doi:10.3772/j.issn.1002-0470.2015.03.005

基于自然语言理解的需求聚类和需求优选方法研究①

童志祥② 马培军 丁 效 郭 琦 初 凯

(哈尔滨工业大学计算机学院 哈尔滨 150001)

摘 要 为了实现大数量软件需求的优选,提出了一种基于自然语言理解的需求聚类和需求优选方法。该方法首先基于自然语言处理技术对需求进行深层次语言理解,找到相同语义的需求并只保留一条。随后,采用基于超图分割的需求聚类方法对大量需求进行聚类。最后,提出了需求优选目标函数,该目标函数根据需求聚类结果为需求优先级打分,并在综合考虑需求与类别的相似性、类别的权重以及聚类标准的权重的基础上,最终给出一个合理的需求优选结果。在大规模真实的需求集合上进行了实验,实验结果表明:基于自然语言处理技术的需求聚类性能优异;需求聚类对后续的需求优选起有着非常重要的作用;基于超图分割的需求聚类,以及在此基础上提出的需求优选函数相对于基线方法有着明显的提高。

关键词 自然语言理解, 需求聚类, 需求优选, 超图分割

0 引 言

随着软件产业的发展,软件系统规模越来越大、复杂程度越来越高[1],软件需求分析活动已不再仅局限于软件开发的初始阶段,它贯穿系统开发的整个生命周期,决定了软件开发的成败。随着软件需要实现的功能的数量以及不同系统间的集成融合程度的不断增大,准确获取用户对目标软件系统的功能、行为、性能、设计等方面的期望的难度越来越大,如何清楚地了解用户希望软件能做什么事情、完成什么样的功能、达到什么样的性能,成为保证软件项目成功实施的重要前提。逐渐地,软件需求分析活动演变成了软件工程的子领域——需求工程(requirements engineering)。需求工程是一个不断反复的需求定义、文档记录、需求演进的过程。20 世纪 80 年代,Herb Krasner 定义了需求工程的五阶段生命周期:需求定义和分析,需求决策,形成需求规格说明,需求实现与验证,需求演进管理。目前学界习惯于将需求工程划分为需求获取,需求分析,需求规格说明,需求验证,需求变更 5 个阶段。事实证明,需求工程方面的问题是导致项目失败的最重要原因。需求工程已成为 863 计划信息技术领域关键技术与应用系统开发研究的基础性工作。然而需求工程的重要性常被低估,有些需求常被忽视、被模糊,甚至有定义不完整和自相矛盾的地方。研究表明,需求缺陷发现得越晚,移除缺陷或者修复缺陷所需要的成本就会越高[2-4],因此,在系统开发的早期就做好需求的获取、分析、优选和管理,既可以减少软件开发中的错误,还可以减少修复错误的费用,从而大大降低软件开发成本,缩短软件开发时间。

软件系统规模、应用和客户规模的不断扩大,开发人员对需求分析过程的重视和强调,必然导致需求重复、需求交叉、需求数量庞大等现象的产生,给后续的需求分析带来极大的工作量和难度。因而,本文在 Herb Krasner 定义的需求工程五阶段的基础

① 863 计划(2012AA011102)资助项目。

② 男,1979 年生,博士生;研究方向:软件工程,需求工程,自然语言处理;联系人,E-mail: tongzhixiang@hit.edu.cn

(收稿日期:2014-10-28)

上增加了需求优选过程,以便在充分获取软件需求的基础上,快速去除重复和非必要的需求,并利用“优选”技术从千差万别的需求中准确地获取核心需求。通过对已有工作的调研发现,现有需求优选技术不能对具有成百甚至上千个潜在需求的项目进行有效的自动优选处理,因此,本文提出了面向大规模需求的优选模型,给出了一种基于自然语言理解的需求聚类和需求优选方法,该方法利用聚类技术将大量繁杂的原始需求凝聚成了数量较少的主题类别,而专家只需要对较少的需求类别的优先级进行评定,最终根据需求与类别的相似性,类别的权重以及聚类标准的权重,给出一个合理的需求优选结果。

