

分类号 TP957

学号 0123456

UDC

密级 公开

工学博士学位论文

国防科大学位论文 L^AT_EX 模板 使用手册

博士生姓名 张三

学科专业 通信与信息工程

研究方向 自动目标识别与模糊工程

指导教师 李四 教授

王五 副教授

国防科学技术大学研究生院

二〇一六年十一月

How to Use the L^AT_EX Document Class for NUDT Dissertations

Candidate: Zhang San

Supervisor: Professor Li Si

A dissertation

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Engineering

in Information and Communication Engineering

Graduate School of National University of Defense Technology

Changsha, Hunan, P. R. China

November 29, 2016

独 创 性 声 明

本人声明所呈交的学位论文是我本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写过的研究成果，也不包含为获得国防科学技术大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文题目：_____国防科学技术大学学位论文 L^AT_EX 模板_____

学位论文作者签名：_____日期：_____年____月____日

学位论文版权使用授权书

本人完全了解国防科学技术大学有关保留、使用学位论文的规定。本人授权国防科学技术大学可以保留并向国家有关部门或机构送交论文的复印件和电子文档，允许论文被查阅和借阅；可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密学位论文在解密后适用本授权书。）

学位论文题目：_____国防科学技术大学学位论文 L^AT_EX 模板_____

学位论文作者签名：_____日期：_____年____月____日

作者指导教师签名：_____日期：_____年____月____日

目 录

摘 要	i
ABSTRACT	iii
第一章 绪论	1
1.1 研究背景	1
1.2 相关研究工作	1
1.3 本文的工作与创新	1
1.4 论文结构	1
第二章 基于语义扩展和文本质量的实时个性化搜索	3
2.1 研究动机	4
2.2 相关定义	6
2.3 方法描述	10
2.3.1 系统框架	10
2.3.2 特征提取模块	12
2.3.3 特征提取模块	14
2.3.4 候选生成模块	14
2.3.5 排序推送模块	18
2.4 实验分析	20
2.4.1 数据集及实验环境	20
2.5 本章小结	20
第三章 基于卷积神经网络的文本分类研究	21
3.1 研究动机	21
3.2 相关定义	21
3.3 方法描述	21
3.4 实验分析	21
3.5 本章小结	21
第四章 社交网络中传播效率最大化	23
4.1 研究动机	24
4.2 相关定义	26
4.2.1 传播模型以及影响力最大化问题	26
4.2.2 传播效率最大化问题	31

4.3	方法描述	32
4.4	实验分析	43
4.4.1	实验设置	43
4.4.2	相似度对比	44
4.4.3	算法精确度对比	45
4.4.4	运行时间对比	47
4.4.5	可扩展性	48
4.5	本章小结	48
第五章	基于概率阅读模型的影响力量化研究	51
5.1	研究动机	51
5.2	相关定义	51
5.3	方法描述	51
5.4	实验分析	51
5.5	本章小结	51
第六章	总结与工作	53
6.1	本文工作总结	53
6.2	未来工作展望	53
致谢	55
参考文献	57
作者在学期间取得的学术成果	61
附录 A	模板提供的希腊字母命令列表	63

表 目 录

表 2.1	推文 JSON 数据的部分字段信息	8
表 2.2	用户 JSON 数据的部分字段信息	9
表 2.3	用户查询 JSON 数据的部分字段信息	9
表 4.1	常用符号列表	26
表 4.2	数据集特征	43

图 目 录

图 2.1	社交网络平台中信息的实时个性化搜索示意图	5
图 2.2	社交网络平台中信息的实时个性化搜索任务图	7
图 2.3	实时个性化搜索系统框架图	11
图 2.4	用户查询的语义扩展示意图	13
图 2.5	<i>WordCnt</i> 的累积分布函数	17
图 2.6	<i>CharCnt</i> 的累积分布函数	17
图 2.7	<i>FollowerCnt</i> 的累积分布函数	17
图 2.8	<i>FolloweeCnt</i> 的累积分布函数	17
图 2.9	<i>FFR</i> 的累积分布函数	17
图 2.10	<i>StatusCnt</i> 的累积分布函数	17
图 2.11	推特数据流的分布	20
图 4.1	信息传播以及影响力最大化示意图	24
图 4.2	社交网络中的信息传播概率图 \mathcal{G}	27
图 4.3	随机实例图 g	27
图 4.4	$R_X(u)$, $R_X(S)$ 以及 $R_X(W)$ 的示意图	38
图 4.5	在 <i>Facebook</i> 数据集上不同 k 下的 Jaccard 相似度对比	45
图 4.6	在 <i>HepPh</i> 数据集上不同 k 下的 Jaccard 相似度对比	45
图 4.7	在 <i>Twitter</i> 数据集上不同 k 下的 Jaccard 相似度对比	45
图 4.8	在 <i>DBLP</i> 数据集上不同 k 下的 Jaccard 相似度对比	45
图 4.9	在 <i>Facebook</i> 数据集上不同 k 下的传播效率对比	46
图 4.10	在 <i>HepPh</i> 数据集上不同 k 下的传播效率对比	46
图 4.11	在 <i>Twitter</i> 数据集上不同 k 下的传播效率对比	46
图 4.12	在 <i>DBLP</i> 数据集上不同 k 下的传播效率对比	46
图 4.13	在 <i>Facebook</i> 数据集上不同 k 下的运行时间对比	47
图 4.14	在 <i>HepPh</i> 数据集上不同 k 下的运行时间对比	47
图 4.15	在 <i>Twitter</i> 数据集上不同 k 下的运行时间对比	47
图 4.16	在 <i>DBLP</i> 数据集上不同 k 下的运行时间对比	47
图 4.17	反向效率采样算法的可扩展性	48

摘 要

国防科学技术大学是一所直属中央军委的综合性大学。1984 年,学校经国务院、中央军委和教育部批准首批成立研究生院,肩负着为全军培养高级科学和工程技术人才与指挥人才,培训高级领导干部,从事先进武器装备和国防关键技术研究的重要任务。国防科技大学是全国重点大学,也是全国首批进入国家“211 工程”建设并获中央专项经费支持的全国重点院校之一。学校前身是 1953 年创建于哈尔滨的中国人民解放军军事工程学院,简称“哈军工”。

关键词: 国防科学技术大学; 211; 哈军工

ABSTRACT

National University of Defense Technology is a comprehensive national key university based in Changsha, Hunan Province, China. It is under the dual supervision of the Ministry of National Defense and the Ministry of Education, designated for Project 211 and Project 985, the two national plans for facilitating the development of Chinese higher education.

NUDT was originally founded in 1953 as the Military Academy of Engineering in Harbin of Heilongjiang Province. In 1970 the Academy of Engineering moved southwards to Changsha and was renamed Changsha Institute of Technology. The Institute changed its name to National University of Defense Technology in 1978.

Key Words: NUDT; MND; ME

符号使用说明

HPC	高性能计算 (High Performance Computing)
cluster	集群
Itanium	安腾
SMP	对称多处理
API	应用程序编程接口
PI	聚酰亚胺
MPI	聚酰亚胺模型化合物, N-苯基邻苯酰亚胺
PBI	聚苯并咪唑
MPBI	聚苯并咪唑模型化合物, N-苯基苯并咪唑
PY	聚吡咙
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡咙薄膜
ΔG	活化自由能 (Activation Free Energy)
χ	传输系数 (Transmission Coefficient)
E	能量
m	质量
c	光速
P	概率
T	时间
v	速度

第一章 绪论

1.1 研究背景

1.2 相关研究工作

1.3 本文的工作与创新

1.4 论文结构

第二章 基于语义扩展和文本质量的实时个性化搜索

随着社交网络中信息爆炸式的增长,用户越来越难以从海量的信息中获取到自身所需的信息,用户所需的信息往往淹没在其中,这种现象可称之为**信息洪流**(*information flood*)现象。在社交网络、电子商务网站、即时通信等应用中,信息产生的速率以及数量都是过去无法比拟的,如在推特(Twitter)、脸书(Facebook)、新浪微博等社交媒体中,每天都会产生海量的文本信息。根据用户输入的查询,在海量的信息流中实时地检索出高质量的、相关的信息是一个极具挑战性的问题。社交网络中的信息流实时个性化搜索相对于传统的信息检索提出了以下挑战:(1)社交网络中充斥着各种各样话题的信息,而且信息大多数都是以短文本表示,相对于传统的新闻等长文本信息,难以进行语义理解;(2)社交网络中的信息质量参差不齐,难以从中遴选出高质量的信息;(3)社交网络中的信息产生速率快,如何能够实时地检索出用户所需要的信息,将其推送给用户也是一大难点。因此,由于社交网络中数据的信息海量性、主题多样性、数据稀疏性以及社交互动性等特性,传统的个性化检索方法不足以解决社交网络中的信息实时个性化搜索问题。本章针对以上的问题,提出了一个面向推特信息流的实时个性化搜索框架来实现用户的信息实时推荐。本章针对社交网络中信息的特性,提出了一种基于语义扩展和文本质量的实时个性化搜索算法,并在 TREC 2015 Microblog Track^[1] 测评中验证了算法的性能。首先,我们构造了一个逻辑规则过滤器来选择核心关键词,提高检索的准确率。其次,我们对文本质量进行建模,利用的标注的数据进行训练,以此来对文本的质量的进行打分。训练好的文本质量模型提高了检索的排序性能。然后,我们使用外部语料库来实现语义扩展,例如搜索引擎,知识库等。语义扩展能够使得我们更好地理解用户的偏好和兴趣。最后,我们采用了一个动态的推送策略来自动地推送高质量且相关的信息给特定的用户,这能够避免信息过载。本章中的算法结合了社交网络文本的语义特征和社交属性,针对不同的用户搜索,做了综合性地排序。我们使用 TREC 2015 Microblog Track 中的真实数据流进行了实验,实验结果显示了本章提出的算法在不同测评指标下,与其他算法相比的优越性。

本章的内容组织如下:第2.1节介绍了研究动机,讨论了在社交网络环境下进行实时个性化搜索的必要性。第2.2节介绍了相关定义,对本章中涉及的相关概念 and 知识进行了符号化的定义。第2.3节介绍了方法描述,详细地阐述了本章提出的系统框架和算法。第2.4节进行了实验分析,验证了本章提出的方法,并且分析了实验结果。最后,第2.5对本章的内容进行了总结。

2.1 研究动机

随着大数据时代的到来，诸如推特 (Twitter)、脸书 (Facebook)、新浪微博等的社交网络平台逐渐取代传统的媒体平台，成为新时代的实时信息交互平台。以推特平台为例，据统计，每天平均约有 58,000,000 条推文 (tweet) 发布，每天平均处理约 2,100,000,000 次搜索查询，每月约有 115,000,000 个活跃的用户¹。在大数据时代，人人都可以是信息的生产者、传播者和接收者，这在一定程度上加速了信息的传播速率。然而，如此庞大的数据量使得用户在社交网络平台上搜索查询时面临了信息过载的问题，用户很难检索到自己需要的信息，亦或用户所需的信息淹没在了众多的信息之中。尤其是社交网络平台中的信息内容囊括了众多领域，其中的话题种类繁多，这使得用户难以搜索到相关性高而且高质量的文本信息。在社交网络平台上，传统的信息检索方法变得耗时长而且信息检索方式难以适用。因此，在大数据时代，为了满足用户实时获取相关信息的需求，需要一种面向社交网络的新的信息检索方法。

在传统的信息检索流程中，往往是用户根据自己所需，输入关键字进行查询，系统根据用户的查询，搜索到相关的结果，并进行排序，返回给用户。而在社交网络平台上，信息产生速率快，信息以数据流的形式给出，同时用户希望系统能够自动地推送相关的信息，而不是用户通过查询来获取信息。因此，在社交网络中的信息实时个性化搜索的流程与传统的信息检索流程不尽相同。在社交网络中，信息的实时个性化搜索流程一般是用户将查询搜索以关键词的形式给出，然后系统根据用户的查询在信息流中实时地处理文本信息，将相关的信息自动地推送给用户。

由美国国家标准与技术研究院 (NIST) 主办的文本检索会议 (TREC) 是由多个测评项目组成的一个致力于解决新时代信息检索问题的测评大会，涉及智能问答、医疗诊断、实体识别、信息推荐等领域。TREC 自 2011 年起设立了微博实时推荐 (Microblog Track) [1-5] 这一个子任务，目标是解决在社交网络中信息的实时个性化搜索问题。从设立后，TREC Microblog Track 便吸引了全世界的参赛者参加，与智能实时问答 (TREC LiveQA Track) 成为了 TREC 中最火热的两个子任务。Microblog Track 的任务是针对不同用户，智能地分析用户的兴趣爱好，自动地、实时地为用户推送相关的、高质量的信息，涉及到众多科学领域，包括机器学习、自然语言处理、信息检索、人工智能等。如图2.1所示，在社交网络平台上，用户是信息的生产者，用户将源源不断地发布信息，其中包括有趣的信息和一些无用的信息。同时，用户又是信息的接收者，用户通过定制自身感兴趣的话题和

¹<http://www.statisticbrain.com/twitter-statistics/>

内容，系统智能地分析其兴趣爱好，针对不同的用户，在推特信息流中自动地、实时地为用户推荐相关的、高质量的信息。例如将饮食信息推荐给美食家、将财经信息推送给金融家、将商旅信息推送给旅行者等，同时将一些低质量无意义或者无人关注的信息丢弃。

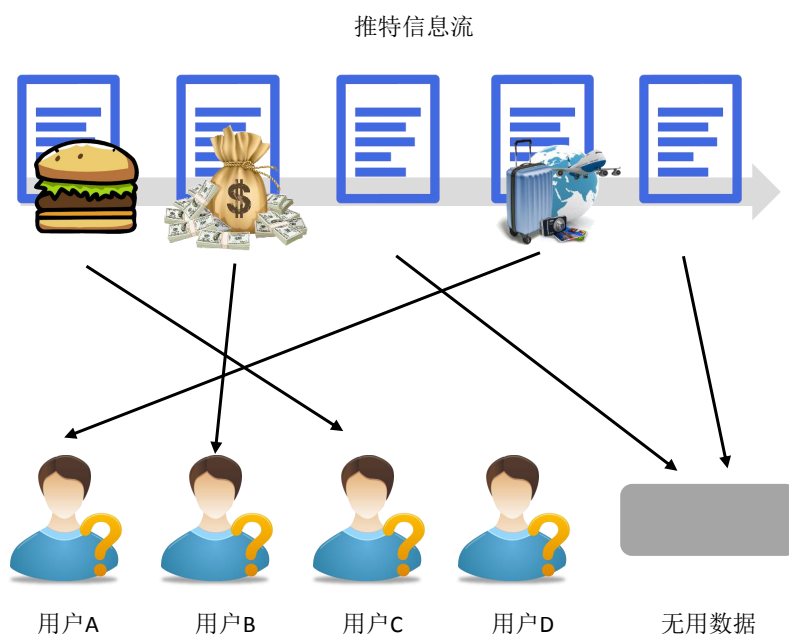


图 2.1 社交网络平台中信息的实时个性化搜索示意图

为了解决针对不同用户，为其实时地搜索出相关性强、高质量的信息，许多已有的个性化推荐^[6-9]以及协同搜索^[10-13]的工作对这个问题进行了一定的研究。但是很少有面向社交网络个性化搜索的研究，特别是对于实时搜索推荐技术的研究。目前已有的机器学习、自然语言处理、信息检索、人工智能等研究对于社交网络中的实时个性化搜索问题提供了许多帮助。文本分类以及排序的研究^[14-17]能够为了社交网络中的信息检索提供支持。同时，查询分析以及优化技术^[18-20]能够使得信息检索有着显著的提升。

然而，社交网络的环境与传统的新闻、论坛等 Web 环境有着较大的区别。因此，为了满足社交网络中用户获取信息的需求，这将需要建立新的模型、研究新的方法来解决这一问题。在社交网络中，实现信息的个性化实时推荐面临的主要挑战可以总结如下：

- **信息海量性**，在社交网络中，信息产生的速率快，网络中的每一个用户同时扮演着信息生产者、信息传播者和信息接收者的角色。如此高容量的信息流需要一个新的模型来适应持续不断变化的语义特征。

- **主题多样性**，社交网络中的信息内容包罗万象，覆盖了许多领域和话题。如果主题模型不能区分众多的话题，这将导致噪声的引入以及不准确的话题模型以及用户模型。
- **数据稀疏性**，在社交网络中，信息在不同的主题上的分布是不均匀的，在某些主题上信息量大，而在某些主题上信息是稀疏的。有效的主题模型需要解决数据稀疏性所带来的影响。
- **社交互动性**，社交网络中的用户之间有着丰富的互动信息，与传统的文本信息不同，社交网络中的信息包含了许多有价值的结构化的社交属性。适当地利用这些社交属性能够提高搜索的性能，但是这需要对这些社交属性进行相关性的选择。

为了解决上述面临的挑战，本章提出了一种基于语义扩展和文本质量的实时个性化搜索框架，该框架综合考虑了用户的偏好、语义特征和社交属性。首先，本章基于语义扩展提出了一种布尔逻辑关键词过滤（Boolean Logic Keyword Filter）的用户模型。该模型依靠外部搜索引擎提供的知识进行建立，建立的用户模型充分利用了查询扩展以及检索结果的重排序来提高推荐结果的相关性。最终的实验评估证明该模型显著地提高了检索的召回率。此外，本章还基于逻辑回归提出了一种文本质量模型，该模型利用推文的社交属性来评估其文本的质量。该模型能够对推文的文本内容，是否受到大众的认可等进行评估，因此它能帮助系统返回高质量的推文。最终的实验评估证明该模型显著地提高了检索结果的排序性能。

2.2 相关定义

本节首先将对社交网络中的实时个性化搜索的任务进行定义，并将其形式化，然后对 TREC 2015 Microblog Track 的具体任务进行描述。

社交网络中的实时个性化搜索任务可以描述如下。在社交网络中平台中，令信息流表示为 \mathbf{T} ，信息流在社交网络中随着时间不断地产生，信息流 \mathbf{T} 中包含许多种类的话题。令 \mathbf{P} 表示用户集合，每一个用户 $p_i \in \mathbf{P}$ 表示着用户感兴趣的一个话题，用户的兴趣爱好可以通过文本等形式来表示。则社交网络中的实时个性化搜索任务可以定义如下，

定义 2.1 (实时个性化搜索): 在社交网络中，给定信息流 \mathbf{T} 以及用户集合 \mathbf{P} ，实时个性化搜索的目的是针对不同的用户兴趣爱好 $p_i \in \mathbf{P}$ ，自动地实时地为其推荐高质量的、相关的信息。

根据定义2.1可知，社交网络中的实时个性化搜索需要解决的几个核心关键问题如下。首先是如何表示信息流与用户，其次是如何实现实时推荐，以及如何保

证信息的高质量和高相关性。社交网络平台上信息的实时个性化搜索任务可以通过图2.2来表示。

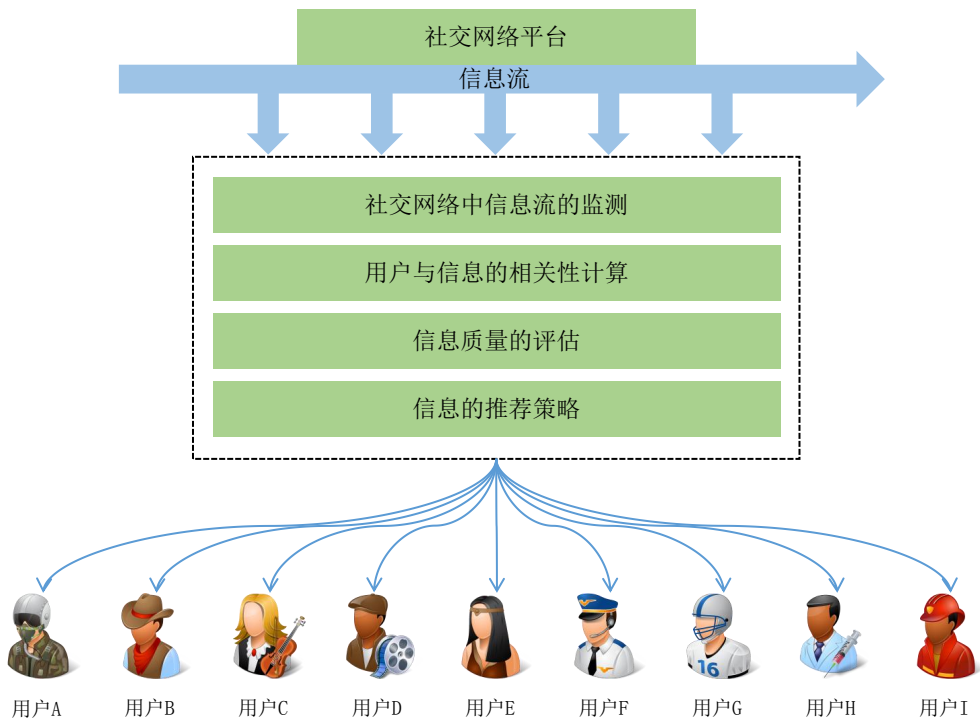


图 2.2 社交网络平台上信息的实时个性化搜索任务图

任务首先要求系统能够实时地监测社交网络中的信息流，然后对不同的用户建立兴趣模型，分析信息与用户之间的相关性。同时任务还要求系统能够对信息的质量做出评估，包括真实性、内容丰富性等。最后，系统需要采取一个良好的推送策略，使得用户不至于淹没在信息中。

