRank 能够发掘更多的长尾内容；在准确性上，Apriori、Word2Vec、基于物品的协同过滤较好。

不同的算法各有千秋，因此可以通过一定的搭配，加上对评价不足的 API 进行随机推荐、对新用户推荐同类型用户购买的服务等方式，混合推荐。这样可以兼顾推荐的多样性和准确性。

同时，因为现有数据集无法很好地评价涉及到关联物品的推荐准确性，本文也提出了群智化的推荐列表效果评价机制，让用户的选择对推荐系统的优劣进行隐性评价，作为改进推荐系统的依据。

致　谢

如果要说最应该致谢的人的话，大概就是我自己了。我能够在几个月内从对推荐系统一无所知，到独立完成全部的任务，算是一个奇迹了吧。也不枉我这几个月一直泡在图书馆里面，借了将近 20 本书了吧。在这期间，我经历了诸多身体不适的情况，但都尽力克服了，在此我还是非常敬佩自己的意志的。

感谢 Python、Numpy、pandas、scikit-learn、Matplotlib、mlxtend、gensim、Visual Studio Code 等等的开源项目及其背后数不清的研发人员与代码贡献者对本文做出的间接贡献。如果没有这些人的无私努力，我的论文也无法完成。

感谢王磊老师让我知道了“群智化”这个词，以及它的明确的含义。否则，这三个字可能永远也不会被我认识到。同时，正因为他，才让我对推荐系统（尤其是服务推荐）有了了解的机会。在论文的修正的工作中，他给予了很大的帮助，让我的论文达到完美。

感谢阿里云为学生提供了便宜的入门级云服务器的租用渠道，在实验前期，服务器帮助我完成了一些矩阵的生成。那时候代码没有优化，需要运行 10 个小时才能生成相似度矩阵，阿里云的云服务器在这个阶段发挥了重要的作用。虽然后来我发现了更快速的方法，但是还是感谢有这样的资源。也感谢阿里云的天池大数据众智平台让我对大数据处理过程进行了初步的了解，为本文的实验做准备。

同时，我在这几个月中，除了查阅大量的文献、书籍外，还查阅了大量的网络资料，这些资料很大程度上帮助我了解了上面所述的那些开源项目的使用方法，从而让我能够出色地完成论文。我想，依靠无数人写成的知识完成任务，这就是群智化吧——人类文明和知识的发展，本来就是群智化的吧。

最后，感谢学校在毕业设计上为我们提供了诸多帮助，让我们能够放心地完成任务。

本篇论文，也是我为这四年画上的句号吧。

参考文献

[1] Doan A, Ramakrishnan R, Halevy A Y. Crowdsourcing systems on the world-wide web[J]. Communications of the ACM, 2011, 54(4): 86-96.

[2] 张秀伟，何克清，王健，刘建晓. Web 服务个性化推荐研究综述[J]. 计算机工程与科学，2013，35(09)：132-140.

[3] 唐明董，张婷婷，杨亚涛，郑子彬，曹步清. 基于因子分解机的质量感知 Web 服务推荐方法[J]. 计算机学报，2018，41(06)：1080-1093.

[4] 陆贝妮，杜育根. 基于社区发现的 Web 服务 QoS 预测[J]. 计算机工程，2019，45(03)：117-124.

[5] 汪浩，肖建茂，龙浩，汪乐约. 基于 Bootstrap 技术的 Web 服务 QoS 值的置信区间估计和预测[J]. 电子学报，2018，46(03)：665-671.

[6] Yi-wen Zhang, Yuan-yuan Zhou, Fu-tian Wang, Zheng Sun, Qiang He. Service recommendation based on quotient space granularity analysis and covering algorithm on Spark[J]. Knowledge-Based Systems, 2018, 147.

[7] 李鸿超，刘建勋，曹步清，石敏. 融合多维信息的主题自适应 Web API 推荐方法[J]. 软件学报，2018，29(11)：3374-3387.

[8] Shi M, Tang Y, Liu J. Functional and Contextual Attention-based LSTM for Service Recommendation in Mashup Creation[J]. IEEE Transactions on Parallel and Distributed Systems, 2018.

[9] 曹步清，刘建勋，唐明董，谢芬方. 基于用户使用历史与信誉评价的 Web API 推荐[J]. 计算机工程，2015，41(06)：43-48+55.

[10] 李凌飞. 基于服务网络和服务之间关系的云服务个性化推荐[D]. 北京邮电大学，2015.

[11] 黄琳. 基于 QoS 度量的 Web 服务选择与推荐方法研究[D]. 北京邮电大学，2018.

[12] 王泽源. 考虑用户满意度的云服务排序推荐方法研究[D]. 合肥工业大学，2018.

[13] 熊丽荣，王玲燕，黄玉柱. 一种联合 LTR 和社交网络的 Top-k 推荐方法[J]. 小型微型计算机系统，2018,39(12)：2577-2584.

[14] 王勇. 基于 ProgrammableWeb 网站的 Web 服务推荐研究[D]. 南京大学，2018.

[15] 黄昕,等. 推荐系统与深度学习[M]. 北京：清华大学出版社，2019：48-93.

[16] Grbovic M, Radosavljevic V, Djuric N, et al. E-commerce in your inbox: Product recommendations at scale[C]//Proceedings of the 21th ACM SIGKDD International

Conference on Knowledge Discovery and Data Mining. ACM, 2015: 1809-1818.

[17] Grbovic M, Djuric N, Radosavljevic V, et al. Scalable semantic matching of queries to ads in sponsored search advertising[C]//Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016: 375-384.

[18] 赵志勇. Python 机器学习算法[M]. 北京: 电子工业出版社, 2017: 206-291.

[19] Harper F M, Konstan J A. The movielens datasets: History and context[J]. Acm transactions on interactive intelligent systems (tiis), 2016, 5(4): 19.

[20] Vig J, Sen S, Riedl J. The tag genome: Encoding community knowledge to support novel interaction[J]. ACM Transactions on Interactive Intelligent Systems (TiiS), 2012, 2(3): 13.

[21] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python[J]. Journal of machine learning research, 2011, 12(Oct): 2825-2830.

[22] Raschka S. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack[J]. The Journal of Open Source Software, 2018, 3(24).

[23] Rehurek R, Sojka P. Software framework for topic modelling with large corpora[C]//In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. 2010.