1 相关工作

需求工程是软件工程领域中的一个重要组成部分[5],需求获取、需求优选、需求分析、需求验证、需求形式化、需求建模在整个软件开发与维护过程中越来越重要,逐步成为软件工程领域的研究热点。随着软件系统的日益庞大和复杂,需求工程中通常会涉及多个雇主甚至大量的雇主,这些雇主通常会有相互冲突的利益需求,需要从不同的方面考虑到不同雇主之间利益的公平性[6],而且,在这些雇主广泛参与、充分表达需求后,会产生数量庞大的原始需求,决策者的另一个重要工作就是从庞大的需求集合中选取近似最优的需求子集,既要尽可能满足雇主的要求,又能确保有足够的资源来实现选定的需求,这样,需求优选就成为需求工程中的一个关键问题[7]。将优选思想应用于软件工程最早可追溯到 20 世纪 70 年代,最早的工作之一就是由 Miller 和 Spooner 于 1976 年发表的将优选方法应用软件测试领域的研究论文[8],并逐步成为需求工程领域的一个重要研究方向。本文通过充分调研前人的工作,将目前主流的需求优选技术分类两大类:基于排序的需求优选技术和基于搜索的需求优选技术。

1.1 基于排序的需求优选技术

任何优选问题的本质,都是通过给不同的优选对象赋予不同的权值,以便建立起一个集合中对象之间的相对顺序。在需求优选中,这些对象就是指待优选的需求。最简单的需求优选量度是顺序量度,通过利用一定的测量尺度对所有的需求进行排序,这样就可以直观地知道哪些需求比其它需求更重要,但是其缺点是不知道更重要的程度是多少。与顺序量度相比,相对量度的功能显然要更强大一些,因为它能够在 0 ~ 100% 的规模范围内,定量地刻画出一种需求比另一种需求重要的程度。一种更强大的量度是绝对量度,但是绝对度量必须要有一个前提,那就是可以为每个需求分配一个绝对数值,比如说实现一个需求需要多少时间成本和经济成本等定量的描述,就会使得更复杂的评估和计算成为可能。以下将描述一些基于排序的需求优选技术。

1.1.1 层次分析法

层次分析法(analytical hierarchy process, AHP)[9]是美国著名运筹学家、匹兹堡大学教授 T. L. Saaty 于 70 年代中期提出的一种决策方法,它把复杂的问题分解为各组成因素,然后将这些因素按支配关系分组以形成有序的递阶层次结构,通过两两比较的方式确定每一层次中因素的相对重要性,然后在递阶层次结构内进行合成以得到决策因素相对于目标的重要性总顺序。AHP 是一种已经被应用在软件需求优化中的系统决策方法,通过对被层次分类的需求进行成对比较来决策[10],通常用 1 ~ 9 来进行量度,1 表示同等重要,9 代表极端重要,从而确定出哪个需求具有更高的优先级以及优先程度。

1.1.2 质量功能展开

质量功能展开(quality function deployment, QFD)是一种用户驱动的产品开发方法,即采用系统化的、规范化的方法调查和分析顾客需求,并将其转换成为产品特征、零部件特征、工艺特征、质量与生产计划等技术需求信息,使所设计和制造的产品能真正地满足顾客需求。一般来说,QFD 实施包括两个基本过程,顾客需求的提取和顾客需求的瀑布式分解过程,顾客需求的瀑布式分解过程就是将顾客需求分解到产品规划阶段、零件配置阶段、工艺规划阶段、质量控制阶段的各个过程,将顾客需求转换成产品开发过程具体的技术要求和质量控制要求。通过对这些技术和质量控制要求的实现来满足顾客的

需求。QFD 是系统工程思想在产品设计和开发全过程中的具体应用,它将注意力集中于规划和问题的预防上,而不仅仅集中于问题的求解上。

1.1.3 累积投票(或 100 美元测试)

累积投票(Cumulative Voting, the 100-Dollar Test)[11]是一种非常直接的优选技术,这里雇主被给予 100 个虚构的单位(金钱或时间等)在需求之间进行分配,优选结果也是以相对比率来表示的。在累积投票中,必须要求每个优选执行人员应该只执行一次优选,并且限制分配给每个需求的最高虚拟单位数目,以避免雇主在优选中给自己中意的需求偏高的估计。