TREC 2015 Microblog Track 的任务目标是**实时信息流过滤**，实质上也是实时个性化搜索的一种形式。在 Microblog Track 中，实时信息流过滤的任务目标是根据不同用户的兴趣模型来监测社交媒体中的信息流。值得注意的是，上述的兴趣模型的概念是不同于传统的临时查询。该场景中，交互流程不是用户输入一次查询，然后系统返回结果给用户。该场景不存在实际的查询，取而代之的是系统根据用户的兴趣模型主动地推送**有趣**的信息给用户。

什么样的信息是有趣的可以考虑如下两个实际的任务模型来帮助更好地理解。

- **场景 A：即时性信息推送**，在场景 A 中，用户是定位在使用移动设备的场景，用户可以即时地查看消息。系统基于用户的兴趣模型来判别消息之于用

户是否是有趣的，被系统判别为有趣的信息将实时地推送到用户的移动设备（例如手机、平板等）。场景 A 中的任务目的是上述的推送需要在消息产生的相对较短的时间内完成，该场景中的信息假设是相对较短的。

- **场景 B：周期性邮件摘要**，在场景 B 中，用户是定位在使用固定设备的场景，用户周期性的查看消息。与场景 A 相同，系统基于用户的兴趣模型来判别消息之于用户是否是有趣的。但是被系统判别为有趣的信息将被聚集成一个邮件摘要，并且周期性的推送给用户。该场景中的每一个单一信息假设是相对较短的，聚集成的邮件摘要可能很长，我们可以认为这是**个性化头条新闻**。

两个场景都假设推送给用户的消息相对较短，是短消息。因此，推送的消息是否有趣可以根据用户阅读消息的长度或者条数来进行衡量。可以看出，TREC 2015 Microblog Track 的任务目标是在两种不同的场景中来考虑社交网络平台上信息的实时个性化搜索问题。

任务中的推文信息（即定义 2.1 中的 **T**）一条推文以 JSON 格式保存，字段信息如表 2.1 所示。

表 2.1 推文 JSON 数据的部分字段信息

字段	描述
<i>created_at</i>	推文的创建时间
<i>id</i>	推文的 ID
<i>text</i>	推文的文本信息
<i>source</i>	推文的发布终端
<i>in_reply_to_status_id</i>	回复的推文的 ID
<i>user</i>	发布推文的用户信息

表 2.1 中 *created_at* 是推文创建的时间，即发布时间。*id* 是推文的 ID，是唯一的。*text* 是推文的文本信息，为短文本，不超过 140 个字符，可能存在超链接。*source* 是推文的发布平台，例如推特 Web 端，推特手机端，或者通过其他的平台转发至推特等。*in_reply_to_status_id* 是指该推文是否为转发回复其他的推文，如果不是则字段为 *null*，如果是则为转发的推文的 ID。*user* 是指发布推文的用户，也是一个 JSON 格式的数据，其字段内容在表 2.2 中给出。

表 2.2 中 *created_at* 为用户账号建立的时间。*id* 为用户的 ID，是唯一的。*screen_name* 为用户的名字。*location* 为用户的地理位置，由用户自己填写。*description* 为用户对自己的描述，为一段文本信息。*followers_count* 为用户的粉丝数，即关注该用户的用户数。*friends_count* 为用户的好友数，即用户关注

表 2.2 用户 JSON 数据的部分字段信息

字段	描述
<i>created_at</i>	用户的创建时间
<i>id</i>	用户的 ID
<i>screen_name</i>	用户的名称
<i>location</i>	用户的地理信息
<i>description</i>	用户的自我描述
<i>followers_count</i>	用户的粉丝数
<i>friends_count</i>	用户的好友数
<i>favourites_count</i>	用户收藏推文的数目
<i>statuses_count</i>	用户发布推文的数目
<i>utc_offset</i>	用户所在时区
<i>profile_image_url</i>	用户的头像
<i>lang</i>	用户的语种

的用户数目。*favourites_count* 为用户收藏的推文的数目。*statuses_count* 为用户历史上发布推文的数目。*utc_offset* 为用户所在的时区的偏差值，单位为秒。*profile_image_url* 为用户的头像，如果用户没有自定义头像，则为系统默认头像。*lang* 为用户的常用语言。

任务中的用户查询（即定义 2.1 中的 **P**）也是按照 JSON 格式保存，字段信息如表 2.3 所示。

表 2.3 用户查询 JSON 数据的部分字段信息

字段	描述
<i>topic_id</i>	用户查询的 ID
<i>title</i>	用户查询的标题
<i>desc</i>	用户查询的描述
<i>narr</i>	用户查询的详情

其中 *topic_id* 为用户查询的 ID，即用来标志一个用户感兴趣的话题。*topic_id* 为用户查询的标题，为一个简短的句子。*desc* 为用户查询的描述，为文本信息，描述用户所需要的信息。*narr* 为用户查询的详情，详细地介绍了用户对于信息需求的细节。

本节介绍了任务的定义以及相关的背景知识，包括推文信息的格式，用户的格式以及用户查询的格式等信息。

2.3 方法描述

在本节中，我们设计了一个面向社交网络的实时信息流推荐的系统框架。首先，与传统的信息检索系统相似，推文信息以及用户查询的语义特征被提取出来，数据的预处理用于过滤掉此阶段中无用的信息。其次，为了计算推文信息与用户查询之间的相似度（即相关性），基于外部语料库训练得到的词向量模型来表述提取的特征。除此之外，系统将文本的质量也考虑在内。高质量的文本往往包含高质量的信息，本节提出了一个基于逻辑回归模型的文本质量评估算法来对文本的质量进行衡量。然后，基于得出的相关性与质量，系统针对不同的用户查询对推文信息进行评估。一条相关的、有意思的信息将被归类到最相关的类别中，即归类到最相关的用户查询。最后，为了避免信息洪流、保持消息推送的实时性，本节提出了一个动态的推送策略。因此，用户将实时地收到相关性高、质量高的信息，并且能够避免淹没在信息中。

2.3.1 系统框架

目前在社交网络中，用户只能接收到其关注用户的信息推送，而且这些信息只是简单的按照时间进行排序。理想的搜索系统应当能够分析用户的兴趣爱好，并且自动地推送相关的、高质量的信息给用户。为了满足这一需求，一个面向推特的实时个性化搜索系统的框架设计如图2.3所示。

如图所示，系统主要包括四个主要的模块，分别如下所示。

- **特征提取模块**，该模块监测推特信息流，并且接收用户的搜索请求。值得注意的是，在实时个性化搜索任务中，用户的搜索请求相当于某个话题内容的订阅。用户事先输入搜索后，在相关的信息发布时实时推送给用户。推特信息流通过 TREC-API²工具监测。用户的搜索由官方提供，以 $p_i \in \mathbf{P}$ 表示。在特征提取之前，数据的预处理将用来过滤掉无用的信息。对于推特信息流，系统提取其语义特征以及社交属性；对于用户查询，系统提取搜索中的关键词作为基础特征。
- **特征表征模块**，该模块通过几种技术来扩展及表征所提取的语义特征。推文信息以及用户查询通过该模块进行了语义增强，该模块使得推文以及用户之间的相关度计算更加的方便。

²<https://github.com/lintool/twitter-tools>

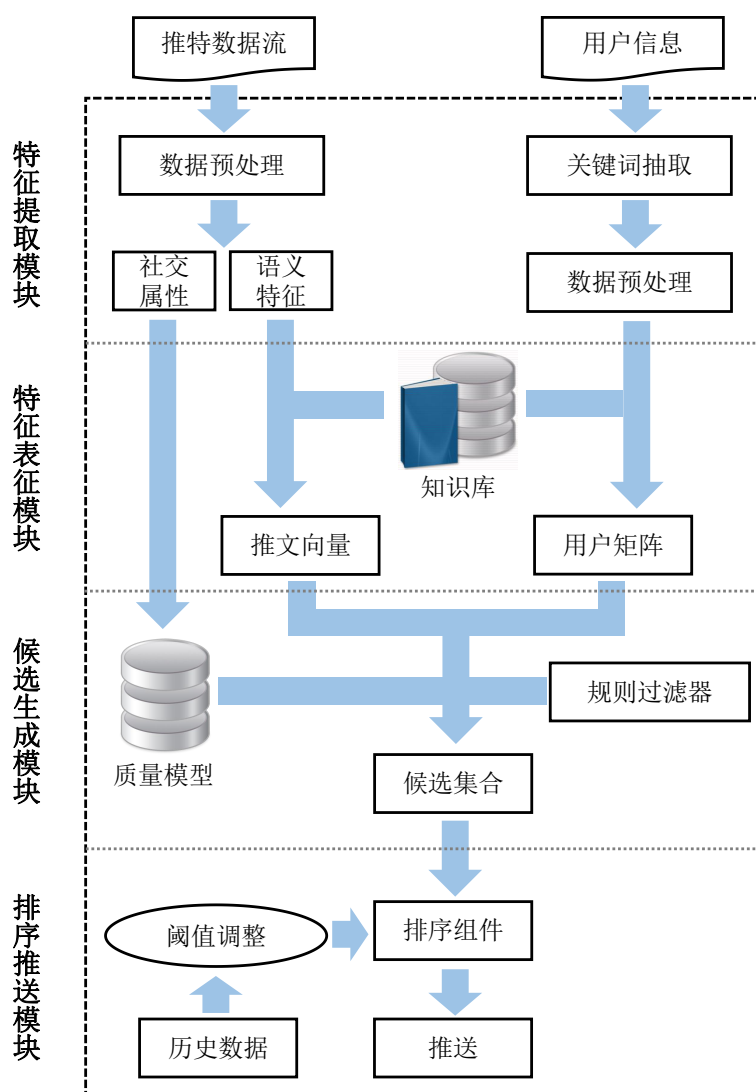


图 2.3 实时个性化搜索系统框架图

- **候选生成模块**，该模块通过语义特征以及社交属性将推文分类到与之最相关的用户。这一过程同时考虑了相关性以及推文的质量，因此用户收到的信息不仅是有趣的而且是高质量的。
- **排序推送模块**，该模块通过最终的得分来对每个用户的推文候选队列进行排序，并且利用历史数据进行阈值的动态调整。该模块能够保证用户不错过重要的高质量的信息，又不至于淹没在过多的信息中。

在之后的小节中，我们将详细地介绍每一个模块应用的算法以及实现。

2.3.2 特征提取模块

推文信息依靠监测推特的抽样信息流来获取^[21]。在获取到推文后，数据的实时预处理用来过滤掉此阶段内无效的信息，减少无意义的以及冗余的信息。推特数据流上的相关数据预处理方法包含如下一些技术，

- **语言识别**，根据任务要求，非英文的推文将被直接过滤掉来简化问题的复杂度。本系统中使用的工具为**无限元模型语言识别** (*Language Detection with Infinity Gram*)，简称为 **ldig**³。ldig 工具是一个针对短文本消息（例如推特）的语言识别器，能够支持对于 17 种语言的识别，准确率达到了 99.1%。除此之外，系统还使用了基于字符编码集的语言识别方法来区分英文句子和非英文句子。通过训练得出一个阈值，包含英文字符多的推文将被保留，其余的将被过滤掉。
- **冗余过滤**，社交网络中由于复制，转发等行为使得相同信息的推文存在着很多的冗余，如果不进行冗余过滤，那么将会有很多相同的内容推送给用户，使得用户淹没在重复的信息中。因此，对于内容相同或者相似的推文，系统只保存一个副本。为了实现该目标，一种方法是利用表 2.1 中提到的推文 *id* 字段和 *in_reply_to_status_id* 字段来识别冗余信息。如果推文是原创信息，则我们记录其 *id* 值；如果推文是转发信息，则可以检查该推文转发推文的 *id* 是否已经存在来判别是否是冗余信息。每当一条新的推文被爬取时，首先来检查推文的 *id* 来做冗余消除。该方法简单有效，能够迅速滤除掉大部分的冗余信息。但是当推文是复制其他推文的信息或者做出微小的改动，则该方法无法识别。因此，另外一种冗余消除技术，*simhash* 技术被应用。*simhash* 技术是用于处理网页冗余的技术，该方法将一个文档转换成一个数字指纹（即一串定长的 0-1 编码），被称为**哈希相似码**。两个文档的哈希相似码之间的汉明距离 (*hamming distance*) 越短，则说明两个文档之间的相似度越大。哈希相似码的计算公式如下所示，

$$s_{hash} = \text{sign}(\sum_{i=1}^n w_i \cdot \mathbf{c}_i) \quad (2.1)$$

其中 \mathbf{c}_i 是第 i 个词的哈希码，为一个定长的向量， w_i 是该词的权重，为标量，*sign* 是一个符号函数，让结果向量中的每一个比特的正数位变成 1，负数位变成 0。*simhash* 算法的过程大概如下。首先，对文档进行关键词抽取，得到特征 \mathbf{c}_i 和权重 w_i 。然后，对所有特征进行加权的位计算。最后，对得到

³<https://github.com/shuyo/ldig>

的结果进行哈希码的生成，这个产生值和具体采用的算法有关，这里采用的一个简单的符号函数。将推文转换成哈希相似码后，我们就可以利用推文之间的汉明距离来判别推文间的相似性，来进行冗余消除。基于推文的 *id* 进行冗余消除是一种高效的方法，能够解决大多数情况下的问题，但是对于复制或者修改的内容无法识别。*simhash* 算法是基于推文的内容进行冗余消除，能够有效地解决内容相似的推文冗余，但是相比于基于推文 *id* 的方法耗时会长一些，因为计算哈希相似码的过程将会有额外的计算量。在本章提出的系统中，两种算法被结合起来完成冗余消除。首先利用基于推文 *id* 的方法进行粗的冗余消除，如果无法判别则计算推文的哈希相似码后，根据汉明距离进行判别。

在数据的预处理后，我们将推文信息和用户查询的语义特征和社交属性抽取出来。对于推文，名词以及动词是句子中最重要的部分。因此，推文中的名词以及动词（除去停用词）被抽取出来作为语义特征，并且每一个词都根据其在句子中的重要性得到一个不同的权重。一条推文的语义特征可以表示成 $\mathbf{ts} = \{w_1\mathbf{t}_1, w_2\mathbf{t}_2, \dots, w_n\mathbf{t}_n\}$ ，其中 \mathbf{ts} 表示一条推文， \mathbf{t}_i 表示一个词语， w_i 为该词语的权重。词语的权重与该词语在句子中的位置、出现的频率等有关。

由于社交网络信息提供了丰富的社交行为属性，社交属性可以被用于评估推文的真实性和质量。推文是按照 JSON 格式存储的，因此获取系统所需的社交属性是非常便捷的，例如发布推文的用户、推文的转发量、推文的评论数、推文的 URL 数、推文的标签、推文中有意义的词数、推文的长度等等。这些信息都可用于进行训练，来评估推文的质量。

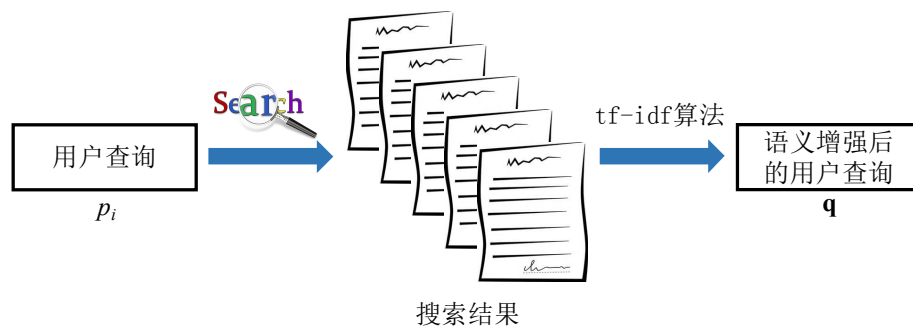


图 2.4 用户查询的语义扩展示意图

对于用户查询，对用户的兴趣爱好描述被当成一个查询，如表 2.3 所示。系统对查询进行分析，计算用户所感兴趣的推文。首先，用户查询中的关键词如同推文一样的被提取出来。但是，用户查询往往比较简短，表述能力是有限的。短文

本的检索面临着词语失配的问题，即用户查询与推文信息的词语重叠相对较少，这将导致很多描述着同一事件的文本因为使用了不同的词汇而检索失败。为了解决该问题，语义扩展方法往往被用于增强检索的性能。语义扩展方法主要分成两大类，基于搜索引擎的语义扩展和基于知识库的语义扩展。本节中系统使用的是基于搜索引擎的语义扩展技术来丰富查询的信息，大致流程如图2.4所示。在第一步中提取出来的关键词被当做搜索引擎的输入。返回的 top-k 个结果的摘要文本部分被收集用来做下一步的计算，每一个返回的结果都包含若干个词语。在整个结果集中可以使用 tf-idf 算法可以被用来计算每一个词语的权重。针对每一个用户查询，一个带有 tf-idf 值的词语列表被计算返回。语义增强后的用户查询可被写作 $\mathbf{q} = \{\mathbf{k}_1 : v_1, \mathbf{k}_2 : v_2, \dots, \mathbf{k}_n : v_n\}$ ，其中 \mathbf{k}_i 表示关键词， v_i 表示词语 \mathbf{k}_i 的 tf-idf 值。

2.3.3 特征提取模块

已有的工作对语义特征的表示模型有着很多的研究，例如**词袋模型** (*Bag of Words Model*) 等。词袋模型忽略了文本的语法和词序，仅仅将文本当作词语的聚集。该模型假设每一个词是独立的，即文本中词的概率与上下文是无关的，它简化了表示方法，但是忽略了语法和词序，在某些情况下带来了一些缺陷。基于此的一些改进算法，例如二元词袋模型、三元词袋模型等在一定程度上解决了这些缺陷。然而，词语失配问题仍然是词袋模型无法解决的问题。

为了解决词语失配问题，**词向量模型** (*Word Vector Model*) 被提出来表示语义特征。使用 *word2vec* 技术来对词语进行向量化，*gensim*⁴是一个比较方便的工具来处理该问题。为了训练词向量模型，维基百科的英文预料库被用于训练⁵。图2.2中的知识库基于维基百科英文语料库使用 *gensim* 工具训练得到。推文 \mathbf{ts} 中的每一个词语 \mathbf{t}_i ，可被表示成一个向量 $\mathbf{t}_i = (\alpha_1, \alpha_2, \dots, \alpha_m)^T$ ，其中 m 为词向量模型设置的维度（例如 200、400 等），参数 α_j 表示词向量第 j 维的值。因此，一条推文 \mathbf{ts} 可向量化为 $\mathbf{ts} = \sum w_i \mathbf{t}_i$ ，其中 w_i 为词语 \mathbf{t}_i 的权重。与处理推文的方式相似，语义增强后的用户查询 q 中的每一个关键词 \mathbf{k}_i 可以表示成 $\mathbf{k}_i = (\beta_1, \beta_2, \dots, \beta_m)^T$ ，其中参数 β_j 为关键词 \mathbf{k}_i 第 j 维的值。而且语义增强后的用户查询 $\mathbf{q} = \sum v_i \mathbf{k}_i$ ， v_i 为词语 \mathbf{k}_i 的权重。按照上述流程，推文信息以及用户查询通过词向量模型转化成了向量形式。

2.3.4 候选生成模块

依靠词向量模型，词语的失配问题得到了比较好的解决。但是该方法也是一把双刃剑，词向量模型在获取更多相关的信息的时候也会带来许多低相关的信息

⁴<http://radimrehurek.com/gensim/index.html>

⁵<https://dumps.wikimedia.org/enwiki/20160113/>

给用户，即词向量模型在提高召回率的时候降低了准确率。为了解决该问题，系统使用了一个规则过滤器**布尔逻辑关键词模型** (*Boolean Logic Keyword Model*) 来提高准确率。布尔逻辑关键词模型将用户查询表示成如下形式，

$$p = \{1 : (t_1 \parallel t_2) \&\& (t_3 \parallel t_4), 0 : t_5 \parallel t_6\} \quad (2.2)$$

其中 p 代表一个用户查询，其中包含两个域 1 和 0。值为 1 的域代表着该用户查询检索到的结果中必须出现的词，值为 0 的域代表着该用户查询检索到的结果中不一定须要出现的词，但是命中会增加推文信息和用户查询之间的相关性。符号 \parallel 表示逻辑上的“或”关系，代表多个词出现某一个即可；符号 $\&\&$ 表示逻辑上的“与”关系，代表多个词须要同时出现才满足。例如公式 (2.2) 表示能够通过过滤器的推文中必须包含词语 t_1 或者 t_2 ，并且包含 t_3 或者 t_4 。如果推文中包含 t_5 或者 t_6 ，则会增加推文信息与用户查询之间的相关性。系统根据语义扩展后的用户查询中的词语的 **tf-idf** 值来生成规则过滤器。我们为规则过滤器设置两个参数阈值 η 和 γ ($\eta < \gamma$)。假设 tf-idf_{\max} 表示一个用户查询 p 中的最大 **tf-idf** 值。则阈值 γ 设置为 $0.7 \cdot \text{tf-idf}_{\max}$ ，阈值 η 设置为 $0.9 \cdot \text{tf-idf}_{\max}$ ，这是一个经验性的参数设置。对于 **tf-idf** 值大于 η 的词语将被加入到 p 中域为 1 的列表中，对于 **tf-idf** 值大于 γ 又小于 η 的词语将被加入到 p 中域为 0 的列表中，对于 **tf-idf** 值小于 γ 的词语将不作处理。

所有从推特信息流中监测到的推文首先都会经过根据布尔逻辑关键词模型生成的规则过滤器。如果推文不与任何规则过滤器匹配，则该条推文将被直接丢弃。如果推文通过了某个规则过滤器，则推文信息与该用户查询之间的语义得分将根据它们之间的相似度计算得出。然后推文将根据公式 (2.2) 以及语义得分分类到某一个用户查询中。已有的研究主要有两类计算语义得分的方法来评估推文信息与用户查询之间的相关性。

- **基于 tf-idf 的相似度**，根据推文信息和用户查询生成的向量以及 **tf-idf** 权重进行相似度计算，计算公式如下所示，

$$\text{sim} = \frac{\mathbf{ts}^T \cdot \mathbf{q}}{\|\mathbf{ts}\| \cdot \|\mathbf{q}\|} = \frac{\left(\sum_{i=1}^m w_i \mathbf{t}_i \right)^T \cdot \sum_{j=1}^m v_j \mathbf{k}_j}{\sqrt{\left(\sum_{i=1}^m w_i \mathbf{t}_i \right)^T \left(\sum_{i=1}^m w_i \mathbf{t}_i \right)} \sqrt{\left(\sum_{j=1}^m v_j \mathbf{k}_j \right)^T \left(\sum_{j=1}^m v_j \mathbf{k}_j \right)}} \quad (2.3)$$

其中 \mathbf{ts} 表示推文信息的向量， \mathbf{q} 表示用户查询的向量。 w_i 为词向量 \mathbf{t}_i 的权重， v_j 为词向量 \mathbf{k}_j 的权重。

- **基于 BM25 的相似度**，利用算法 Okapi BM25 的权重函数来评估推文信息和用户查询之间的相似度。算法 Okapi BM25 是一个基于词袋模型的算法，根

据文档中出现的查询词的频率进行权重计算。推文信息与用户查询之间的相似度计算如下,

$$sim = \sum_{q_i \in Q} idf(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)} \quad (2.4)$$

其中 D 表示一个文档, 即一条推文信息, Q 表示用户查询。函数 $f(q_i, D)$ 表示查询词 q_i 在文档 D 中的词频, $|D|$ 表示文档 D 的长度, $avgdl$ 表示文档集中文档的平均长度, k_1 以及 b 为调整参数。系统一般经验性的设置 $k_1 = 2$, $b = 0.75$ 。

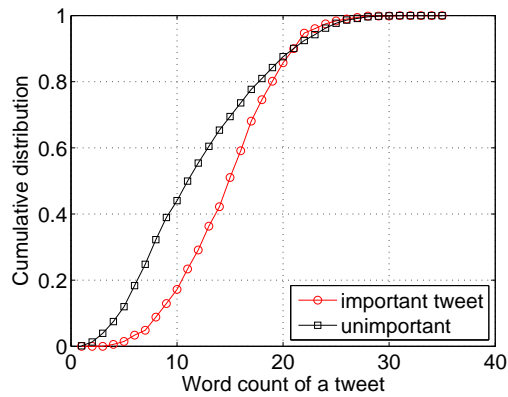
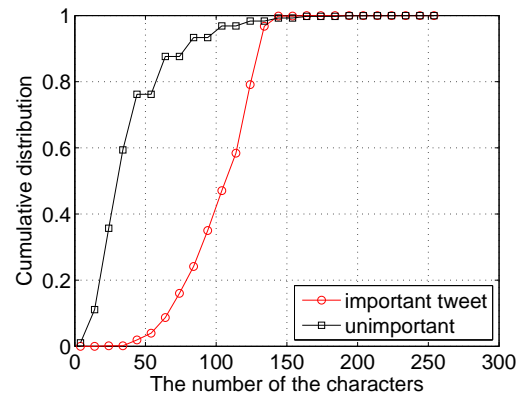
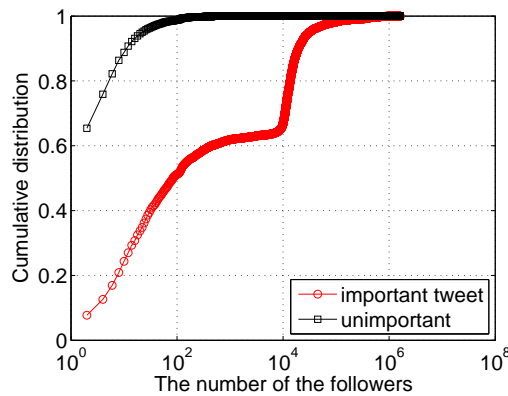
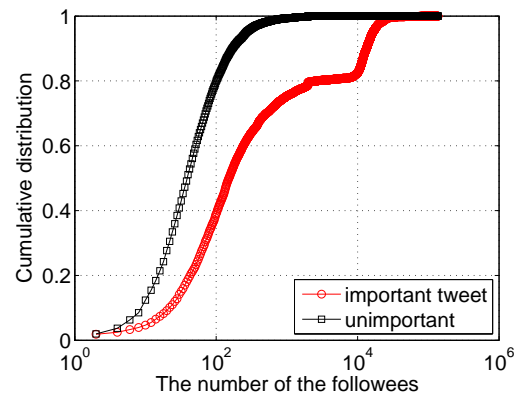
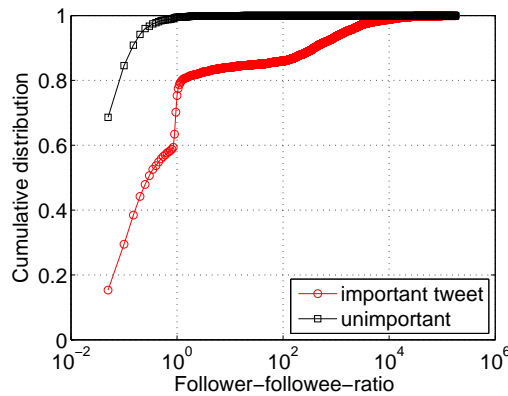
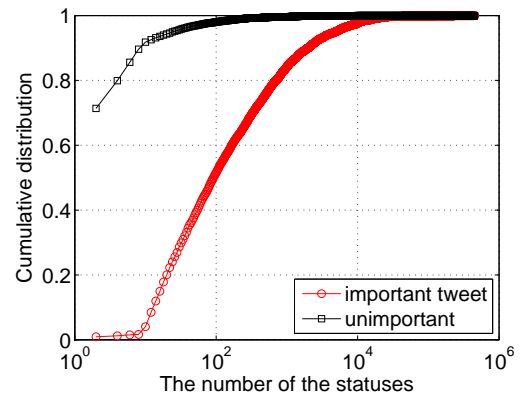
以上是系统对推文信息和用户查询之间相关性的处理, 下面介绍系统对于推文的质量的评估。在特征提取模块中抽取的社交属性可被用来训练推文的质量模型 (*Quality Model*)。我们首先人工地标注推文的质量得分, 标注不依据推文的类别, 单纯考虑推文是否能够带来信息量。质量高的推文标注 1, 质量低的推文标注 0。如果推文能提供更多的信息, 并且撰写得更完整, 则推文会获得高的得分。我们使用逻辑回归模型 (*Logistic Regression Model*) 来对文本的质量模型进行建模, 模型如下所示,

$$h_{\theta}(\mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \quad (2.5)$$

其中 $\mathbf{x} = (x_1, x_2, \dots, x_n)$ 是抽取的社交属性, $\theta = (\theta_1, \theta_2, \dots, \theta_n)$ 为待训练的参数。实现一般来说, 用户更倾向于接受文本质量高的、由权威发布的信息, 基于以上假设, 我们选取如下的社交属性作为特征。

- **用户粉丝数** (*FollowerCnt*), 表示发布推文信息的用户的粉丝数。粉丝数高的用户表示该用户在社交网络中是受欢迎的, 因此该用户发布的推文信息很大概率上是高质量的信息。
- **用户推文数** (*StatusCnt*), 表示发布推文信息的用户历史上发布的推文数。用户发布推文的数目预示着用户的活跃度, 而一个活跃的用户更可能发布高质量的信息。
- **推文转发数** (*RetweetCnt*), 表示推文的转发数。一条推文的转发数越大说明这一条推文越受人欢迎, 质量越高。
- **推文转发等级** (*RetweetLvl*), 表示对转发数使用对数函数后的转发等级。
- **推文点赞数** (*CollectCnt*), 表示推文的点赞数。用户如果认为一条推文是吸引人的, 信息丰富的, 则可以点赞或者收藏。

- **推文词语数 (*WordCnt*)**, 表示推文中除去停用词的文本长度, 即词语数。一般来说, 长的文本相对于短的文本的信息量会大一些。
- **推文字符数 (*CharCnt*)**, 表示推文中除去停用词的文本的字符数。
- **短链接数 (*UrlCnt*)**, 表示推文中短链接的数目。信息量丰富的文本往往会在推文末尾给出一个短链接。

图 2.5 *WordCnt* 的累积分布函数图 2.6 *CharCnt* 的累积分布函数图 2.7 *FollowerCnt* 的累积分布函数图 2.8 *FolloweeCnt* 的累积分布函数图 2.9 *FFR* 的累积分布函数图 2.10 *StatusCnt* 的累积分布函数

为了检验抽取出来的社交属性的有效性，我们计算了人工标注数据集中的各个社交属性中的**累计分布函数** (*cumulative distribution function*) 来进行观察。推文根据其文本质量被标记成 1 或者 0，其中 1 表示这是一条质量高的推文，反之则表示这是一条质量不高的推文。社交属性的累计分布函数结果如图2.5至图2.10所示，两个分类的累计分布函数曲线显示了特征的区分能力。从结果图可知，*CharCnt*、*textitFolloweeCnt*、*FFR* 以及 *StatusCnt* 是区分推文是否高质量的有效的特征。由图2.6可知，超过 95% 的低质量的推文的字符数都小于 100，而超过 50% 的高质量推文的字符数都大于 100。由图2.7和图2.9可知，拥有高粉丝数和高粉丝-好友比的用户更可能发布高质量的推文。由图2.10可知，活跃用户更有可能发布高质量的推文。因此，抽取的社交属性能够被用于训练评估推文质量的分类器。

在计算得到推文信息和用户查询之间的语义相似度和推文的质量得分后，二者被综合起来用来评估推文的最终得分，详细流程在下一个小节中介绍。

2.3.5 排序推送模块

基于语义特征和社交属性，我们得到了推文信息和用户查询的语义相似度以及推文的质量得分。以上两种得分的值域为 $[0,1]$ 。对于一个用户查询，一条推文的最终得分考虑了以上的两个方面，形式化为如下所示，

$$f(\mathbf{ts}, \mathbf{q}) = \text{sim} \cdot h_{\theta}(x) \quad (2.6)$$

其中 *sim* 为语义相似度得分，参见公式 (2.3) 和公式 (2.4)。基于 *tf-idf* 和基于 *BM25* 算法的语义相似度得分在系统中是可选的。 h_{θ} 为推文的质量得分，参见公式 (2.5)。

当计算完一条推文和各个用户查询之间的分数后，推文将分类到得分最高的用户查询作为候选。候选队列中的推文按照得分 $f(\mathbf{ts}, \mathbf{q})$ 进行排序。按照任务需求，如果推文对于一个用户查询是相关的且高质量的，那么系统将把该推文推送给感兴趣的用户。但是如果一直推送给用户信息，那么将导致信息淹没。任务要求系统在保证用户不被信息淹没的情况下，同时实时地推送给用户消息，这是一个约束满足问题。问题可形式化为，用户每天接受信息的数量是一定的，系统需要根据当前队列中的信息来决策何时推送给用户。系统提出了一种**动态阈值调整算法**来保证相关的高质量推文及时的推送，且在时间上是均匀分布的。算法根据一个用户查询的选集队列的历史数据，可以得到一个最高得分 f_{\max} ，设计一个梯度阈值函数 $\zeta(f_{\max}, t)$ 如下所示，

$$\zeta(f_{\max}, t) = \begin{cases} (0.9 - d) \cdot f_{\max} & d < 0.4 \\ 0.5 \cdot f_{\max} & d \geq 0.4 \end{cases} \quad (2.7)$$

其中 d 表示衰减值, 且 $d = c \cdot \text{floor}(t/2)$ 。式中的 c 表示衰减系数, floor 为向下取整函数, t 为一天中的时间, $t \in [0, 24]$ 。观察公式 (2.7) 可知, 阈值的变化与队列中的最高得分 f_{\max} 和时间 t 相关。为了得到相关性强、质量高的推文, 阈值在开始时相对较高。在 $t = 0$ 时刻, 阈值为 $0.9 \cdot f_{\max}$ 。为了给用户带来更多更全面的信息, 阈值随着时间的变化不断减小。为了避免推送不相关或者低质量的信息, 阈值在降低一定程度后不再变化, 最终保持在 $0.5 \cdot f_{\max}$ 。例如, 经验性地设置衰减系数 $c = 0.05$, 当 $t = 16$ 时, 阈值衰减为 $0.5 \cdot f_{\max}$ 后不再变化。动态阈值调整算法的思想是在一开始设置一个相对较高的阈值来获取高质量的推文, 然后降低阈值来保证信息的数量。当推文的最终得分超过当时的阈值, 则系统将立即推送该信息给相应的用户。由上述的各个模块描述可知, 实时个性化搜索算法如算法所示, 且阈值将随时间动态变化。

算法 2.1 实时个性化搜索算法

已知: $\mathbf{T} = \{\mathbf{ts}_1, \mathbf{ts}_2, \dots, \mathbf{ts}_N\}$, $\mathbf{P} = \{p_1, p_2, \dots, p_M\}$, Z

求: $\mathbf{R} = \{\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_M\}$

```

1: for all  $p_i \in \mathbf{P}$  do
2:      $\mathbf{q}_i = \{\mathbf{k}_1 : v_1, \mathbf{k}_2 : v_2, \dots, \mathbf{k}_n : v_n\}$ 
3:      $p_i = \{1 : (\mathbf{t}_1 || \mathbf{t}_2) \&\& (\mathbf{t}_3 || \mathbf{t}_4), 0 : \mathbf{t}_5 || \mathbf{t}_6\}$ 
4: end for
5: for all  $\mathbf{k}_j$  do
6:      $\mathbf{k}_j = (\beta_1, \beta_2, \dots, \beta_m)^T$ 
7:      $\mathbf{q}_i = \sum v_i \mathbf{k}_i$ 
8: end for
9: while  $|\mathbf{R}_i| < Z \&\& \mathbf{T} = \emptyset$  do
10:     $\mathbf{ts}_i = \text{pop}(\mathbf{T})$ 
11:     $\mathbf{ts}_i = \{w_1 \mathbf{t}_1, w_2 \mathbf{t}_2, \dots, w_n \mathbf{t}_n\}$ 
12:    if  $\mathbf{ts}_i$  matches  $p_j$  then
13:        for all  $\mathbf{t}_k \in \mathbf{ts}_i$  do
14:             $\mathbf{t}_k = (\alpha_1, \alpha_2, \dots, \alpha_m)^T$ 
15:        end for
16:         $\mathbf{ts}_k = \sum w_i \mathbf{t}_i$ 
17:         $f(\mathbf{ts}_i, \mathbf{q}_j) = \text{sim} \cdot h_\theta(x)$ 
18:    end if
19:    if  $f(\mathbf{ts}_i, \mathbf{q}_j) \geq \zeta$  then
20:         $\mathbf{R}_j = \mathbf{R}_j \cup \mathbf{ts}_i$ 
21:    end if
22: end while

```

2.4 实验分析

本节中，我们通过 TREC 2015 Microblog Track 的实时测评对系统进行了多方面的验证。并对实验结果进行了分析。首先，本节介绍了实验的设置，其次介绍了评测标准，最后对实验结果进行了分析。本节中的实验基于八核、16GB 内存、SSD 硬盘的机器上运行，操作系统为 Ubuntu 14.04 64 位系统。系统主要基于 Python 和 Java 语言实现，Python 版本为 2.7，JDK 版本为 1.8 版本。

2.4.1 数据集及实验环境

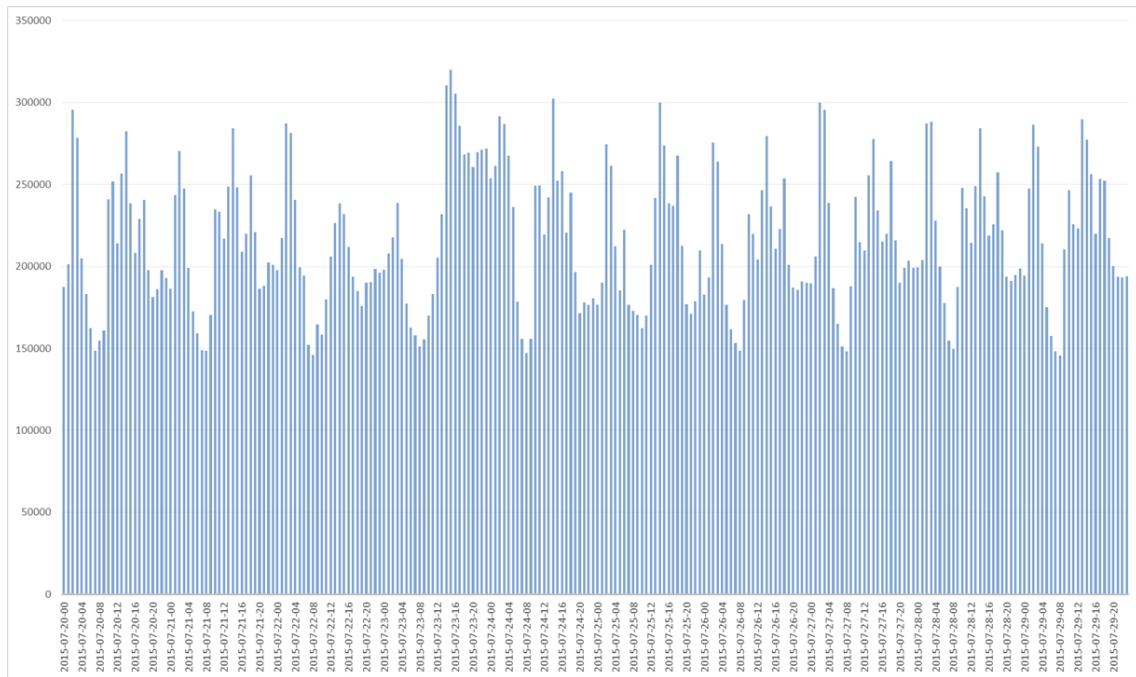


图 2.11 推特数据流的分布

为了评估系统的稳定性和算法的有效性，TREC 2015 Microblog Track 的实时测评采用了 10 天的推特信息流数据，时间从 2015 年 07 月 20 日 00:00:00 UTC 至 2015 年 07 月 29 日 23:59:59 UTC，时区为世界标准时间。数据集总共包含 51,770,318 条真实推文，共有 225 个用户查询由官方提供。推文的时间分布如图 2.11 所示，横轴代表时间，纵轴代表推文的数量。从图中可知，推文的分布具有一定的周期性，周期大约为 12 个小时。