1.1.4 数值分配

数值分配(Numerical Assignment, Grouping)是一种最常用的优化技术,并由 RFC 2119 and IEEE Std. 830-1998 推荐使用。这种方法基于将需求组织成不同的优先级组,优先级组的数目是可以变化的,但实际中,三组是很常见的[11,12]。每组代表与雇主相关的一个类别,比如关键的、标准的和可选的,以得到需求的优先顺序,从而给需求分配排序的序号。

1.1.5 排序技术

排序技术(Ranking)和数值分配一样,排序也是基于顺序排序的,但是每个需求对应唯一的顺序号(最重要的排序为 1,最不重要的排序为 n)。需求的排序可以由多种方法得到,比如冒泡排序或二叉搜索树算法等等[13],都是比较经典的排序方法。

1.1.6 前十需求

在前十需求(Top-Ten Requirements)技术中,雇主会从一个大的需求集合中挑选他们认为排在前十的需求,而无需为需求分配内部排序号。这种方法特别适合多个同等重要的雇主的情况[14]。不进行排序的原因就是避免雇主间的冲突。

表 1 列出了上述 6 种方法的不足。

表 1 基于排序的需求优选技术存在的不足

序号	相关技术	存在不足
01	层次分析法(AHP)	(1)需求成对比较数目为 $n\times(n-1)/2$,不适合大量需求[15]; (2)只能处理位于同一个更高层次节点下的需求间优选关系,没有解决需求之间的依赖关系等问题
02	质量功能展开(QFD)	(1)需求数量限制在 30 个左右,难以处理大量的需求; (2)很难对需求进行改变,改变就需要全部重新计算;(3)无法检验雇主诚实度,无法解决需求间依赖关系
03	累积投票(100 美元测试)	(1)无法有效处理太多的需求数目[16]; (2)雇主可能会不依据实际的优先级来进行优选,把所有的虚拟单位都分配给他们最喜欢的需求[11]
04	数值分配	(1)属于同一组的需求具有同样的优先级,每个需求不一定具有唯一的优先级,优先级的效用被消弱了[17]; (2)雇主可能会认为所有的需求都是关键的
05	排序技术(Ranking)	(1)可能会导致两个需求具有相同的优先级,即两个需求不可比较的情况出现,不能得到需求间的相对差异; (2)只是适合于只有一个雇主的情况,使多个雇主的意见一致是比较困难的
06	前十需求	有可能产生雇主间的冲突,甚至导致所有的雇主都不满意[14]

1.2 基于搜索的需求优选技术

平衡不同雇主之间的不同期望以及平衡系统和雇主需求之间的矛盾,是需求优选过程中无法回避的难题,这些难题依靠基于排序的优选方式是无法

得到有效解决的,往往需要寻求决策支持算法来平衡这些相互竞争和冲突的条件约束。这些问题本质上都是复杂的优化问题,是典型的 NP 问题,往往没有精确的决策算法,不可能在多项式时间内找到最优解,通常需要采用基于搜索的寻优算法来探索和解决这类复杂的优选问题。基于搜索的需求优选技术,就是要将问题塑造成基于搜索的优化问题,在适应度函数的指导下,寻找最优或近似最优的解决方案。

1992 年,Xanthakis 等[18] 首次将启发式搜索技术应用于解决软件工程问题。2001 年,Harman 和 Jones 首次提出了基于搜索的软件工程(SBSE)这个概念,并首次在文献中阐述了基于搜索的优选算法可以被应用来解决贯穿整个软件工程领域的各个问题,这篇论文对于 SBSE 具有里程碑的意义。

1.2.1 基于单目标的需求优选

Bagnall 等[19] 在研究软件下一发行版本问题(next release problem, NRP)时,发现这其实是一个典型的需求发布计划问题,也就是在软件的下一个发行版本中应该选择实现哪些需求的问题,其优化目标就是在有限的资源约束条件下,最大化所满足的雇主群体。Bagnall 等在研究中,首先将 NRP 形式化为一个有约束的单目标优化问题,运用贪婪算法、分支定界算法、模拟退火算法和爬山算法等多种启发式优化算法,寻找理想的需求集合来平衡雇主期望与资源约束之间的矛盾,并应用一组仿真数据集来阐述基于搜索的软件工程(SBSE)技术对于解决此类下一发行版本问题(NRP)(需求优选问题)的可行性。这个基于单目标优化搜索的 NRP 模型是将 SBSE 方法应用于需求分析领域的首次尝试。