2.5 本章小结

第三章 基于卷积神经网络的文本分类研究

3.1 研究动机

3.2 相关定义

3.3 方法描述

3.4 实验分析

3.5 本章小结

第四章 社交网络中传播效率最大化

影响力最大化 (Influence Maximization) 在许多的社交网络应用中扮演着举足轻重的角色, 例如市场营销、商业活动以及竞选活动等等。研究信息传播模型以及机理的一个典型目的便是市场营销。在现实社会中, 人或者事物都是通过某种关系来连接, 形成了一个巨大的网络。因此, 信息传播可以选择一小部分节点激活做为种子节点集合, 然后通过信息的传递, 在整个网络中产生一个大范围的影响。从技术方面来说, 影响力最大化问题是在给定网络和传播模型的条件下, 研究如何选择种子节点集合使得传播的影响范围最大化。影响力最大化问题由于其的问题真实、应用性广的特点, 吸引了许许多多的研究, 包括信息传播模型的研究和计算影响力最大化的方法。但是, 仍然有着一些需求在实际应用中得不到满足。

在信息传播过程中, 一个被激活的节点将会尝试在下一个时刻激活它的邻居节点。所以, 除去种子节点外, 每一个被激活的节点在被激活之前都会有一个时间延迟, 我们称之为**传播时延 (propagation time delay)**。如果一个节点在信息传播结束时, 仍然未被激活, 则该节点的传播时延可以看作无穷大。在传统的影响力最大化问题中, 传播时延没有被考虑在内, 而研究整个网络的传播时延是非常有意义的, 它可以度量选择的种子节点集合的传播效率。出于这种需求, 本章提出了一个新的问题, 传播效率最大化问题。该问题将传播时延考虑在内, 在给定网络和传播模型的条件下, 研究如何选择种子节点集合使得传播效率最大化。传播效率最大化问题与影响力最大化问题虽然相似, 但是两个问题的侧重点不尽相同。传统的影响力最大化问题研究的是如何使得传播的范围最广, 不考虑节点的传播时延。而本章提出的传播效率最大化问题将传播时延考虑在内, 研究如何使得传播效率最大化。

本章主要的工作可以总结如下。首先, 基于传统的影响力最大化问题, 我们将传播时延考虑在内了来探究传播效率, 提出了传播效率最大化问题, 研究给定网络和传播模型的条件下, 研究如何选择种子节点集合使得传播效率最大化。其次, 本章证明了传播效率最大化问题是一个 **NP-hard** 问题, 而且在独立级联模型下计算传播效率的过程是一个 **#P-hard** 的问题。然后, 我们证明了传播效率函数在独立级联模型下是**子模 (submodular)** 的。最后本章设计了三个算法来解决提出的传播效率最大化问题, 并且在真实数据集上验证了这些算法。实验结果展示了所提出的算法的性能。

本章的内容组织如下: 第4.1节介绍了研究动机, 讨论了传统的影响力最大化问题的不足和考虑了传播时延的传播效率最大化问题的意义。第4.2节介绍了相关定义, 对本章中相关概念和所提出的问题进行了符号化的定义。第4.3介绍了方法

描述，详细地阐述了本章解决问题的方法以及相关证明过程。第4.4节进行了实验分析，设计了一系列的实验，验证了本章所提出的方法，并对实验结果进行了分析。最后，第4.5对本章的内容进行了总结。

4.1 研究动机

随着社交网络的兴盛发展，信息传播的速率变得越来越快。在传统的媒体环境下，一条消息需要通过较长时间才能传播到一定的范围。在社交网络中，个人通过关系（例如关注、好友等）连接形成网络，信息在网络中通过转发、复制等行为进行传播，信息传播的速率极大的加快了。在传统的媒体环境下，个人都是通过单一信息源（例如新闻、论坛等）来接收信息，接收信息的渠道受限，信息在个体之间难以形成传播扩散。而在社交网络环境下，个人既可以是信息的接收者也可以是信息的发布者，个人可以接收到在网络中与之相连的个人推送的信息。与此同时，信息通过个人构成的网络进行传播。影响力最大化问题是信息传播中的一个典型问题，是在给定网络和传播模型的条件下，研究如何选择种子节点集合使得传播的影响范围最大化。

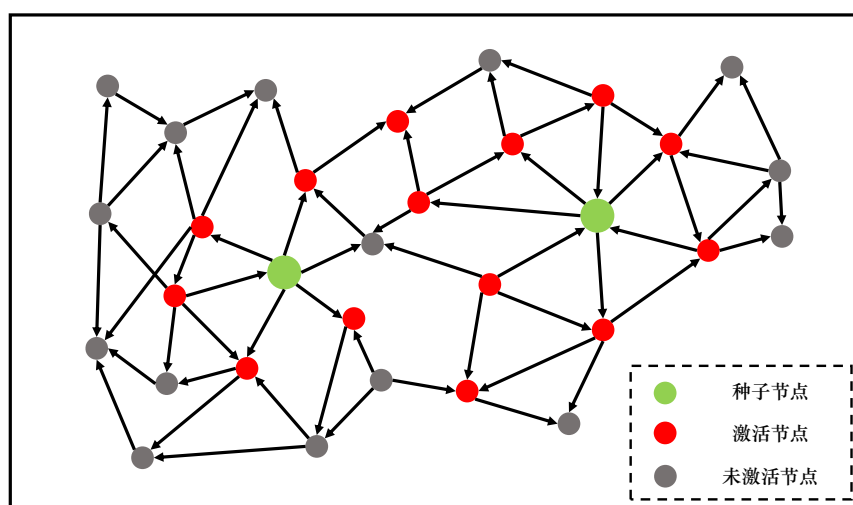


图 4.1 信息传播以及影响力最大化示意图

图4.1为一个信息传播以及影响力最大化示意图，其中的点代表着个体，边代表着社交关系。图中所示的例子为选中了绿色节点为种子节点，进行信息传播后最终产生的传播范围。红色的节点代表接受了信息，灰色的节点代表未接受信息。影响力最大化的问题即是研究如何选取初始的种子节点集合，使得传播范围最大化。

市场营销是研究信息传播模型以及机理的一个典型驱动。市场营销的过程可以简述如下，一个公司希望在社交网络中通过“口碑效应”推动一款新产品或者

一种新理念。一种有成本效益的方法是寻找到整个网络中有影响力的个人，然后投入资源来使他们接受这款产品，例如赠送样品、免费试用、优惠折扣等。这样的举措是希望这些有影响力的个人在接受新产品或者新理念后，能够驱使社交网络中的其他个人也来接受新产品或者新理念，然后在社交网络中产生一个大的级联效应，从而使得更多的个体接受新产品或者新理念。为了达到市场营销这一目的，我们需要对两个重要的问题进行研究：(1) 如何对网络中的信息传播过程进行建模，包括模型参数的学习等；(2) 如何在给定的传播模型的条件下，设计一个有效的方法来寻找能够最大化影响力的节点集合。本章着重对第二个问题进行研究。

影响力最大化问题作为市场营销的一种算法技术首先被 Domingos 和 Richardson^[22] 所提出，该问题基于马尔科夫随机场的概率框架。而后，Kempe 等人^[23] 首次将影响力最大化问题形式化成为一个离散的随机优化问题。信息传播的过程可以简述如下，在一个给定的网络中，选择部分节点做为种子集合，这些种子节点将会按照一定的规则去激活它们的邻居节点。被激活的节点在下一时刻将拥有能力去激活它们的邻居节点，这个过程将一直持续到没有新的节点可以被激活，整个信息传播的过程才会停止。影响力最大化问题是在给定网络和传播模型的条件下，研究如何选择种子节点集合使得传播的影响范围最大化，即激活的节点数目最大化。从上述信息传播过程的描述中，我们可以得知，一个被激活的节点会在被激活的下一时刻尝试激活它的邻居节点。因此，除去种子节点外，网络中的节点在被激活前都会有一个时间延迟，我们称之为传播时延。如果一个节点在信息传播结束时，仍然未被激活，则该节点的传播时延可以看作无穷大。而影响力最大化问题仅仅考虑了传播范围，忽略了节点的传播时延。在真实的场景中，例如市场营销、商业活动、竞选活动等，传播时延在信息传播中是一个非常重要的因素。为了让其他人接受自己的新产品或者新理念，人们总是希望能够尽快地将消息传播到群体中。我们以**传播效率** (*influence efficiency*) 来表示传播时延的倒数，传播时延小，则传播效率大。如果节点在信息传播结束时仍然未被激活，那么该节点的传播效率则为 0。以上是针对单个节点的传播时延和传播效率的分析，下面我们对整个网络的传播时延和传播效率进行分析。在给定一个传播网络和初始的种子节点集合的情况下，如果整个网络中所有节点的传播效率高，这就意味着在信息传播过程中，网络中的节点将被迅速地激活。我们设想如下，针对传统的影响力最大化问题有两种选择种子节点集合的策略，它们有着相同的传播范围，即能够在信息传播过程结束时能够激活相同的节点数目。但是，这两种策略中，网络中的传播时延可能是不同的，即不同策略在同一网络中的传播效率是不同的，而这一问题在传统的影响力最大化问题中是没有讨论的。

4.2 相关定义

在本节中，第4.2.1节首先对**独立级联模型** (*Independent Cascade Model*) 以及在独立级联模型下的影响力最大化问题进行回顾，然后介绍了几种解决影响力最大化问题的方法，并且对这些算法进行分析，包括影响力函数期望的单调性和子模性等。其次，第4.2.2节提出了**传播效率最大化** (*Influence Efficiency Maximization*) 问题，该问题是基于传统的影响力最大化问题，将传播时延考虑在内，研究如何使得整个网络的传播效率最大化。同时，第4.2.2节对传播效率最大化问题进行了形式化的描述，分析了传播效率最大化和影响力最大化问题的区别。

为了便于参照，表4.1中列出了频繁使用的符号。

表 4.1 常用符号列表

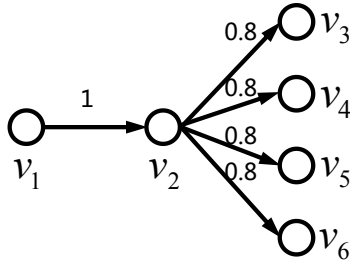
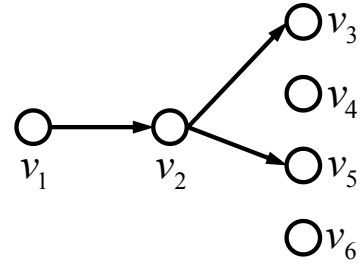
符号	描述
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	\mathcal{G} 是社交网络构成的图， \mathcal{V} 是节点集合， \mathcal{E} 是边的集合
$\mathcal{H} = (\mathcal{V}, \mathcal{Z})$	\mathcal{H} 是基于图 \mathcal{G} 生成的超图 (参见4.3)， \mathcal{V} 是节点集合， \mathcal{Z} 是超边集合
n	\mathcal{G} 或者 \mathcal{H} 的节点的数目
m	\mathcal{G} 的边的数目
k	种子节点集合的大小
$p_{u,v}^{\mathcal{G}}$	节点 u 激活节点 v 的概率
$I(S)$	种子节点集合 S 的影响力
$RR(v)$	节点 v 的反向可达集合 (参见定义4.1)
$e_{u,v}$	节点 u 到节点 v 的传播效率 (参见公式 (4.4))
$T(S)$	种子节点集合 S 在图 \mathcal{G} 中的传播效率 (参见公式 (4.5))
$T'(S)$	种子节点集合 S 在超图 \mathcal{H} 中的传播效率 (参见算法4.3)

4.2.1 传播模型以及影响力最大化问题

本章采用一种广泛采用的信息传播模型，独立级联模型，进行传播影响的研究。在该模型下，一个社交网络可被建模表示为一个有向图 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ，其中 \mathcal{V} 代表网络中的个体， \mathcal{E} 表示个体之间的社会关系。此外，图中的每一条边 $(u, v) \in \mathcal{E}$ 上都关联着一个传播概率 $p_{u,v}^{\mathcal{G}}$ ，表示着节点 u 到节点 v 的影响力度。如果传播概率 $p_{u,v}^{\mathcal{G}}$ 越大，则节点 u 更加可能激活节点 v 。如果图 \mathcal{G} 与上下文无关，本章则用 $p_{u,v}$ 来表示节点 u 到节点 v 的传播概率。

独立级联模型描述了一个直观的信息传播过程，其过程如下。在独立级联模型下，网络中的个体会被其邻居所影响，这些影响之间是独立的。给定一个种子

节点集合 $S \subseteq \mathcal{V}$ ，信息传播在独立级联模型下是如下运作的。定义 S_t 为在 $t \geq 0$ 的时刻激活的节点集合。显然，在 $t = 0$ 时刻时，满足 $S_0 = S$ 。在 $t + 1$ 时刻，每一个在 t 时刻被激活的节点 $u \in S_t$ 会独立地去尝试激活它的出度边指向的未被激活的邻居节点 $v \in \mathcal{V} \setminus \cup_{0 \leq i \leq t} S_i$ ，激活节点 v 的概率等于 $p_{u,v}$ 。当节点 u 尝试了去激活所有它的出度边指向的节点后，它在信息传播的之后过程中将不再会有机会去激活。即在 t 时刻被激活的节点 u 只会在 $t + 1$ 时刻去尝试激活它的出度边指向的邻居节点。当 t 满足 $S_t = \emptyset$ 时，整个信息传播过程停止。

图 4.2 社交网络中的信息传播概率图 \mathcal{G} 图 4.3 随机实例图 g

以图4.2中的社交网络 \mathcal{G} 为例，考虑种子节点集合 $S = \{v_1\}$ 的信息传播过程。图4.2中边上的数值代表着节点之间的传播概率 $p_{u,v}^{\mathcal{G}}$ 。整个信息传播的过程可以描述如下。在 $t = 0$ 时刻，由于节点 v_1 是种子节点集合中唯一的节点，因此节点 v_1 被激活。在 $t = 1$ 时刻，因为节点 v_1 在 $t = 0$ 时刻被激活，而且图 \mathcal{G} 中存在一条从 v_1 到 v_2 概率为 1 的边，节点 v_1 将依概率 1 去激活节点 v_2 。因此，节点 v_2 将在 $t = 1$ 时刻被激活， $S_1 = \{v_2\}$ 。此后，在 $t = 2$ 时刻，节点 v_2 将尝试去激活节点 v_3 、 v_4 、 v_5 以及 v_6 。假设这一次信息传播过程如图4.3所示，节点 v_3 和 v_5 被激活，则 $S_2 = \{v_3, v_5\}$ 。然后，在 $t = 3$ 时刻，由于被激活的节点没有后继节点可被激活，整个信息传播过程在此时刻停止。定义 $I(S)$ 为在种子节点集合是 S 的条件下，整个信息传播过程中激活的节点数目，代表信息传播过程中的影响力。在上述的图 \mathcal{G} 的一次信息传播过程中，影响力 $I(S) = 4$ 。

给定一个种子节点集合 S ，定义 $\mathbb{E}_{\mathcal{G}}[I(S)]$ 表示种子节点集合在图 \mathcal{G} 中影响力的期望，它等于以种子节点集合为传播源，在图 \mathcal{G} 中信息传播结束时激活节点数目的期望值。在独立级联模型下，影响力最大化问题的目标是寻找一个大小至多为 k 的种子节点集合，使得影响力函数的期望值 $\mathbb{E}_{\mathcal{G}}[I(S)]$ 最大化。给定一个输入 k ，影响力最大化问题可以形式化为如下，

$$\begin{aligned} S^* &= \arg \max \mathbb{E}_{\mathcal{G}}[I(S)] \\ \text{s.t. } S &\subseteq \mathcal{V}, |S| = k \end{aligned} \quad (4.1)$$

以图4.2的社交网络 \mathcal{G} 为例，给定输入 $k = 1$ 来考虑影响力最大化问题。为了解决例子中的影响力最大化问题，根据公式4.1所示，我们需要计算所有 $k = 1$ 的种子节点集合的影响力期望值，即每个节点的影响力期望值。对于节点 v_1 组成的种子节点集合，其影响力期望值 $\mathbb{E}_{\mathcal{G}}[I(\{v_1\})] = 1 + 1 + 4 \times 0.8 = 5.2$ 。对于种子节点集合 $\{v_2\}$ ，影响力期望值 $\mathbb{E}_{\mathcal{G}}[I(\{v_2\})] = 1 + 4 \times 0.8 = 4.2$ 。对于其他的种子节点集合， $\mathbb{E}_{\mathcal{G}}[I(\{v_3\})] = \mathbb{E}_{\mathcal{G}}[I(\{v_4\})] = \mathbb{E}_{\mathcal{G}}[I(\{v_5\})] = \mathbb{E}_{\mathcal{G}}[I(\{v_6\})] = 1$ 。由此，我们可以得出，在给定图 \mathcal{G} 以及 $k = 1$ 的条件下，影响力最大化问题的最优解为 $S^* = \{v_1\}$ 。

在上述的例子中，因为图 \mathcal{G} 的结构简单，并且种子节点集合的大小 $k = 1$ ，所以能够直接计算得出种子节点集合的影响力期望值，从而选择得出最优解。而在实际情况下，图 \mathcal{G} 的结构往往会复杂得多，而且 $k > 1$ ，很难直接计算种子节点集合的影响力期望值。Kempe 等人^[23] 首先证明了在独立级联模型下，影响力最大化问题是一个 NP-hard 问题。因此，直接计算出有影响力的节点是十分困难的。为了解决此问题，Kempe 等人又证明了在独立级联模型下，影响力函数的期望 $\mathbb{E}_{\mathcal{G}}[I(S)]$ 是单调的以及子模的。这两个性质为求解影响力最大化问题的近似算法提供了理论保证。形式上，一个单调的函数对于任意的节点 u 和任意集合 S ，都满足 $f(S \cup \{u\}) \geq f(S)$ 。而一个子模的函数对于任意的节点 u 以及任意的两个集合 $S \subseteq W$ ，都满足 $f(S \cup \{u\}) - f(S) \geq f(W \cup \{u\}) - f(W)$ 。针对具有单调性以及子模性的函数，Nemhauser 等人^[24] 提出了一种朴素的贪心算法来解决此类问题。算法的核心思想是首先从一个空的种子节点集合 $S = \emptyset$ 开始，重复地选择当前边际收益（即 $f(S \cup \{u\}) - f(S)$ ）最大的节点 u 加入到种子节点集合 S 中，直到满足种子节点集合的大小为 k 时结束迭代。选择节点 u 的准则可以形式化如下，

$$u = \arg \max_{w \in V \setminus S} (\mathbb{E}_{\mathcal{G}}[I(S \cup w)] - \mathbb{E}_{\mathcal{G}}[I(S)]) \quad (4.2)$$

Nemhauser 等人证明了朴素的贪心算法得出解以 $1 - 1/e$ 的因子近似于最优解，即任意一个由贪心算法的出来的解 S 都满足 $I(S) \geq (1 - 1/e)I(S^*)$ ，其中 S^* 代表最优解， e 为自然常数。虽然贪心算法的核心思想比较简单，但是由于计算影响力期望值的过程是 #P-hard^[25] 的问题，因此实现该算法并不是简单的。为了解决该问题，Kempe 等人^[23] 提出了使用蒙特卡罗方法 (Monte Carlo method) 来对 $\mathbb{E}_{\mathcal{G}}[I(S)]$ 在一定精度内进行估计。蒙特卡罗方法的步骤如下。假设我们对图 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 中的所有边 $e \in \mathcal{E}$ 都进行抛硬币实验，图 \mathcal{G} 中边的连接的概率为 $p(e)$ ，我们以 $1 - p(e)$ 的概率移除掉边 e 。定义 g 为得到的结果图， $R_g(S)$ 为在图 g 中从种子节点集合 S 出发可达的节点集合。我们需要注意的是，图 g 中的边不再是概率性连接的边，而是确定性的边。对于任意节点 $v \in g$ ，如果在图 g 中存在

一条路径从节点集合 S 出发到达节点 v ，那么称节点 v 是从节点集合 S 可达的。Kempe 等人^[23] 证明了 $R_g(S)$ 的期望值与 $\mathbb{E}_{\mathcal{G}}[I(S)]$ 是相等的，即可表示为如下，

$$\mathbb{E}_{\mathcal{G}}[I(S)] = \mathbb{E}_{g \sim \mathcal{G}}[I_g(S)] \quad (4.3)$$

其中 $I_g(S) = |R_g(S)|$ ，即在图 g 中的影响力等于从种子节点集合出发可达的节点数目。因此，我们可以通过估计 $R_g(S)$ 的期望值来估计 $\mathbb{E}_{\mathcal{G}}[I(S)]$ ，即通过估计图 g 中可达的节点数目的期望来估计原图 \mathcal{G} 中的影响力期望值。在实际操作中，我们首先根据原来的社交网络生成多个实例 $g \sim \mathcal{G}$ ，然后对每一个实例进行计算其影响力 $I_g(S)$ ，最终计算其平均值作为 $\mathbb{E}_{\mathcal{G}}[I(S)]$ 的一个估计。假设我们在估计 $\mathbb{E}_{\mathcal{G}}[I(S)]$ 的过程中生成了 r 个实例图 g ，并且 r 足够大，那么在独立级联模型下，贪心算法能够得到一个 $(1 - 1/e - \varepsilon)$ 的近似最优解，其中 ε 是一个与图 \mathcal{G} 和 r 相关的常数^[26, 27]。一般来说，Kempe 等人建议设置 $r = 10,000$ ，许多其他的工作都采用了相似的设置参数。

尽管朴素的贪心算法是有效的，但是在复杂网络的应用中，算法的效率是极其低的。算法的时间复杂度为 $O(knmr)$ 。确切来说，算法进行了 k 次迭代来选择种子节点集合，每一次迭代需要对 $O(n)$ 个节点进行影响力的期望值的估计。每一次估计需要对生成的 r 个实例进行计算，而每一次计算需要消耗 $O(m)$ 的时间。因此，整个计算过程的时间复杂度为 $O(knmr)$ 。

对于具有子模性的函数，**惰性计算** (*lazy evaluations*) 技术是一种比较知名的优化方法，它能够大大的降低计算的次数，而不改变贪心算法的输出。这个技术首先由 Minoux^[28] 作为一种加速的贪心算法提出，Leskovec^[29] 等人通过实验验证了惰性计算针对影响力最大化问题能够加速近 700 倍。

即使惰性计算能够提高计算的性能，但是贪心算法仍然在效率上是不足的。究其原因，贪心算法的弊端主要是在于计算影响力期望值的过程中，它需要对 $O(kn)$ 个节点进行估计。然而，其中大多数估计都是无用的，因为我们只关心影响力期望值最大的节点。在朴素贪心算法的框架下，这些无用的计算又是不可避免的。

为了解决朴素贪心算法的这一弊端，Borgs 等人^[26] 提出了一种新的方法，突破了朴素贪心算法的限制。Tang 等人^[30] 将这种方法称之为**反向传播采样** (*Reverse Influence Sampling*)，并且阐述了其工作原理。为了解释反向传播采样算法的工作原理，我们首先引入如下的概念。

定义 4.1 (反向可达集合): 给定一个图 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ，我们对图中的 \mathcal{G} 中的每一条边 $e \in \mathcal{E}$ 进行抛硬币实验，依概率 $1 - p(e)$ 移除掉边 e 。定义 g 为得到的图，对于任意的节点 $v \in \mathcal{V}$ ，节点 v 在图 g 中的反向可达集合 $RR(v)$ 定义为图 g 中可达节

点 v 的节点集合。这就是说，如果节点 $u \in RR(v)$ ，则至少在图 g 中存在一条路径从节点 u 到达节点 v 。

根据定义4.1可知，如果节点 u 在节点 v 的反向可达集合 $RR(v)$ 中，那么节点 u 在图 \mathcal{G} 中能够依一定概率通过一条路径到达节点 v 。这也就表示，如果采用节点 u 作为种子节点集合 $S = \{u\}$ 在图 \mathcal{G} 中进行信息传播，那么节点 u 是有一定概率激活节点 v 的。此外，Borgs 等人给出了反向可达集合的性质如下，

引理 4.1: 如果一个节点 v 的反向可达集合 $RR(v)$ 有 ρ 的概率与节点集合 S 存在交集，那么如果以 S 为种子节点集合在图 \mathcal{G} 中进行信息传播，则节点 v 将依概率 ρ 被激活。

证明： 假定 g 为基于 \mathcal{G} 依概率 $1 - p(e)$ 移除每一条边 $e \in \mathcal{E}$ 生成的实例图。定义 ρ_2 为节点集合 S 在图 g 中可达节点 v 的概率， ρ_1 为图 g 中存在一条从节点集合 S 到达节点 v 的概率。那么，根据定义4.1可知， $\rho_1 = \rho_2$ 。 ■

基于以上的理论，反向传播采样方法的算法流程按照如下的两步进行。第一步，首先在图 \mathcal{G} 中等概率地任意选择一个节点 $v \in \mathcal{V}$ ，按照定义4.1来生成反向可达集合 $RR(v)$ 。然后，重复上述的过程来生成多个实例。第二步，选择 k 个节点来覆盖最多的反向可达集合。当且仅当一个节点 $u \in RR(v)$ 时，我们称节点 u 覆盖集合 $RR(v)$ 。最后，方法采用朴素贪心算法来得出一个下限为 $1 - 1/e$ 的近似最优解 S 作为结果返回。

反向传播采样方法的核心思想可以描述如下。如果一个节点 u 出现在许多的反向可达集合 $RR(v)$ 中，那么节点 u 就有更高的概率去激活更多的节点。而且，节点 v 是等概率地从节点集合 \mathcal{V} 中抽取，因此节点 u 将有概率激活图 \mathcal{G} 中更多的节点，即影响力 $I(\{u\})$ 的值会更大。这也就是说，如果一个种子节点集合 S 覆盖了最多的反向可达集合 $RR(v)$ ，那么 S 在图 \mathcal{G} 中将有最大的影响力期望值。

以图4.2中的社交网络 \mathcal{G} 为例，考虑在独立级联模型以及 $k = 1$ 的条件下，反向传播采样方法的工作流程。第一步，首先反向传播采样方法将等概率地随机从图 \mathcal{G} 中选取节点 v ，然后依照概率 $1 - p(e)$ 移除每一条边 e 得到图 g ，然后计算反向可达集合 $RR(v)$ 。假设我们选择了节点 v_3 并且得到的结果图 g 如图4.3所示，则我们可以计算得到 $RR_1 = \{v_1, v_2, v_3\}$ ，其中下标表示生成的反向可达集合的序号。这是因为在图 g 中， v_1 、 v_2 以及 v_3 是可达节点 v_3 的节点。然后，重复上述过程生成多个反向可达集合。假设这个过程中生成的其他反向可达集合为 $RR_2 = \{v_1\}$ ， $RR_3 = \{v_1, v_2\}$ ， $RR_4 = \{v_4\}$ ， $RR_5 = \{v_1, v_2, v_5\}$ 以及 $RR_6 = \{v_6\}$ 。在这个情况下，我们

可以得出节点 v_1 覆盖了最多的反向可达集合，因为节点 v_1 包含在集合 RR_1, RR_2, RR_3, RR_5 中。因此，反向传播采样方法返回 $S = \{v_1\}$ 作为最终结果。

与朴素贪心算法对比，反向传播采样算法之所以效率更高是因为避免了在计算 $O(kn)$ 次迭代的影响力期望值的无效计算。算法的核心关键点是以反向可达集合 RR 取代了对信息传播的迭代模拟。为了平衡反向传播采样方法的有效性和高效性，算法需要控制生成反向可达集合的数目。Borgs 等人证明了，为了在独立级联模型下得到一个 $(1 - 1/e - \varepsilon)$ 的近似最优解， RR 的数目至少为 $\Theta(k(m+n)\log n/\varepsilon^3)^{[26]}$ 。

4.2.2 传播效率最大化问题

传统的影响力最大化问题的目标是解决在给定种子节点集合大小 k 的条件下，计算得到使得影响力期望值最大的种子节点集合 S 。这个问题没有考虑信息的传播时延，给定不同的种子节点集合会使得网络中的节点在不同的时刻 t 被激活。在实际应用中，人们不仅关心传播影响范围的大小，也关注信息的传播效率。例如，在市场营销中，如何迅速地将一个新产品的信息传播给潜在用户是十分重要的。由此产生了一个新的问题，在一定条件下，我们如何能够计算出一个种子节点集合，使得网络的传播效率最大化。我们对传播效率定义如下。

定义 4.2 (传播效率): 假如存在一条路径从节点 u 到达节点 v 且路径中的每一个节点在信息传播结束时都是激活的，那么我们称这是一条从 u 到 v 的通路。网络中节点 u 到节点 v 之间可能存在多条通路。给定一个种子节点 u ，当信息传播过程结束时，对于图中的每一个节点 $v \in \mathcal{V}$ ，如果节点 u 到节点 v 之间不存在通路，则节点 u 到节点 v 的传播效率 $e_{u,v} = 0$ ，否则节点 u 到节点 v 的传播效率形式化如下，

$$e_{u,v} = \frac{1}{t_{u,v} + 1} \quad (4.4)$$

其中 $t_{u,v}$ 是从节点 u 到节点 v 的传播时延，即网络中节点 u 到节点 v 的最短通路的路径长度。对于节点 u 自身，传播时延 $t_{u,u} = 0$ 。给定种子节点集合 S ，传播效率函数 $T(S)$ 形式化如下，

$$T(S) = \sum_{v \in \mathcal{V}} \frac{1}{t_{S,v} + 1} \quad (4.5)$$

其中 $t_{S,v}$ 是从集合 S 到节点 v 的传播时延，即网络中从集合 S 到节点 v 的最短通路的路径长度。

以图4.2为例，考虑种子节点集合 $S = \{v_1\}$ 在图 \mathcal{G} 中的信息传播过程。假设信息传播过程的结果如图4.3所示。在 $t = 0$ 时刻，节点 v_1 作为种子节点被激活。下

一时刻, $t = 1$ 时节点 v_2 被节点 v_1 激活。然后, $t = 2$ 时刻, 节点 v_3 和节点 v_5 被激活。最后, 在 $t = 3$ 时刻, 没有新的节点可以被激活, 信息传播过程结束。因此, 传播效率 $T(S) = 1 + \frac{1}{2} + 2 \times \frac{1}{3} = 13/6$ 。

定义 $\mathbb{E}_{\mathcal{G}}[T(S)]$ 为种子节点集合 S 在图 \mathcal{G} 中的传播效率的期望值。那么传播效率最大化问题是在给定种子节点集合大小 k 的情况下, 计算得出使得传播效率期望值 $\mathbb{E}_{\mathcal{G}}[T(S)]$ 最大化的种子节点集合 S 。给定一个输入 k , 传播效率最大化问题可以形式化如下,

$$\begin{aligned} S^* &= \arg \max \mathbb{E}_{\mathcal{G}}[T(S)] \\ \text{s.t. } S &\subseteq \mathcal{V}, |S| = k \end{aligned} \quad (4.6)$$

以图4.2为例, 给定 $k = 1$ 以及图 \mathcal{G} , 考虑在独立级联模型下的传播效率最大化问题。我们能计算得出图 \mathcal{G} 中每一个节点的传播效率期望值。对于种子节点集合 $S = \{v_1\}$ 的情况, $\mathbb{E}_{\mathcal{G}}[T(\{v_1\})] = 1 \times 1 + 1 \times \frac{1}{2} + 4 \times 0.8 \times \frac{1}{3} \approx 2.57$ 。对于 $S = \{v_2\}$ 的情况, $\mathbb{E}_{\mathcal{G}}[T(\{v_2\})] = 1 \times 1 + 4 \times 0.8 \times \frac{1}{2} = 2.6$ 。对于其他的节点作为种子节点集合时, $\mathbb{E}_{\mathcal{G}}[T(\{v_3\})] = \mathbb{E}_{\mathcal{G}}[T(\{v_4\})] = \mathbb{E}_{\mathcal{G}}[T(\{v_5\})] = \mathbb{E}_{\mathcal{G}}[T(\{v_6\})] = 1$ 。因此, 在图 \mathcal{G} 中, 当 $k = 1$ 的情况下, 传播效率最大化问题的最优解为 $S^* = \{v_2\}$ 。与影响力最大化问题相比, 我们可以得知, 影响力期望值最大的种子节点集合不一定是传播效率期望值最大的集合。例如图4.2表示的社交网络中, 在 $k = 1$ 的情况下, 种子节点集合 $\{v_1\}$ 提供了最大的影响力期望值, 种子节点集合 $\{v_2\}$ 提供了最大的传播效率期望值。

为了解决传播效率最大化问题, 我们依旧可以运用蒙特卡罗方法来估计 $\mathbb{E}_{\mathcal{G}}[T(S)]$ 的值。假设图 g 是依概率 $1 - p(e)$ 移除图 \mathcal{G} 中的每一条边 $e \in \mathcal{E}$ 后得到的实例图。在独立级联模型下, 我们能够对传播效率的期望值估计如下,

$$\mathbb{E}_{\mathcal{G}}[T(S)] = \mathbb{E}_{g \sim \mathcal{G}}[T_g(S)] \quad (4.7)$$

其中 $T_g(S)$ 为种子节点集合 S 在图 g 中的传播效率。

在第4.2节中, 我们回顾了影响力最大化问题以及影响力函数的一些性质。影响力函数的单调性和子模性为问题的近似最优解算法提供了理论保证。基于该问题, 本节将传播时延考虑在内, 然后提出了传播效率最大化问题的定义。

4.3 方法描述

本节首先对传播效率最大化问题的复杂度, 其次证明了近似最优解算法的保证, 最后提出了**反向效率采样** (Reverse Efficiency Sampling) 算法来解决传播效率最大化问题。

众所周知，影响力最大化问题在独立级联模型下是一个 NP-hard 的问题。本节首先对传播效率最大化问题的复杂度进行分析。

定理 4.1: 传播效率最大化问题在独立级联模型下是一个 NP-hard 的问题。

证明： 考虑一个 NP-complete 的问题实例，集合覆盖 (Set Cover) 问题。问题的定义如下，给定一个集合 $U = \{u_1, u_2, \dots, u_n\}$ 的若干个子集 S_1, S_2, \dots, S_m ，我们需要求解是否存在 k 个子集的并集与集合 U 相等。下面我们证明集合覆盖问题可以看作是传播效率最大化问题的一个特例。

给定任意一个集合覆盖问题的实例，可以定义一个与之相对应的二部图 \mathcal{G} ，节点数目等于 $n + m$ 。图中的节点 i 对应于子集 S_i ，节点 j 对应于集合中的每一个元素 $u_j \in U$ 。对于每一个节点 $u_j \in S_i$ ，图中都有一条从节点 i 到节点 j 的边 (i, j) ，且边的传播概率为 $p_{i,j} = 1$ 。集合覆盖问题等同于判断在构造的二部图 \mathcal{G} 中，是否存在一个 k 个节点的集合 A 使得传播效率期望值 $\mathbb{E}_{\mathcal{G}}[T(A)] \geq k + \frac{n}{2}$ 。因为二部图 \mathcal{G} 中边的传播概率非 0 即 1，对应的信息传播是一个确定性的过程。初始化的 k 个种子节点等同于在集合覆盖问题中选择 k 个子集，而激活其余的 n 个节点相当于覆盖集合 U 。因此，如果存在 k 个节点的集合 A 满足 $\mathbb{E}_{\mathcal{G}}[T(A)] \geq k + \frac{n}{2}$ ，则集合覆盖问题能够被解决。

此外，另一个重要的问题是计算一个种子节点集合的传播效率期望值是非常耗时的，时间复杂度为 $O(mr)$ 。给定一个种子节点集合 S ，没有一个高效的方法直接计算传播效率期望值 $\mathbb{E}_{\mathcal{G}}[T(S)]$ 。本节将这个问题规约到 s - t 连通性问题，证明了在独立级联模型下计算传播效率期望值是一个 #P-hard 的问题。

定理 4.2: 在独立级联模型下，给定一个种子节点集合 S ，计算其传播效率期望值 $\mathbb{E}_{\mathcal{G}}[T(S)]$ 是一个 #P-hard 的问题。

证明： 我们通过将这个问题规约到 s - t 连通性问题来证明定理。给定一个有向图 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 以及两个节点 s 和 t ， s - t 连通性问题是来计算图 \mathcal{G} 中能够连通节点 s 和节点 t 的子图数。定义 $p_{s,t}^{\mathcal{G}}$ 为图 \mathcal{G} 中节点 s 与节点 t 连通的概率。当使得图 \mathcal{G} 的每一条边有 $1/2$ 的概率连通， $1/2$ 的概率不连通时，我们可以直观地观察到， s - t 连通性问题等同于求解 $p_{s,t}^{\mathcal{G}}$ 。下一步，我们将传播效率最大化问题归约到 s - t 连通性问题。首先，令种子节点集合 $S = \{s\}$ ，且图 \mathcal{G} 中的每一条边 $(u, v) \in \mathcal{E}$ 的传播概率 $p_{u,v}^{\mathcal{G}} = 1/2$ 。此时，图 \mathcal{G} 中，以 S 为种子节点集合的传播效率期望值定义为 $\mathbb{E}_{\mathcal{G}}[T_0(S)]$ 。下一步，我们在图 \mathcal{G} 增加一个节点 t_1 以及一条从节点 t 到节点 t_1

的边。我们称得到的新的图为 \mathcal{G}_1 ，以及新增的边的传播概率为 $p_{t,t_1}^{\mathcal{G}_1} = 1$ 。在图 \mathcal{G}_1 中，以 S 为种子节点集合的传播效率期望值定义为 $\mathbb{E}_{\mathcal{G}}[T_1(S)]$ 。而且我们可计算得知， $\mathbb{E}_{\mathcal{G}}[T_1(S)] = \mathbb{E}_{\mathcal{G}}[T_0(S)] + p_{t,t_1}^{\mathcal{G}_1} \cdot \sum_{i=1}^n p_{s,t,d=i}^{\mathcal{G}} \cdot \frac{1}{d+1} = \mathbb{E}_{\mathcal{G}}[T_0(S)] + \sum_{i=1}^n p_{s,t,d=i}^{\mathcal{G}} \cdot \frac{1}{d+1}$ ，其中 $p_{s,t,d=i}^{\mathcal{G}}$ 表示在图 \mathcal{G} 中，从节点 s 到节点 t 且距离为 d 的概率（从 s 达到 t 经过通过 d 次传播）， $n = |\mathcal{V}|$ 。下一步，我们继续在图 \mathcal{G}_1 中，添加节点 t_2 以及一条从 t_1 到 t_2 的边，得到新的图 \mathcal{G}_2 ，且令 $p_{t_1,t_2}^{\mathcal{G}_2} = 1$ 。在图 \mathcal{G}_2 中，以 S 为种子节点集合的传播效率期望值定义为 $\mathbb{E}_{\mathcal{G}}[T_2(S)]$ 。可以计算得知， $\mathbb{E}_{\mathcal{G}}[T_2(S)] = \mathbb{E}_{\mathcal{G}}[T_1(S)] + p_{t_1,t_2}^{\mathcal{G}_1} \cdot p_{t,t_1}^{\mathcal{G}_1} \cdot \sum_{i=1}^n p_{s,t,d=i}^{\mathcal{G}} \cdot \frac{1}{d+2} = \mathbb{E}_{\mathcal{G}}[T_1(S)] + \sum_{i=1}^n p_{s,t,d=i}^{\mathcal{G}} \cdot \frac{1}{d+2}$ 。我们可以重复上述步骤，得到 n 个以 S 为种子节点集合的传播效率期望值 $T_1(S), \dots, T_n(S)$ 。且递推公式为 $\mathbb{E}_{\mathcal{G}}[T_j(S)] = \mathbb{E}_{\mathcal{G}}[T_{j-1}(S)] + \sum_{i=1}^n p_{s,t,d=i}^{\mathcal{G}} \cdot \frac{1}{d+j}$ 。这些传播效率期望值方程可以表示如下，

$$\begin{bmatrix} \frac{1}{1+1} & \frac{1}{1+2} & \cdots & \frac{1}{1+n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{i+1} & \frac{1}{i+2} & \cdots & \frac{1}{i+n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{n+n} \end{bmatrix} \begin{bmatrix} p_{s,t,d=1}^{\mathcal{G}} \\ \vdots \\ p_{s,t,d=i}^{\mathcal{G}} \\ \vdots \\ p_{s,t,d=n}^{\mathcal{G}} \end{bmatrix} = \begin{bmatrix} \mathbb{E}_{\mathcal{G}}[T_1(S)] - \mathbb{E}_{\mathcal{G}}[T_0(S)] \\ \vdots \\ \mathbb{E}_{\mathcal{G}}[T_i(S)] - \mathbb{E}_{\mathcal{G}}[T_{i-1}(S)] \\ \vdots \\ \mathbb{E}_{\mathcal{G}}[T_n(S)] - \mathbb{E}_{\mathcal{G}}[T_{n-1}(S)] \end{bmatrix} \quad (4.8)$$