在后来的研究中,Ruhe 等[20-22] 提出了基于遗传算法的需求优选方法,分别将平衡需求实现代价和可利用资源之间的冲突、需求实现顺序和雇主自身优先级之间的冲突以及需求变化和需求间依赖关系的冲突作为需求优化的单一目标,逐一用迭代的方式获得候选需求集合,最后通过赋予每一个解集以权重,综合各方面考虑,为 NRP 提供最后的需求集合。Feather 和 Menzies[23] 应用模拟退火算法建立了一个迭代模型进行优化搜索,将需求分析作为选择和优化问题来解决,并首次将帕累托最优应用于 SBSE 问题,然而这里的帕累托前端(pareto front)不是由多目标优化技术产生的,而是由一个单目标公式的迭代使用而产生的。Baker 等[24] 利用贪婪算法和模拟退火算法,对候选的软件组进行排序和选择,筛选出在一个软件系统的不同版本中所需要包括的组成部分。同样 Harman 等[25] 也将需求分析看作是特征子集优选问题,并将此方法成功应用于来自摩托罗拉的一个真实数据集合。

1.2.2 基于多目标的需求优选

现今的软件系统,一方面有很多的雇主,而且往往这些雇主又对需求的选择持有相互矛盾的意见,那么亟待解决的问题就是如何选择出能够反映很多不同雇主需要的需求集合;另一方面,每个雇主都期待软件有更完整的功能、更满意的质量、更快捷的效率、更低廉的成本。因此,在很多情况下,需求优选过程中需要同时兼顾多个目标。由于不同的目标评价很难进行统一表示和计算,不能简单地将它们合并为单一(加权)的目标函数,因而必须将每个目标都作为一个单独的优化目标,通过优化算法选择那些会使某一个目标更好,而不会使其它目标变坏的解,称作非劣最优解集,也就是所谓的帕累托(Pareto)最优解集。Pareto 最优解集中的每个解决方案都表示一种可能的需求分配或选择方案,这个需求优选方案可以在提高某些优化目标的同时,不损害其它优化目标,并且最大限度地提高整体的优化目标。Srinivas 和 Deb 在 1995 年,基于 Pareto 最优概念,将非支配排序遗传算法(NSGA)[26] 运用于多目标优化,在基本遗传算法的基础上,对选择再生方法进行改进,将每个个体按照它们的支配与非支配关系进行分层,再做选择操作,从而使得算法在多目标优化方面得到非常满意的结果。

Finkelstein 等[27,28] 的研究表明多目标优化技术可应用于解决需求分配的公平性,他们将不同定义的公平性(fairness)作为不同的优化目标,利用多目标优化技术来同时优化不同的公平性目标。Zhang 和 Durillo 等[29,30] 同时考虑最小化供应商的成本和最大化雇主满意度的双重目标。Saliu 和 Ruhe[31] 同时考虑了实施和需求这两个层面的优化目标,依据

商业需求满足和实施效益两个角度,来优化软件的发布计划。Zhang[29]等在2007年的研究论文中,首次将NRP问题概括为多目标NRP(MONRP)问题,提出了基于多目标优化的NRP问题模型,将"成本"限制和"价值"追求作为两个独立的目标进行优化,来平衡和优化价值和成本之间的矛盾。他们应用非支配排序遗传算法(NSGA),将多目标搜索技术用于一个实证研究来阐述基于多目标优化搜索的需求优选或NRP问题。Zhang[29]等于2011年首次提出将每个雇主的满意度作为单独的优化目标。

1.2.3 基于搜索的需求互作管理

在进行软件开发规划或需求优选时,需求间的相互依赖关系是一个重要的影响因素,这些依赖关系反映了需求之间的技术、结构、功能等方面的相互关联。举一个简单的例子,假如根据已有方法计算出需求A的优先级高于需求B,但是,如果需求A必须要在需求B完成之后才能进行,也就是说需求A是依赖于需求B的,那么该需求优选模型给出的结果则是不合适的。这就使得需求优选中的优化问题变得更加复杂[32,33]。分析和管理这些需求间的相互依赖关系就叫做需求互作管理(requirement interaction management, RIM)[34,35]。RIM会对需求的选择过程,以及需求的追溯管理、重用和进化过程产生直接影响。