根据上述过程，可令 A 表示方程组 (4.8) 中的矩阵， $\mathbf{x} = (p_{s,t,d=1}^{\mathcal{G}}, p_{s,t,d=2}^{\mathcal{G}}, \dots, p_{s,t,d=n}^{\mathcal{G}})^T$ ， $\mathbf{b} = (\mathbb{E}_{\mathcal{G}}[T_1(S)] - \mathbb{E}_{\mathcal{G}}[T_0(S)], \dots, \mathbb{E}_{\mathcal{G}}[T_n(S)] - \mathbb{E}_{\mathcal{G}}[T_{n-1}(S)])^T$ 。方程组 (4.8) 可表示为 $A\mathbf{x} = \mathbf{b}$ 。根据引理 4.1 可知矩阵 A 是一个非奇异的矩阵，因此线性方程组 (4.8) 可以通过高斯-赛德尔方法在多项式时间内解决，而且构造这一个线性方程组是在线性时间内可完成的。我们可以直观地计算出图 \mathcal{G} 中节点 s 到 t 的连通概率 $p_{s,t}^{\mathcal{G}} = \sum_{i=1}^n p_{s,t,d=i}^{\mathcal{G}}$ ，这也就是说节点 s 到节点 t 的概率在线性时间内可以计算。因此， s - t 连通性问题是可解决的。而我们已知 s - t 连通性问题是一个 #P-complete 的问题。因此，计算传播效率期望值是一个 #P-hard 的问题。

定理 4.2 证明中的矩阵 A 如公式 (4.9) 所示，

$$A = \begin{bmatrix} \frac{1}{1+1} & \frac{1}{1+2} & \cdots & \frac{1}{1+n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{i+1} & \frac{1}{i+2} & \cdots & \frac{1}{i+n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{n+n} \end{bmatrix} \quad (4.9)$$

上述过程将计算传播效率期望值 $\mathbb{E}_{\mathcal{G}}[T(S)]$ 的问题规约到了 s - t 连通性问题, 证明了计算传播效率期望值 $\mathbb{E}_{\mathcal{G}}[T(S)]$ 的问题是一个 $\#P$ -hard 的问题。其中引用了矩阵 A 为非奇异矩阵的事实, 下面我们证明上述的矩阵 A 是非奇异的。首先, 我们可以观察到矩阵 A 是一个对称矩阵, 其次我们可以得到矩阵中通项 a_{ij} 。因此, 我们考虑使用行列变换对矩阵 A 的行列式进行降维, 发现行列式的递推公式, 计算行列式的值。如果行列式不为零, 则可证明矩阵 A 是非奇异的。引理以及其详细证明过程如下。

引理 4.1: 矩阵 $A \in \mathbb{R}^{n \times n}$ 且 $a_{ij} = \frac{1}{i+j}$, 则矩阵 A 是非奇异的。

证明: 令 Δ_n 表示矩阵 A 的行列式,

$$\Delta_n = |A| = \begin{vmatrix} \frac{1}{1+1} & \frac{1}{1+2} & \cdots & \frac{1}{1+n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{i+1} & \frac{1}{i+2} & \cdots & \frac{1}{i+n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{n+n} \end{vmatrix} \quad (4.10)$$

下一步, 令行列式 $|A|$ 中的第 i 行 ($2 \leq i \leq n$) 减去第 1 行, 可得到 $a_{ij} = \frac{1}{i+j} - \frac{1}{1+j} = \frac{1-i}{(i+j)(1+j)}$ 。则行列式 Δ_n 可表示为如下所示。

$$\begin{aligned} \Delta_n &= \begin{vmatrix} \frac{1}{1+1} & \frac{1}{1+2} & \cdots & \frac{1}{1+n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1-i}{(i+1)(1+1)} & \frac{1-i}{(i+2)(1+2)} & \cdots & \frac{1-i}{(i+n)(1+n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1-n}{(n+1)(1+1)} & \frac{1-n}{(n+2)(1+2)} & \cdots & \frac{1-n}{(n+n)(1+n)} \end{vmatrix} \\ &= \frac{\prod_{2 \leq i \leq n} (1-i)}{\prod_{1 \leq j \leq n} (1+j)} \begin{vmatrix} 1 & 1 & \cdots & 1 \\ \frac{1}{2+1} & \frac{1}{2+2} & \cdots & \frac{1}{2+n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{i+1} & \frac{1}{i+2} & \cdots & \frac{1}{i+n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{n+n} \end{vmatrix} \\ &= \frac{\prod_{2 \leq i \leq n} (1-i)}{\prod_{1 \leq j \leq n} (1+j)} |B| \end{aligned} \quad (4.11)$$

下一步,令行列式 $|B|$ 中的第 j 列($2 \leq j \leq n$)减去第1列,可得 $b_{ij} = \frac{1}{i+j} - \frac{1}{i+1} = \frac{1-j}{(i+j)(i+1)}$ 。则行列式 Δ_n 可表示为如下所示。

$$\begin{aligned}
 \Delta_n &= \frac{\prod_{2 \leq i \leq n} 1-i}{\prod_{1 \leq j \leq n} 1+j} \begin{vmatrix} 1 & 0 & \cdots & 0 \\ \frac{1}{2+1} & \frac{1-2}{(2+2)(2+1)} & \cdots & \frac{1-n}{(2+n)(2+1)} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{i+1} & \frac{1-2}{(i+2)(i+1)} & \cdots & \frac{1-n}{(i+n)(i+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1-2}{(n+2)(n+1)} & \cdots & \frac{1-n}{(n+n)(n+1)} \end{vmatrix} \\
 &= \frac{\prod_{2 \leq i \leq n} 1-i \prod_{2 \leq j \leq n} 1-j}{\prod_{1 \leq j \leq n} 1+j \prod_{2 \leq i \leq n} 1+i} \begin{vmatrix} 1 & 0 & \cdots & 0 \\ \frac{1}{2+1} & \frac{1}{2+2} & \cdots & \frac{1}{2+n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{i+1} & \frac{1}{i+2} & \cdots & \frac{1}{i+n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{n+n} \end{vmatrix} \\
 &= \frac{\prod_{2 \leq i \leq n} 1-i \prod_{2 \leq j \leq n} 1-j}{\prod_{1 \leq j \leq n} 1+j \prod_{2 \leq i \leq n} 1+i} \begin{vmatrix} \frac{1}{2+2} & \frac{1}{2+3} & \cdots & \frac{1}{2+n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{i+2} & \frac{1}{i+3} & \cdots & \frac{1}{i+n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+2} & \frac{1}{n+3} & \cdots & \frac{1}{n+n} \end{vmatrix} \\
 &= \lambda_n \Delta_{n-1}
 \end{aligned} \tag{4.12}$$

其中 λ_n 是非零的。因此,我们可以重复上述步骤,则行列式 Δ_n 可计算得到如下。

$$\begin{aligned}
 \Delta_n &= \prod_{k=1}^{n-1} \lambda_k \Delta_1 = \prod_{k=1}^{n-1} \frac{\prod_{k+1 \leq i \leq n} k-i \prod_{k+1 \leq j \leq n} k-j}{\prod_{k \leq j \leq n} k+j \prod_{k+1 \leq i \leq n} k+i} \Delta_1 \\
 &= \frac{1}{2n} \frac{\prod_{2 \leq k+1 \leq i \leq n} k-i \prod_{2 \leq k+1 \leq j \leq n} k-j}{\prod_{1 \leq k \leq j \leq n} k+j \prod_{2 \leq k+1 \leq i \leq n} k+i} \Delta_1
 \end{aligned} \tag{4.13}$$

由上述可知 Δ_n 中的每一项都不为零,所以可知 $\Delta_n \neq 0$,因此 A 是非奇异的。

由此可知,直接来求解传播效率最大化问题是非常困难的。针对这种情况, Nemhauser 等人^[24]证明了一个非负的,单调的且子模性的函数 $f(\cdot)$,朴素的贪心

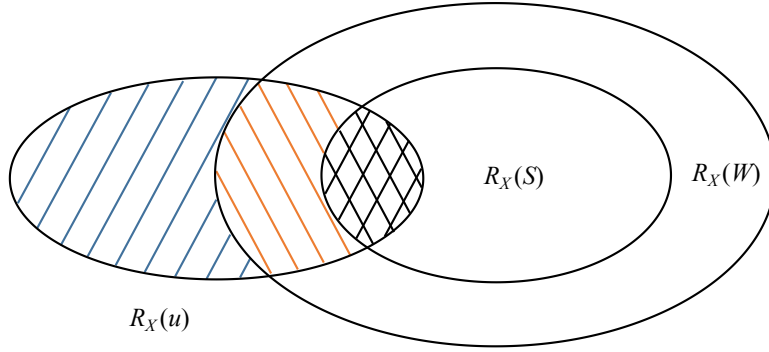
算法能够提供一个 $(1 - 1/e)$ 的近似最优解。在独立级联模型模型下的传播效率最大化问题的近似最优解的保证可以表示如下，

定理 4.3: 对于在独立级联模型下的一个网络 \mathcal{G} 的任意一个实例，传播效率函数的期望 $\mathbb{E}_{\mathcal{G}}[T(\cdot)]$ 是具有子模性的。

$\mathbb{E}_{\mathcal{G}}[T(\cdot)]$ 的单调性是显而易见的，因为向任意一个种子节点集合 S 中增加一个节点 u 不会使得传播效率减小。这也就是说，对于任意的种子节点集合 S 与节点 $u \in \mathcal{V}$ ， $\mathbb{E}_{\mathcal{G}}[T(S \cup \{u\})] \geq \mathbb{E}_{\mathcal{G}}[T(S)]$ 始终是成立的。为了证明定理4.3的结论，我们需要计算 $\mathbb{E}_{\mathcal{G}}[T(S \cup \{u\})] - \mathbb{E}_{\mathcal{G}}[T(S)]$ 的值，即我们需要计算将节点 u 加入到集合 S 中的边际收益。但是直接计算边际收益是十分困难的，因为整个信息传播过程是不确定的，节点被激活的顺序也是不确定的。为了解决这个难点，我们采用另一种等价的视角^[23] 来看待信息传播过程，它与节点激活的顺序无关。它提供了一种另外的途径来证明 $\mathbb{E}_{\mathcal{G}}[T(\cdot)]$ 的子模性。考虑如下情景，在独立级联模型下的信息传播过程中，当节点 u 在上一时刻被激活，尝试依概率 $p_{u,v}$ 去激活它的出度边指向的邻居节点 v 。我们可以对该过程用抛一枚偏置为 $p_{u,v}$ 的硬币的结果来模拟，我们直观上地可以知道抛硬币实验在当节点 u 被激活后进行还是在信息传播过程的一开始就进行，然后在节点 u 被激活后再揭晓结果，这个对于模拟是不影响的。因此，我们能在信息传播过程一开始，就对于图 \mathcal{G} 中的每一条边 $e = (u, v)$ 进行偏置为 $p_{u,v}$ 的抛硬币实验，然后在节点 u 被激活而节点 v 没有激活的时刻揭晓结果。

令所有的抛硬币的模拟在信息传播过程的一开始就完成，则信息传播过程可以看作如下的过程。每条边 $e = (u, v)$ 的抛硬币模拟，代表着节点 u 到节点 v 的尝试激活。如果激活成功，则我们称 e 是一条连通边。此外，我们可以称从节点 u 到节点 v 的路径是一条通路，如果路径中的每一条边都是连通边。基于上述的定义，我们可以很清晰地计算出在所有的抛硬币模拟都已确定的情况下，独立级联模型下种子节点集合为 S 的传播效率。信息传播过程结束时，当且仅当存在一条从 S 到节点 v 的通路时，节点 v 是激活的。我们可以根据从 S 到 v 的最短的通路，计算得到节点 v 的传播效率 $e_{S,v}$ 。考虑如下一个概率空间，空间中的每一个节点代表着图 \mathcal{G} 中所有边 e 对应的抛硬币模拟的结果。令 X 表示空间中的一个样本点，且定义 $T_X(S)$ 为以 S 为种子节点集合，抛硬币模拟结果为 X 的传播效率， $T_X(S)$ 可根据公式 (4.5) 计算得出。本节对定理4.3进行证明如下。

证明： 首先，我们证明对于概率空间中任意的确定的样本点 X ，传播效率函数 $T_X(\cdot)$ 是具有子模性的。为了证明上述的结论，令 S 和 W 为两个节点集合，且满足 $S \subseteq W \subseteq \mathcal{V}$ ，令 $u \in \mathcal{V}$ 为图 \mathcal{G} 中的任意一个节点。令 $\delta(u|S)$ 为添加节点 u 至集

图 4.4 $R_X(u)$, $R_X(S)$ 以及 $R_X(W)$ 的示意图

合 S 后传播效率的边际收益, $\delta(u|S) = T_X(S \cup \{u\}) - T_X(S)$ 。然后, 我们对 $\delta(u|S)$ 和 $\delta(u|W)$ 进行比较。为了方便比较, 令 $R_X(u)$ 表示节点 u 在样本点 X 下的可达节点集合。因为 $S \subseteq W$, 所以可得 $R_X(S) \subseteq R_X(W)$ 。当节点 u 加入到集合 S 和 W 中时, 会出现如图 4.4 三种情况。首先, 第一种情况如图中的斜线区域, 即对于节点 $v \in R_X(u) \setminus R_X(W)$ 。在此种情况下, 在加入节点 u 之前, 图中不存在从集合 S 或者集合 W 到达节点 v 的通路。因此, 可知第一种情况中 $\delta(u|S) = \delta(u|W)$ 。第二种情况是图中的反斜线区域, 即对于节点 $v \in (R_X(u) \cap R_X(W)) \setminus R_X(S)$ 。在此种情况下, 在加入节点 u 之前, 图中存在从集合 W 到节点 v 的通路, 但是不存在从集合 S 到节点 v 的通路。在此情况下, 可知 $\delta(u|S) \geq \delta(u|W)$ 。第三种情况是图中的网状区域, 即对于节点 $v \in R_X(u) \cap R_X(S)$ 。在此种情况下, 在节点 u 前, 从集合 S 或者集合 W 都有到节点 v 的通路。因为 $S \subseteq W$, $e_{W,v} \geq e_{S,v} > 0$, 所以在第三种情况下 $\delta(u|S) \geq \delta(u|W)$ 也是成立的。综合上述可知, 结论 $\delta(u|S) \geq \delta(u|W)$ 在任意情况下都是成立的, 即 $T_X(S \cup \{u\}) - T_X(S) \geq T_X(W \cup \{u\}) - T_X(W)$ 对于集合 $S \subseteq W$ 是成立的, 因此 $T_X(\cdot)$ 是具有子模性的。而又有 $\mathbb{E}_{\mathcal{G}}[T(S)] = \sum_X \Pr[X] \cdot T_X(S)$, 即传播效率期望值是概率空间中所有可能样本 X 的加权平均值。一个非负的线性组合不会改变函数的子模性, 故传播效率函数的期望 $\mathbb{E}_{\mathcal{G}}[T(\cdot)]$ 也是具有子模性。

根据定理 4.3 我们可知传播效率函数的期望值 $\mathbb{E}_{\mathcal{G}}[T(\cdot)]$ 是单调且子模性的, 因此朴素贪心算法 (*greedy*) 能够得到一个 $1 - 1/e$ 的近似最优解。朴素贪心算法的过程如算法 4.1 所示。算法的流程如下, 需要求解的种子节点集合 S 首先置为空集。其次, 遍历所有节点, 选择传播效率期望值边际收益最大的节点 u 加入到集合 S 中。然后, 重复第二步直到集合 S 的大小等于 k 。

算法 4.1 *greedy*(k, T)已知: k, \mathcal{G}, T 求: S

```

1: initialize  $S = \emptyset$ 
2: for  $i=1$  to  $k$  do
3:     select  $u = \arg \max_{w \in V \setminus S} (\mathbb{E}_{\mathcal{G}} [T(S \cup w)] - \mathbb{E}_{\mathcal{G}} [T(S)])$ 
4:      $S = S \cup u$ 
5: end for
6: return  $S$ 

```

此外，由于传播效率期望值具有子模性，因此惰性计算技术^[28]可以用来提升朴素贪心算法的性能。惰性计算能够极大地减少模拟评估的次数而不改变贪心算法的输出。正如惰性计算的名字，该方法的核心思想是尽可能地避免不必要的模拟评估。给定一个单调且子模性的函数 f ，令 $f(u|S)$ 为集合 S 加入节点 u 后的边际收益，即 $f(u|S) = f(S \cup \{u\}) - f(S)$ 。假设第 i 轮迭代贪心算法得出的种子节点集合为 S ，且我们对节点 $u \in \mathcal{V} \setminus S$ 的边际收益 $f(u|S)$ 进行了计算。在更早的迭代中，种子节点集合为 $S' \subset S$ ，且我们计算了节点 $w \in \mathcal{V} \setminus S'$ 的边际收益 $f(w|S')$ 。那么当 $f(w|S') \leq f(u|S)$ 时，根据子模性可知 $f(w|S) \leq f(w|S') \leq f(u|S)$ 。这就是说，在第 i 轮迭代中，节点 w 不可能成为最大化传播效率的节点，因此计算 $f(w|S)$ 是没有意义的。上述的思想可以通过一个优先队列来实现，如算法4.2所示，称之为惰性贪心 (*lazy greedy*) 算法。

算法的具体实现过程如下。对于每一个节点 u ，我们为之设计一种结构体，包含两个域值 $u.mg$ 以及 $u.i$ 。其中 $u.mg$ 表示最新一次迭代将节点 u 加入集合 S 时的传播效率期望值边际收益， $u.i$ 表示 $u.mg$ 更新时的迭代轮数。初始化时，我们对所有节点 u 计算 $T(u|\emptyset)$ 作为 $u.mg$ 加入优先队列 Q 中，以 $u.mg$ 作为队列的键，此时所有的 $u.i = 1$ 。此后的每一轮迭代，我们选择队列 Q 的队首（具有最大的边际收益）的节点 u 进行检查。如果 $u.i = i$ ，即节点 u 确实是第 i 迭代时加入到种子节点集合 S 中的边际收益最大的节点，则将节点 u 加入到种子节点集合 S 中。如果 $u.i \neq i$ ，则更新 $u.mg$ 为 $T(u|S)$ ，更新 $u.i$ 为当前的迭代轮数，然后将节点 u 插入回优先队列 Q 中。

惰性贪心算法极大地减少了无效模拟评估的次数。Leskovec 等人^[29]通过实验证明该算法对影响力最大化问题的速度提升了近 700 倍。对于传播效率最大化问题，由于传播效率函数的单调性和子模性，惰性贪心算法也能减少无效模拟评估次数，从而进行加速。

由定理4.2可知，直接计算传播效率期望值是很困难的。许多相关的工作^[29, 31, 32]在贪心算法的框架下对此类问题进行了改进。然而，这些改进算法仍然

算法 4.2 *LazyGreedy*(k, T)

已知: k, \mathcal{G}, T 求: S

```

1: initialize  $S = \emptyset$ , priority queue  $Q = \emptyset$ , iteration  $i = 1$ 
2: for  $j = 1$  to  $n$  do
3:    $u.mg = T(u|\emptyset)$ ,  $u.i = 1$ 
4:   put  $u$  into  $Q$  with  $u.mg$  as the key
5: end for
6: while  $i \leq k$  do
7:   pop the top element  $u$  of  $Q$ 
8:   if  $u.i = i$  then
9:      $S = S \cup \{u\}$ ,  $i = i + 1$ 
10:  else
11:     $u.mg = T(u|S)$ ,  $u.i = i$ 
12:  end if
13: end while
14: return  $S$ 

```

不够高效。另一方面，启发式的算法能够解决效率问题，但是不能提供理论上的性能保证。本节借鉴了 Borgs 等人^[26]的反向传播采样思想来解决传播效率最大化问题。

算法 4.3 *RES*($k, r, T'(\cdot)$)

已知: k, r, T' 求: S

```

1: initialize  $\mathcal{H} = (\mathcal{V}, \emptyset)$ 
2: for  $i = 1$  to  $r$  do
3:   stochastically select a node  $v \in \mathcal{V}$  uniformly
4:   generate  $z_i = RR(v)$ 
5:   calculate  $e_{u,v} = \frac{1}{t_{u,v} + 1}$  for  $u \in RR(v)$ 
6:   add  $z_i$  to the edge set  $\mathcal{Z}$  of  $\mathcal{H}$ 
7: end for
8: initialize  $S = \emptyset$ 
9: for  $j = 1$  to  $k$  do
10:   $w = \arg \max_{v \in \mathcal{V}} (T'(S \cup v) - T'(S))$ 
11:  add  $w$  to  $S$ 
12:  remove  $w$  from  $\mathcal{V}$ 
13: end for
14: return  $S$ 

```

如算法4.3所示，我们称之为**反向效率采样** (*Reverse Efficiency Sampling*)，算法的流程可描述如下。反向效率采样算法按照两步进行。第一步是根据图 \mathcal{G} 建立超图 $\mathcal{H} = (\mathcal{V}, \mathcal{Z})$ ，其中 \mathcal{V} 是与图 \mathcal{G} 相同的节点集合， \mathcal{Z} 是超边集合。每一条超边 $z_i \in \mathcal{Z}$ 代表一个反向可达集合 $RR(v)$ ，其中节点 $v \in \mathcal{V}$ ，是随机等概率选取的。超边 z_i 可以形式化为一个集合 $\{u_1, u_2, \dots, u_j\}$ ，代表着存在一条从节点 $u \in z_i$ 到节点 v 的通路。我们不断地随机等概率选取节点 v ，然后基于图 \mathcal{G} 生成图 g ，计算反向可达集合 $RR(v)$ ，即超边 z_i ，然后添加到超边集合 \mathcal{Z} 中。该过程通过模拟图 \mathcal{G} 的反向图中的信息传播过程来实现。在反向图中首先激活节点 v ，然后按照宽度优先搜索来概率性地激活出度边指向的邻居节点。当信息传播过程停止时，被激活的节点集合构成了超图 \mathcal{H} 中的一条边。同时，我们还需要记录超边上每一个节点 $u \in z_i$ 的传播效率 $e_{u,v} = \frac{1}{t_{u,v}+1}$ 。我们重复生成超边的过程来构建超边集合 \mathcal{Z} 直到集合的大小到达预定的值 r 。

在第二步中，我们基于超图 \mathcal{H} 来计算种子节点集合 S 。在这一步中，算法重复地选择当前传播效率边际收益最大的节点 $w \in \mathcal{V}$ 加入到集合 S 中，其中边际效益为 $T'(S \cup w) - T'(S)$ ，且 $T'(S) = \sum_{i=1}^r \frac{1}{t_{S,v_i}+1}$ ， v_i 为第 i 轮迭代中随机等概率选取的节点，对应的超边为 z_i 。最终，算法得到一个大小为 k 的节点集合 S 。

下一步，我们对算法4.3进行详细地分析。首先，我们可以观察到种子节点集合 S 的传播效率期望值等于 n 倍从集合 S 到节点 u 的传播效率 $e_{S,u}$ ，其中节点 u 从图 g 中随机等概率选取的，图 g 是基于图 \mathcal{G} 生成的实例图。上述现象可表述为如下，

定理 4.4: $\mathbb{E}_{g \sim \mathcal{G}} [T_g(S)] = n \Pr_{v, g \sim \mathcal{G}} [S \cap RR(v) \neq \emptyset] \cdot \frac{1}{d_{S,v}+1}$

证明：

$$\begin{aligned}
 \mathbb{E}_{g \sim \mathcal{G}} [T_g(S)] &= \sum_{v \in \mathcal{V}} \Pr_{g \sim \mathcal{G}} [\exists u \in S, v \in R_g(u)] \cdot \frac{1}{d_{S,v}+1} \\
 &= \sum_{v \in \mathcal{V}} \Pr_{g \sim \mathcal{G}} [\exists u \in S, u \in RR_g(v)] \cdot \frac{1}{d_{S,v}+1} \\
 &= n \Pr_{v, g \sim \mathcal{G}} [\exists u \in S, u \in RR_g(v)] \cdot \frac{1}{d_{S,v}+1} \\
 &= n \Pr_{v, g \sim \mathcal{G}} [S \cap RR_g(v) \neq \emptyset] \cdot \frac{1}{d_{S,v}+1}
 \end{aligned}$$

定理4.4表示我们可以通过观察事件 $S \cap RR_g(v) \neq \emptyset$ 的概率来计算 $\mathbb{E}_{\mathcal{G}} [T(S)]$ 。超图 \mathcal{H} 中节点 $u \in \mathcal{V}$ 的度即为信息传播模拟过程中成功激活的节点次数。因此，

由定理4.4可知，我们能够基于超图 \mathcal{H} 来计算传播效率期望值。给定种子节点集合 S 与超图 \mathcal{H} 下的传播效率函数如下所示，

$$T'(S) = \sum_{z_i \in \mathcal{Z}} \frac{1}{t_{S, v_i} + 1} \quad (4.14)$$

其中 S 为种子节点集合， z_i 为随机等概率选取的节点 v_i 对应的超边。下一步，我们分析在超图 \mathcal{H} 中的传播效率函数期望值 $\mathbb{E}_{\mathcal{H}}[T'(\cdot)]$ 。我们给出定理如下，

定理 4.5: 对于在独立级联模型下的一个网络 \mathcal{G} 的任意一个实例，对应生成的超图 \mathcal{H} 中传播效率函数的期望 $\mathbb{E}_{\mathcal{H}}[T'(\cdot)]$ 是具有子模性的。

证明： 易证 $\mathbb{E}_{\mathcal{H}}[T'(\cdot)]$ 是单调性的，因为将任意节点 u 加入到任意种子节点集合 S 中都不会减少传播效率期望值。为了证明子模性，我们可以先证明 $T'(\cdot)$ 的子模性。假设种子节点集合 S 与 W 满足 $S \subseteq W \subseteq \mathcal{V}$ ，节点 $u \in \mathcal{V}$ 是超图 \mathcal{H} 中的节点。如果集合 $S \cap z \neq \emptyset$ ，则我们称集合 S 覆盖超边 z 。与定理4.3相同，当将节点 u 加入集合 S 与 W 时，将有三种情况。第一种情况是集合 S 与 W 都不覆盖超边 z ，则 $T'(S \cup u) - T'(S) = T'(W \cup u) - T'(W)$ ，即二者的边际效益相等。第二种情况是集合 S 不覆盖超边 z ，集合 W 覆盖超边 z 。此种情况下 $e_{S,v} = 0$ ，而 $e_{W,v} > 0$ ，故 $T'(S \cup u) - T'(S) \geq T'(W \cup u) - T'(W)$ 。第三种情况是集合 S 与 W 都覆盖超边 z 。此种情况下由于 $S \subseteq W$ ，所以 $e_{S,v} \leq e_{W,v}$ ，故 $T'(S \cup u) - T'(S) \geq T'(W \cup u) - T'(W)$ 。所以所有情况，我们都能得出结论 $T'(S \cup u) - T'(S) \geq T'(W \cup u) - T'(W)$ 在独立级联模型下， $S \subseteq W$ 的情况下是成立的，因此 $T'(\cdot)$ 是子模性的。而期望是对 $T'(\cdot)$ 的加权线性求和，故 $\mathbb{E}_{\mathcal{H}}[T'(\cdot)]$ 也是子模性的。

由于传播效率函数期望值 $\mathbb{E}_{\mathcal{H}}[T'(\cdot)]$ 是单调且子模的，则算法4.3中的反向效率采样算法能够得到一个 $(1 - 1/e - \varepsilon)$ 的近似最优解来解决独立级联模型下的传播效率最大化问题。其中 ε 与采样的次数 r 以及图 \mathcal{G} 的结构有关。如何计算 ε 与 r 以及 \mathcal{G} 的关系仍然是一个开放性的问题。下面本节对上述提及的算法的时间复杂度进行分析。其中朴素贪心算法和惰性贪心算法都是基于贪心算法框架下实现的。这一类算法进行 k 轮迭代来选择种子节点集合 S ，每一轮迭代需要对 $O(n)$ 个节点进行传播效率期望值的估计。而每一次传播效率期望值的估计需要在 r 个生成的实例图 g 上进行 $O(m)$ 次计算。因此，朴素贪心算法和惰性贪心算法的时间复杂度都是 $O(knmr)$ 的。但是由于惰性贪心算法基于惰性计算避免了一些不必要的模拟评估，惰性贪心算法在实际运行中比朴素贪心算法的效率要高。相比之下，反向效率采样算法是在另一个框架下设计的。首先，反向传播采样算法生成了 r 条超边，构造了超图 \mathcal{H} ，每生成一条超边需要消耗 $O(m)$ 的时间，因此生成整个超

图 \mathcal{H} 需要消耗 $O(rm)$ 的时间。然后, 算法迭代 k 轮来生成种子节点集合 S , 每一轮迭代需要对 $O(n)$ 个节点进行模拟估计, 每一次模拟估计需要对 r 条超边进行运算, 因此计算种子节点集合 S 的时间复杂度为 $O(knr)$ 。有上述分析可知, 反向效率采样的时间复杂度相比于朴素贪心算法和惰性贪心算法要低。

4.4 实验分析

本节设计了若干组实验来验证上述提出的算法。首先, 我们介绍实验的设置。其次, 我们详细地展示了实验结果, 并对其进行分析。本节中的实验基于八核 (Intel Xeon 1.80GHz CPU)、64GB 内存的机器运行, 操作系统为 64 位 CentOS release 6.7。以上所有算法都是基于 Java 实现, 在 JDK 1.8.0_40 环境下编译。

4.4.1 实验设置

本小节介绍实验的设置, 包括数据集、传播模型以及算法等。

表 4.2 数据集特征

数据集	节点数	边数	类型	平均度
<i>Facebook</i>	4k	88k	无向图	43.7
<i>HepPh</i>	35k	422k	有向图	24.4
<i>Twitter</i>	81k	1.8M	有向图	43.5
<i>DBLP</i>	655k	2M	无向图	6.1

数据集, 表4.2描述了实验所用的数据集, 包括脸书 (*Facebook*)、高能物理论文引用网络 (*HepPh*)、推特 (*Twitter*)、计算机科学合作网络 (*DBLP*)。这些网络都是相关研究中使用频繁的标注数据集, 可以从 SNAP^[33] 网站上下载得到。实验数据集中的网络有着不同大小的网络和特性, 因此能够比较好地验证本章提出的算法。实验数据集包括两个无向图以及两个有向图。其中 *Facebook* 与 *Twitter* 是社交网络, 如果在 *Facebook* 网络中用户 u 与用户 v 是好友关系或者在 *Twitter* 网络中用户 u 关注了用户 v , 则在对应的图中存在一条从节点 u 到节点 v 的边 $e = (u, v)$ 。网络 *HepPh* 为 arXiv 电子出版高能物理论文的引用网络。如果论文 u 引用了论文 v , 则存在一条从节点 u 到节点 v 的边。而在实验中, 我们将 *HepPh* 网络的边调换了顺序来表示论文 v 影响了论文 u , 如果论文 u 引用了论文 v 。网络 *DBLP* 提供了一个丰富的计算机科学论文领域的合作网络。如果两位作者曾经合作过一篇文章, 则图中存在一条连接两位作者的边。以上的四个网络是具有代表性意义的网络, 包含了丰富的社交关系, 网络的边集合的大小从万级别到百万级别不等。

传播模型，在实验中，我们考虑了如下两种传播模型，即**均匀独立级联模型** (*Uniform Independent Cascade*) 和**加权独立级联模型** (*Weighted Independent Cascade*)。其中均匀独立级联模型中的节点 u 到节点 v 传播概率 $p_{u,v}$ 设置为一个常数。在本章的实验中，根据相关的研究工作^[32]，对所有网络，我们设置经验参数 $p_{u,v} = 0.01$ 。而对于加权独立级联模型，传播概率 $p_{u,v}$ 被设置为 $\frac{1}{d_v}$ ，其中 d_v 表示节点 v 的入度。该模型满足 $\sum_{u \in \mathcal{V}} p_{u,v} = 1$ ，因此被先前的相关工作广泛使用^[23, 30, 32, 34]。

算法，本节中将提出的反向效率采样算法与其他的三种方法进行了比较，反向传播采样算法、惰性贪心算法以及**最大度算法** (*DegreeGreedy*)。首先，我们将反向效率采样算法与反向传播采样算法进行对比来展示传播效率最大化问题与影响力最大化问题的区别。然后，我们进行了实验来验证算法的正确性以及效率。*CELF* 算法是在不打破近似最优解理论保证下来解决影响力最大化问题的一个加速算法，我们借鉴 *CELF* 算法的核心思想实现了惰性贪心算法来解决传播效率最大化问题。最大度算法是一个启发式的算法，在每一轮迭代中选择出度最大的节点做为种子节点。上述的反向传播采样算法、惰性贪心算法、最大度算法和我们提出的反向效率采样算法都是基于 Java 实现的。

参数设置，在比较传播效率以及运行时间时，种子节点集合 S 的大小 k 设置为 1, 5, 10, \dots ，直到 50。对于惰性贪心算法，我们根据之前相关的工作，经验性地设置蒙特卡罗方法的重复模拟次数 $r = 10,000$ 。对于反向效率采样算法，我们设置 $r = n \log n$ 来得到一个 $(1 - 1/e - \varepsilon)$ 的近似最优解，其中 ε 与 r 以及网络相关。在所有实验中，我们都重复算法 5 次，记录其平均值作为实验结果。

4.4.2 相似度对比

第一组实验将反向效率采样算法与反向传播采样算法进行对比，来说明传播效率最大化问题与影响力最大化问题的区别。令反向效率采样算法求解最大化传播效率期望值得到的种子节点集合为 S ，反向传播采样算法求解最大化影响力期望值得到的种子节点集合为 S' 。我们可得到不同 k 下，集合 S 与集合 S' 之间的 **Jaccard 相似度**。令 $J(S, S')$ 表示集合 S 与集合 S' 之间的 Jaccard 相似度，可形式化如下，

$$J(S, S') = \frac{|S \cap S'|}{|S \cup S'|} \quad (4.15)$$

其中 $|S \cap S'|$ 表示集合 S 与集合 S' 的交集大小， $|S \cup S'|$ 表示集合 S 与集合 S' 的并集大小。实验结果如图4.5至图4.8所示。从实验结果可知，在 k 较小的时候，例如 $k = 1$ 时，Jaccard 相似度是不稳定的。在 $k = 1$ 的情况下，*Facebook*, *HepPh*

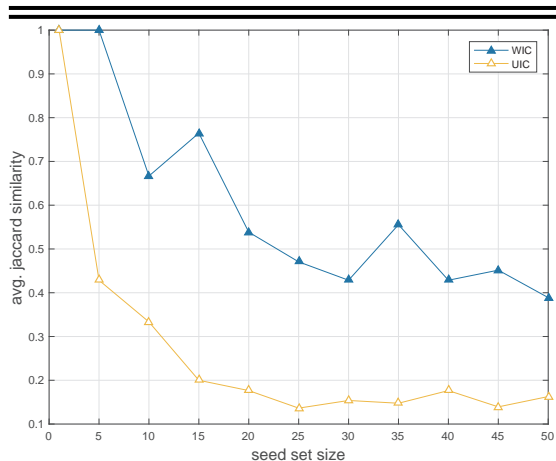


图 4.5 在 *Facebook* 数据集上不同 k 下的 Jaccard 相似度对比

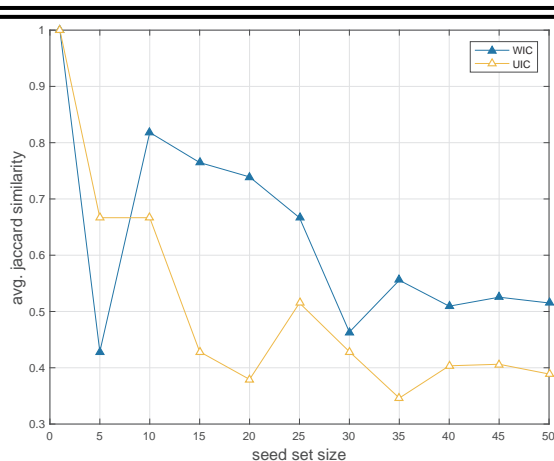


图 4.6 在 *HepPh* 数据集上不同 k 下的 Jaccard 相似度对比

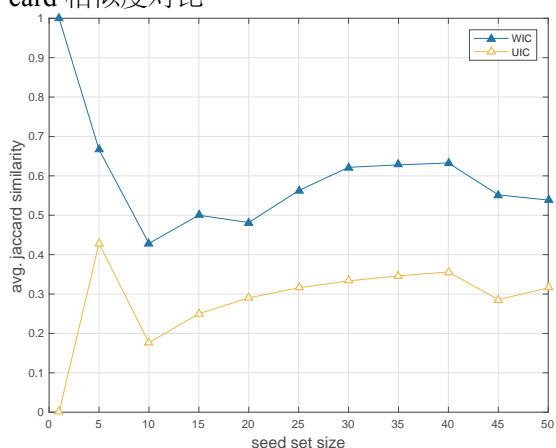


图 4.7 在 *Twitter* 数据集上不同 k 下的 Jaccard 相似度对比

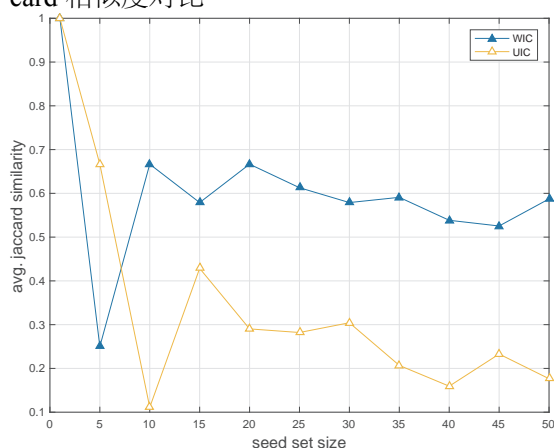


图 4.8 在 *DBLP* 数据集上不同 k 下的 Jaccard 相似度对比

and *DBLP* 网络中的 $S = S'$, 而 *Twitter* 网络中的 $S \neq S'$ 。由此可知, 当 k 比较小的时候, 使得网络有最大影响力期望值的节点也可能是使得网络有最大传播效率期望值的节点。当 k 比较小的时候, 大部分节点处于未激活状态, 能够激活尽可能多的节点也许就能获得更大的传播效率。随着 k 的增加, 例如超过 10 以后, 在均匀独立级联模型和加权独立级联模型下的 $J(S, S')$ 值变得相对稳定。在均匀独立级联模型中 $J(S, S')$ 约等于 0.2, 在加权独立级联模型中 $J(S, S')$ 约等于 0.5。因此, 我们可知传播效率最大化问题的种子节点集合与影响力最大化问题的种子节点集合并不是相同的, 即传播效率最大化问题与影响力最大化问题是两个不同的问题。

4.4.3 算法精确度对比

在本小节中, 我们计算各种算法得出种子节点集合的传播效率, 对比各种算法的精确度。对于结果种子节点集合, 我们运行 10,000 次蒙特卡罗模拟来估计信息传播过程中的传播效率。此外, 对于每一个数据集, 我们调整种子节点集合大