为了解决需求之间的依赖问题,Carlshamre[36]曾经利用线性规划技术将需求间的相互依赖关系考虑在优选模型内。Ruhe和Saliu[37]同样提出了一种基于整数线性规划的方法通过整合计算智能和人为判断以解决他们之间的目标冲突。Van den Akker等[38,39]进一步扩展了此类线性规划技术,开发了一个基于整数线性规划的优化工具,以期找到最佳的一组需求,在有限的预算限制下,能够获得最大收益。Bagnall等[19]考虑了需求间的优先级依赖关系(Precedence dependency),即一个需求必须在另一个需求之前优先被实现。在他们的模型中,这种依赖关系被表示成为一个有向无环图,图中节点表示需求,有向边表示需求间的优先级依赖关系。Greer和Ruhe[33]对Bagnall的工作进行了扩展,在考虑优先级依赖关系的同时还加入了"And"依赖关系,即一个需求必须和另一个需求必须同时被选择或实现,他们在进化模型中将"Precedence"和"And"依赖关系作为两个约束条件来指导优化搜索。Frank Moisiadis针对需求之间相互依赖问题,利用二进制矩阵或树结构来表示需求之间依赖关系,开发出了需求优选工具(RPT)[40]。

1.2.4 基于雇主社交网络和协同过滤的需求优选

为解决雇主增加和需求增加带来的计算复杂度急剧增加,Lim等[41]提出了一个利用雇主社交网络和协同过滤来识别和优选大量雇主需求的算法,叫做StakeRare。StakeRare通过计算雇主对软件项目及需求的影响因子来给需求打分,根据得分选出优选需求。此算法主要有以下4个步骤:(1)利用社交网络识别和优选雇主;(2)收集雇主需求信息;(3)需求预测;(4)需求优选。该方法已经被应用于伦敦大学学院的RALIC[42]项目,结果显示,这个方法能够识别出一个合理的雇主和需求的集合,并且正确地优选需求。

表2列出了上述4种方法的不足。

1.3 相关工作简析

基于排序的需求优选算法,虽然都是通过给不同的优选对象赋予不同的权值,从而能够清晰地给出需求之间的相对排序,但是无论是层次分析法、质量功能展开,还是累积投票、数值分析等,都无法有效解决大量需求的有效排序、需求之间的依赖关系处理、雇主主观偏好规避、多雇主之间的冲突平衡(如表1)。这些问题的存在,都给基于排序的需求优选技术的广泛应用带来了挑战。

虽然基于多目标的需求优选可以为解决无约束或简单约束的优化问题找到很好的方案,在多目标优化、需求依赖、雇主冲突等方面取得了一些突破,但是仍不能说就已经找到了需求优选的万能钥匙,特别是在解决大量需求优选、多重需求依赖管理、大量雇主冲突平衡等方面,还有许多亟待提高和探索的空间(如表2)。

通过文献调查,我们发现现有优选技术不能对有上百需求甚至上千潜在需求的项目进行有效自动分类处理,因此有必要对现有技术和方法进行改进,利用聚类技术将原始大量繁杂的需求凝聚成数量较

少的主题或类别,这样就只需要在类别之间进行比较,可以有效减少比较次数,提高算法效率,使雇主有更多的精力关注更高层次或更有影响力的需求,在需求聚类层次作决策或需求评价,而不是陷入对具体的、繁杂的需求评级之中,使得需求优化得到更好的结果,这是影响需求优选的一个基础性问题。

表 2 基于搜索的需求优选技术存在的不足

序号	相关技术	存在不足
01	基于单目标的需求优选	(1)无法满足多个雇主、多个目标期望; (2)最优化一个目标的同时可能以牺牲另一个潜在目标的最优化为代价,导致解的搜索偏向于解空间的某一个特定部分
02	基于多目标的需求优选	(1)解决高约束问题,或优化目标急剧增多后,会遇到困难; (2)优选算法以假设需求之间是相互独立为基础,没有考虑需求之间的互斥、相关等依赖关系,无法达到理想的优选效果
03	基于搜索的需求互作管理	(1)无法有效体现成本相关等不能转化成不等式约束条件的弱依赖关系; (2)现有算法还不能满足实际存在的多重需求依赖关系解决需要
04	基于雇主社交网络和协同过滤的需求优选	算法假设雇主对项目的影响因子是固定不变的,但事实上雇主对项目的影响是随着时间变化的,固定不变的影响因子可能会导致不准确的优选结果