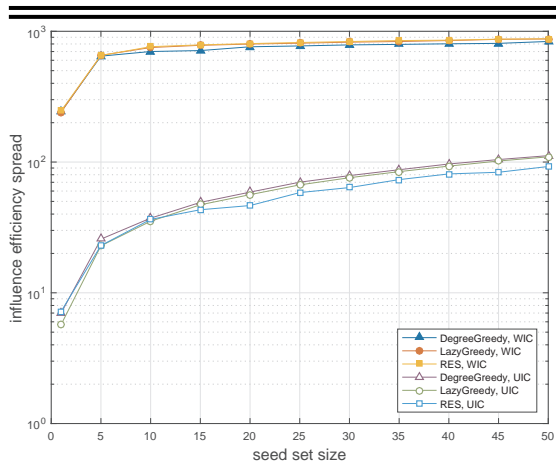


图 4.9 在 *Facebook* 数据集上不同 k 下的传播效率对比

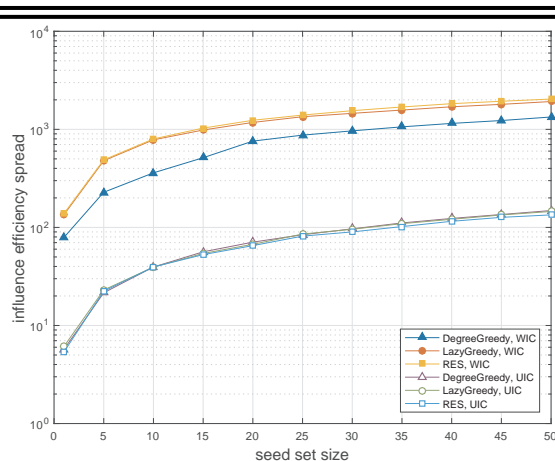


图 4.10 在 *HepPh* 数据集上不同 k 下的传播效率对比

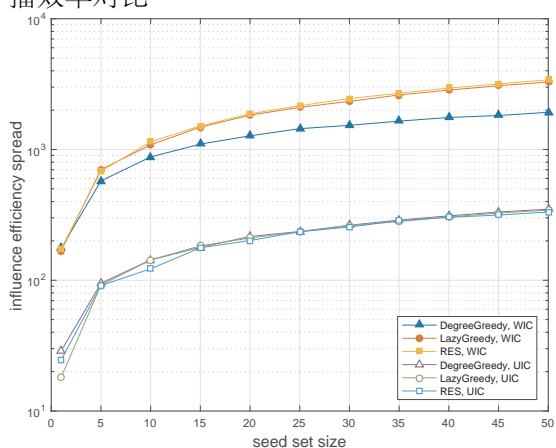


图 4.11 在 *Twitter* 数据集上不同 k 下的传播效率对比

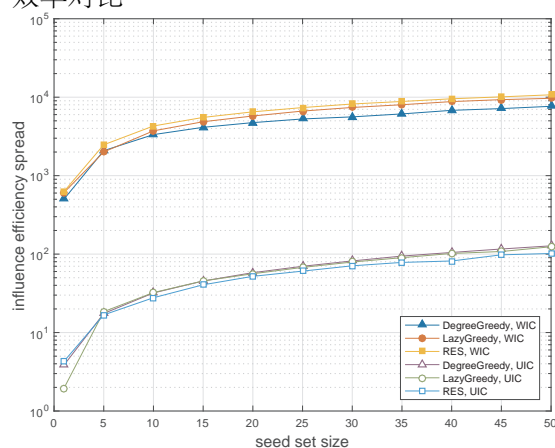


图 4.12 在 *DBLP* 数据集上不同 k 下的传播效率对比

小 k 从 1 到 50 来观察不同大小种子节点集合的传播效率。我们在均匀独立级联模型以及加权独立级联模型下进行实验，实验结果如图 4.9 至图 4.12 所示。如图所示，反向效率采样算法与惰性贪心算法、最大度算法在均匀独立级联模型下各数据集集中的传播效率性能相似。这是由于在均匀独立级联模型下不能保证 $\sum_{u \in \mathcal{V}} p_{u,v} = 1$ 的成立，且反向效率采样算法得到的近似最优解的近似率不高，所以在某些网络中（例如图 4.9），反向效率采样算法的性能要稍差一些。但是我们可以得知，反向效率采样算法的精确度在最差的情况下（*Facebook* 网络）与最好结果相差少于 17%，不会相差一个数量级。在加权独立级联模型下，反向效率采样算法在各数据集集中的精确度都是最高的，并且与惰性贪心算法的精确度相似，而最大度算法的精确度相比较低。例如在图 4.11 中 $k = 50$ 的情况下，最大度算法与反向效率采样算法的精确度相差 56.5%，即最大度算法得出种子节点集合的传播效率只有反向效率采样算法的一半左右。

由上述在均匀独立级联模型以及加权独立级联模型下各数据集上的实验结果可知，反向效率采样算法和惰性贪心算法都有理论上的精确度保证，性能优于启发式算法，例如最大度算法。

4.4.4 运行时间对比

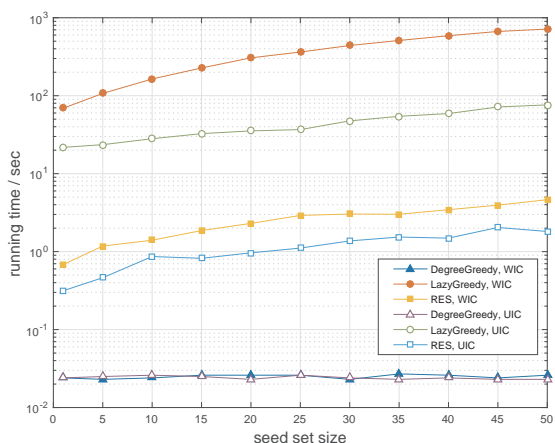


图 4.13 在 *Facebook* 数据集上不同 k 下的运行时间对比

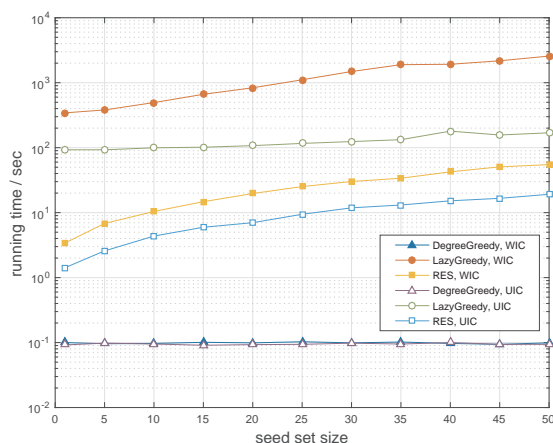


图 4.14 在 *HepPh* 数据集上不同 k 下的运行时间对比

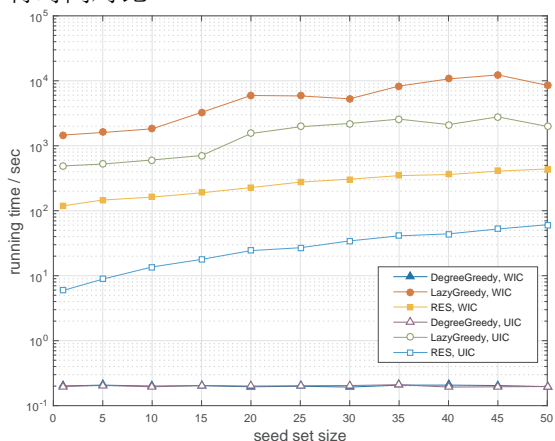


图 4.15 在 *Twitter* 数据集上不同 k 下的运行时间对比

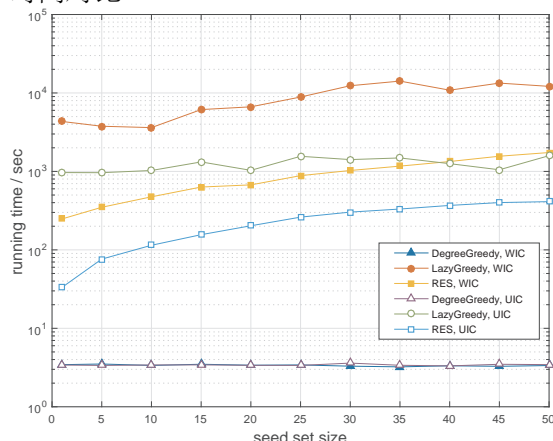


图 4.16 在 *DBLP* 数据集上不同 k 下的运行时间对比

本小节我们对各算法的运行时间进行对比实验。为了验证反向效率采样算法的效率，我们在均匀独立级联模型以及加权独立级联模型下各数据集上不同 k 的情况下，与惰性贪心算法以及最大度算法进行实验对比，结果如图4.13至图4.16。如图所示，在均匀独立级联模型以及加权独立级联模型下，反向效率采样算法相比惰性贪心算法都能提高一个数量级的效率。而最大度算法是一种启发式的算法，因此运行时间最少，且不随 k 的变化而变化，但是如第4.4.3小节所示，该算法不能提供性能的保证。因为最大度算法算法简单，本小节利用该算法来作运行时间

的一个下限基准。从图中我们可以得出，反向效率采样算法和惰性贪心算法的运行时间都随着 k 的增加而增加，其中惰性贪心算法运行时间的增长速率高于反向效率采样算法。例如，在图4.15中在 $k = 1$ 时，惰性贪心算法的运行时间约为反向效率采样的 10 倍，在 $k = 50$ 时，增长为约 20 倍。因此，反向效率采样算法的效率优于其他的算法。

4.4.5 可扩展性

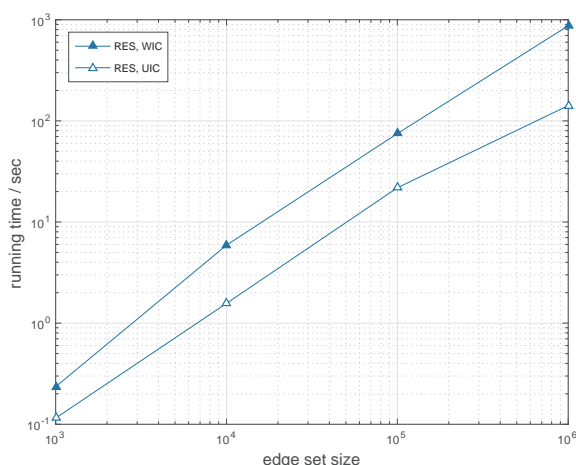


图 4.17 反向效率采样算法的可扩展性

本小节进行了反向效率采样算法的可扩展性实验。实验使用 *DBLP* 网络来生成各种大小的子图。生成过程如下所示。首先，我们随机地选取图中的一个节点，然后做宽度优先搜索来添加边到子图中，该过程直到子图的大小满足要求时停止。如果宽度优先搜索过程在子图大小没有满足前就停止了，那么在重新随机地选择一个不在目前子图中的节点，继续进行宽度优先搜索，重复上述过程直到满足要求。利用该方法，我们可以得到各种大小的子图。本小节的实验中，我们令子图的边的大小从 $m = 1,000$ 到 $m = 1,000,000$ 变化，然后运行反向效率采样算法，记录运行的时间。在均匀独立级联模型以及加权独立级联模型下的实验结果如图4.17所示。由图可知，随着图大小的增长，运行时间的增长是线性的。因此，反向效率采样算法是可扩展的。

4.5 本章小结

为了更好地理解网络中的传播效率，本章将信息传播过程中的传播时延考虑在内，提出了一个新的问题，传播效率最大化问题。该问题的目的在于最大化网络中的传播效率期望值。然后，我们证明了在独立级联模型下，传播效率最大化问题是一个 NP-hard 的问题，而且计算传播效率的过程是一个 #P-hard 的问题。为

了解决该问题，我们证明了在独立级联模型下，传播效率函数的期望值是单调且子模性的。基于上述的性质，我们设计了三种算法来解决该问题。最后，我们在真实的数据集上进行了一系列的实验来验证所提出的算法。实验结果表明了传播效率最大化问题与影响力最大化问题的不同，证明了提出的算法的正确性和有效性。本章所研究的内容对充分理解信息传播过程有着一定的帮助。

第五章 基于概率阅读模型的影响力量化研究

5.1 研究动机

5.2 相关定义

5.3 方法描述

5.4 实验分析

5.5 本章小结

第六章 总结与工作

6.1 本文工作总结

6.2 未来工作展望

致 谢

衷心感谢导师 xxx 教授和 xxx 副教授对本人的精心指导。他们的言传身教将使我终生受益。

感谢 NudtPaper，它的存在让我的论文写作轻松自在了许多，让我的论文格式规整漂亮了许多。

参考文献

- [1] Lin J, Efron M, Wang Y, et al. Overview of the TREC-2015 Microblog Track [C]. In Proceedings of the 24th Text REtrieval Conference (TREC 2015). 2015.
- [2] Ounis I, Macdonald C, Lin J, et al. Overview of the TREC-2011 Microblog Track [C]. In Proceedings of the 20th Text REtrieval Conference (TREC 2011). 2011.
- [3] Soboroff I, Ounis I, Macdonald C, et al. Overview of the TREC-2012 Microblog Track. [C]. In Proceedings of the 21th Text REtrieval Conference (TREC 2012). 2012.
- [4] Lin J, Efron M. Overview of the TREC-2013 Microblog Track [C]. In Proceedings of the 22th Text REtrieval Conference (TREC 2013). 2013.
- [5] Lin J, Efron M, Wang Y, et al. Overview of the TREC-2014 Microblog Track [C]. In Proceedings of the 23th Text REtrieval Conference (TREC 2014). 2014.
- [6] Xie H, Li X, Wang T, et al. Personalized Search for Social Media via Dominating Verbal Context [J]. Neurocomputing. 2015, 172 (C): 27–37.
- [7] Sontag D, Collins-Thompson K, Bennett P N, et al. Probabilistic models for personalizing web search [C]. In Proceedings of the fifth ACM international conference on Web search and data mining. 2012: 433–442.
- [8] Wang H, He X, Chang M-W, et al. Personalized ranking model adaptation for web search [C]. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. 2013: 323–332.
- [9] Tang L, Jiang Y, Li L, et al. Personalized recommendation via parameter-free contextual bandits [C]. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2015: 323–332.
- [10] Liang C, Leng Y. Collaborative filtering based on information-theoretic co-clustering [J]. International Journal of Systems Science. 2014, 45 (3): 589–597.
- [11] Vosecky J, Leung K W-T, Ng W. Collaborative personalized twitter search with topic-language models [C]. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval. 2014: 53–62.
- [12] Xue G-R, Han J, Yu Y, et al. User language model for collaborative personalized search [J]. ACM Transactions on Information Systems (TOIS). 2009, 27 (2): 11.

-
-
- [13] Yang X, Guo Y, Liu Y, et al. A survey of collaborative filtering based social recommender systems [J]. *Computer Communications*. 2014, 41: 1–10.
 - [14] Canuto S, Gonçalves M, Santos W, et al. An Efficient and Scalable MetaFeature-based Document Classification Approach based on Massively Parallel Computing [C]. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2015: 333–342.
 - [15] Severyn A, Moschitti A. Learning to rank short text pairs with convolutional deep neural networks [C]. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2015: 373–382.
 - [16] Ren Z, Peetz M-H, Liang S, et al. Hierarchical multi-label classification of social text streams [C]. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 2014: 213–222.
 - [17] Paik J H. A novel TF-IDF weighting scheme for effective ranking [C]. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 2013: 343–352.
 - [18] Gao J, Xu G, Xu J. Query expansion using path-constrained random walks [C]. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 2013: 563–572.
 - [19] Letelier A, Pérez J, Pichler R, et al. Static analysis and optimization of semantic web queries [J]. In: *Proc. of the 31st Symposium on Principles of Database Systems (PODS)*. 2012, 38 (4): 84–87.
 - [20] Si J, Li Q, Qian T, et al. Users' interest grouping from online reviews based on topic frequency and order [J]. *World Wide Web*. 2014, 17 (6): 1321–1342.
 - [21] Wang Y, Sherman G, Lin J, et al. Assessor Differences and User Preferences in Tweet Timeline Generation [C]. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2015: 615–624.
 - [22] Domingos P, Richardson M. Mining the network value of customers [C]. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 2001: 57–66.
 - [23] Kempe D, Kleinberg J, Tardos É. Maximizing the spread of influence through a social network [C]. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003: 137–146.

-
-
- [24] Nemhauser G L, Wolsey L A, Fisher M L. An analysis of approximations for maximizing submodular set functions—I [J]. *Mathematical Programming*. 1978, 14 (1): 265–294.
 - [25] Chen W, Wang C, Wang Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks [C]. In *KDD 2010*. 2010: 1029–1038.
 - [26] Borgs C, Brautbar M, Chayes J, et al. Maximizing social influence in nearly optimal time [C]. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2014: 946–957.
 - [27] Kempe D, Kleinberg J, Tardos É. Influential nodes in a diffusion model for social networks [M] // Kempe D, Kleinberg J, Tardos É. *Automata, languages and programming*. Springer, 2005: 2005: 1127–1138.
 - [28] Minoux M. Accelerated greedy algorithms for maximizing submodular set functions [M] // Minoux M. *Optimization Techniques*. Springer, 1978: 1978: 234–243.
 - [29] Leskovec J, Krause A, Guestrin C, et al. Cost-effective outbreak detection in networks [C]. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2007: 420–429.
 - [30] Tang Y, Xiao X, Shi Y. Influence maximization: Near-optimal time complexity meets practical efficiency [C]. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014: 75–86.
 - [31] Goyal A, Lu W, Lakshmanan L V. Celf++: optimizing the greedy algorithm for influence maximization in social networks [C]. In *Proceedings of the 20th international conference companion on World wide web*. 2011: 47–48.
 - [32] Chen W, Wang Y, Yang S. Efficient influence maximization in social networks [C]. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009: 199–208.
 - [33] Leskovec J, Krevl A. SNAP Datasets: Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data>. June 2014.
 - [34] Cheng S, Shen H, Huang J, et al. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization [C]. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. 2013: 509–518.

作者在学期间取得的学术成果

发表的学术论文

- [1] Yang Y, Ren T L, Zhang L T, et al. Miniature microphone with silicon- based ferroelectric thin films. *Integrated Ferroelectrics*, 2003, 52:229-235. (SCI 收录, 检索号:758FZ.)
- [2] 杨轶, 张宁欣, 任天令, 等. 硅基铁电微声学器件中薄膜残余应力的研究. *中国机械工程*, 2005, 16(14):1289-1291. (EI 收录, 检索号:0534931 2907.)
- [3] 杨轶, 张宁欣, 任天令, 等. 集成铁电器件中的关键工艺研究. *仪器仪表学报*, 2003, 24(S4):192-193. (EI 源刊.)
- [4] Yang Y, Ren T L, Zhu Y P, et al. PMUTs for handwriting recognition. In press. (已被 *Integrated Ferroelectrics* 录用. SCI 源刊.)
- [5] Wu X M, Yang Y, Cai J, et al. Measurements of ferroelectric MEMS microphones. *Integrated Ferroelectrics*, 2005, 69:417-429. (SCI 收录, 检索号:896KM.)
- [6] 贾泽, 杨轶, 陈兢, 等. 用于压电和电容微麦克风的体硅腐蚀相关研究. *压电与声光*, 2006, 28(1):117-119. (EI 收录, 检索号:06129773469.)
- [7] 伍晓明, 杨轶, 张宁欣, 等. 基于 MEMS 技术的集成铁电硅微麦克风. *中国集成电路*, 2003, 53:59-61.

研究成果

- [1] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连接的方法: 中国, CN1602118A. (中国专利公开号.)
- [2] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102. (美国发明专利申请号.)

附录 A 模板提供的希腊字母命令列表

大写希腊字母:

Γ \Gamma	Λ \Lambda	Σ \Sigma	Ψ \Psi
Δ \Delta	Ξ \Xi	Υ \Upsilon	Ω \Omega
Θ \Theta	Π \Pi	Φ \Phi	
Γ \varGamma	Λ \varLambda	Σ \varSigma	Ψ \varPsi
Δ \varDelta	Ξ \varXi	Υ \varUpsilon	Ω \varOmega
Θ \varTheta	Π \varPi	Φ \varPhi	

小写希腊字母:

α \alpha	θ \theta	\omicron \omicron	τ \tau
β \beta	ϑ \vartheta	π \pi	υ \upsilon
γ \gamma	ι \iota	ϖ \varpi	ϕ \phi
δ \delta	κ \kappa	ρ \rho	φ \varphi
ϵ \epsilon	λ \lambda	ϱ \varrho	χ \chi
ε \varepsilon	μ \mu	σ \sigma	ψ \psi
ζ \zeta	ν \nu	ς \varsigma	ω \omega
η \eta	ξ \xi	\kappaappa \kappaappa	\digamma \digamma
α \upalpha	θ \uptheta	o \mathrm{o}	τ \uptau
β \upbeta	ϑ \upvartheta	π \uppi	υ \upupsilon
γ \upgamma	ι \upiota	ϖ \upvarpi	ϕ \upphi
δ \updelta	κ \upkappa	ρ \uprho	φ \upvarphi
ϵ \upepsilon	λ \uplambda	ϱ \upvarrho	χ \upchi
ε \upvarepsilon	μ \upmu	σ \upsigma	ψ \uppsi
ζ \upzeta	ν \upnu	ς \upvarsigma	ω \upomega
η \upeta	ξ \upxi		

希腊字母属于数学符号类别, 请用\bm 命令加粗, 其余向量、矩阵可用\mathbf。