2 本文方法

基于对大规模需求的优选问题分析,本文提出了基于自然语言处理和需求聚类的需求优选模型,该模型包括:(1)基于自然语言处理技术的需求数据去噪;(2)基于超图分割的需求聚类方法;(3)基于需求聚类结果的需求优选目标函数构建;(4)根据需求优选目标函数输出的每个需求的优先级分数,对需求进行排序,最终得到需求优选结果。该模型对需求的优选过程如图 1 示。

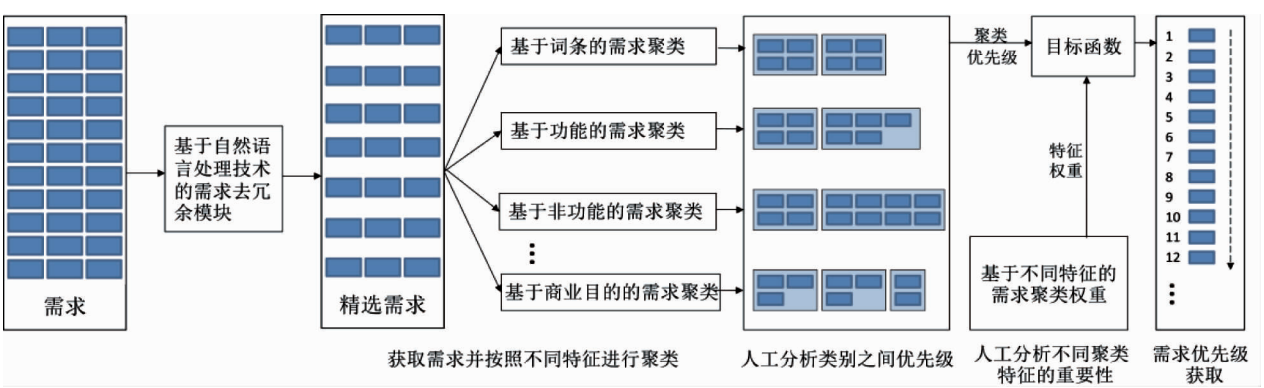


图 1 面向大规模需求的优选模型

2.1 基于自然语言处理技术的需求数据去噪

本文提出基于自然语言处理技术对需求进行深层次语言理解,以期达到去噪声的目的。我们看这样一个例子:“系统的响应速度要快”和“系统响应的时间要短”这两个需求仅从字面上看是两个不同的需求,但是经过深入分析可知,“速度要快”和“时间要短”在语义上完全相同。这样两条需求同时出现在项目中实际上是重复的,因此本文通过自然语言处理技术去掉这些重复需求。具体处理流程如下图 2 示。

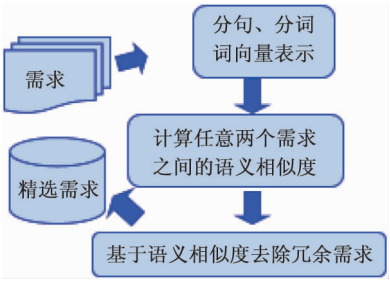


图 2　利用自然语言处理技术去除冗余需求的方法流程

(1)对原始需求进行分句、分词等基本自然语言处理,将需求表示成词向量。例如:"点击确定按钮可以登录系统"可以表示成 < 点击,确定,按钮,可以,登录,系统 > 这样一个词向量。

(2)计算任意两个需求之间的语义相似度。需求相似度的计算由需求词之间的相似度计算体现。本文中,利用语义相似度描述两个需求词 V_i 和 V_j 的相似度,其相似度值 $Sim(V_i, V_j)$ 根据 HowNet(中文的开源语义词典)计算得到,$Sim(V_i, V_j)$ 由 V_i 和 V_j 相同"义原"(最基本的不易于再分割的语义的最小单位)的数量 N_s 除以两者的"义原"总和实现归一化得到,如式

$$Sim(V_i, V_j) = \frac{2N_s}{N_i + N_j} \tag{1}$$

所示。其中 N_s 表示需求词 V_i 和 V_j 在 HowNet(中文语义词典)概念意义 DEF (the concept definition in HowNet) 中具有相同"义原"的数量,N_i 和 N_j 分别表示 V_i 和 V_j 概念定义中"义原"的数量。因此,计算两个需求之间的语义相似度,即为两个词的相同义原数除以两个词各自义原的总数。由于归一化需要,保证 $Sim(V_i, V_j)$ 最大为 1,最小为 0。当 $N_i = N_j$ 时,$N_s = N_i = N_j$,这时 $Sim(V_i, V_j)$ 应该取得最大值 1,如果不乘以 2 的话,此时最大的 $Sim(V_i, V_j)$ 只能等于 0.5。

(3)当需求之间的相似度达到某一设定阈值时,则认为这两个需求实际上在表述同一件事件,可以去掉一个。

2.2　基于超图分割的需求聚类方法

本文采用基于超图分割的需求聚类方法对需求进行聚类,其相对于在需求聚类领域常用的基于成对比较的算法(如层次聚类等)的优势在于:(1)可以满足类别内部的高度一致性,成对比较的需求聚类在计算需求相似性时,只考虑两两需求之间的信息,而超图则考虑全局的需求信息;(2)可以满足类别外部的高度不一致性,成对比较的需求聚类只考虑了类别内部的相似性,而没有考虑类别之间的高度不一致性;(3)传统的需求聚类算法一般需要事先指定或预判聚类数量,而聚类数的设定往往带有经验性并且对聚类结果的影响非常大,基于超图的需求聚类则不需预先指定类别数量。

基于超图的需求聚类方法主要分为 4 个主要步骤:

(1)数据建模,将需求表示为一系列抽象词根组成的向量,并利用欧氏距离来进行需求之间的相似性评分,作为需求间的距离量度。

(2)超图构造,利用需求间的距离以及 k-近邻算法构造出 k-近邻超图,并将两点之间的相似度作为边的权值。

(3)超图分割,采用超图的 hMeTiS 算法对前面构造的带权 k-近邻超图进行分割,形成大量的小的聚类。

(4)优化聚类,利用聚类的类间连接性和相关性计算类间的相似性,若相似性大于某给定阈值时则合并,否则不合并,最终形成优化的聚类。

算法框架示意图如图 3 示。

(1)数据建模。我们获得的每个原始需求是一段文本描述,对这些需求文本进行预处理就是去除其中常见的停用词,并且把剩余的词转换为词根形式。这样每个需求就被表示成为一系列的抽象词条。在数据建模时,将需求展开形成一个 n 维词条向量,即:$\overrightarrow{R_{e_l}.R_eP}$ = $(R_{e_i}.R_eP_1, R_{e_i}.R_eP_2, \cdots, R_{e_i}.R_eP_n)$,其中 R_{e_i} 为第 i 个需求,$R_{e_i}.R_eP_k$ 为第 i 个需求的第 k 个词。因此可以采用欧氏距离计算两个需求 R_{e_i} 和 R_{e_j} 之间的距离,如公式

$$d(R_{e_i}, R_{e_j}) \equiv \sqrt{\sum_{x=1}^{n} w_x \cdot (R_{e_i}.R_eP_x - R_{e_j}.R_eP_x)} \tag{2}$$

所示。其中 $\sum_{x=1}^{n} w_x = 1$。

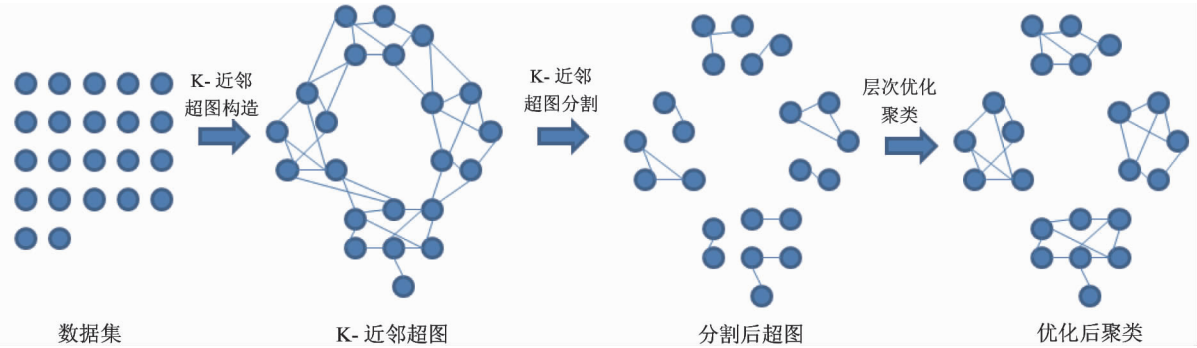


图 3　基于超图分割的需求聚类方法示意

（2）超图构造。在构造超图 $H=(V, E)$ 过程中，将需求集映射为超图中的顶点集 V，那么构造超图的关键问题就变成了如何构造超边以及如何确定超边的权重，本文将欧式距离作为边的权重，并采用 k-近邻法进行超图构造，超图的边为节点的 k-近邻，而边的权值为节点间的相似度。其算法原理如图 4 示。

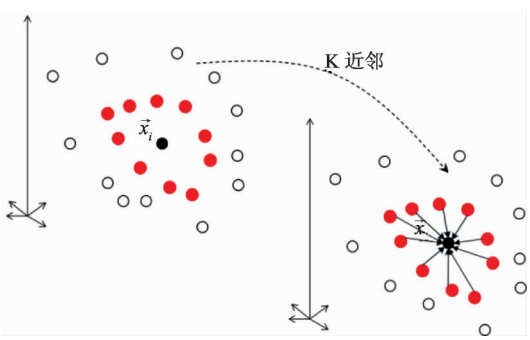


图 4　超图构造示意图

（3）超图分割。在完成超图构造后，要对超图进行分割，本文采用 hMeTiS 算法进行分割。hMeTiS 算法主要分为 3 个步骤：（i）超图粗化；（ii）二分超图；（iii）分割精化。其中超图粗化是将原始图中的相似的顶点合并成一个顶点以达到降低原始超图规模的目的，然后对小规模超图进行分割，最后通过超图粗化的反向细化逐步迭代分割，直到映射回原始超图，并将超图划分成大量的小的聚类。

（4）优化聚类。优化聚类是在 k-近邻分割后，通过聚类的类间相关性和类间连接性计算出两个类之间的相似性，若相似性大于某个相似阈值，则将两个聚类合并，依次合并，直到相似性值小于相似阈值或者形成了一个分类，即可得到树形层次聚类结果，计算每一层的聚类的性能值，性能值最大的为最优聚类。其中类间的相关性计算公式为

$$\mathrm{In}(C_i, C_j) = \frac{V_{\{C_i, C_j\}}}{\frac{|C_i|}{|C_i|+|C_j|} \times V_{C_i} + \frac{|C_j|}{|C_i|+|C_j|} \times V_{C_j}} \tag{3}$$

其中 $V_{\{C_i, C_j\}}$ 为连接聚类 C_i 和 C_j 的顶点权值（本文认为需求是等重要的，权值都设为 1）的平均值，同样 V_{C_i} 则是将聚类 C_i 进行二分之后的连接顶点的权值的平均值，$|C_i|$ 是聚类 C_i 的顶点个数。可见相关性越高，类间相似性越大。

类间的连接性计算公式为

$$\mathrm{Ex}(C_i, C_j) = \frac{2 \times |W_{\{C_i, C_j\}}|}{|W_{C_i}|+|W_{C_j}|} \tag{4}$$

其中 $W_{\{C_i, C_j\}}$ 为连接聚类 C_i 和 C_j 的超边的权值之和，而 W_{C_i} 则是将聚类 C_i 进行二分之后的割边权值之和。可见连接性越高，类间相似性越大。

因此类间的相似性可以表达为

$$Sim(C_i, C_j) = Ex(C_i, C_j) \times \mathrm{In}(C_i, C_j) \tag{5}$$

即类间相似性等于类间连接性乘以类间相关性。若 $Sim(C_i, C_j)$ 大于某一相似性阈值，则合并。

2.3　基于需求聚类结果的需求优选目标函数构建

本文提出的需求聚类与以往工作的不同之处在于，以往的需求聚类仅是根据需求的某一方面需求，比如，根据商业目的对需求进行聚类。本文提出综合考虑需求的不同特征进行聚类，例如需求的词特征、需求的功能性特征、需求的非功能性特征等。其中，基于需求的词特征是根据词的欧氏距离进行相似度计算，而基于功能性特征和非功能性特征的聚

类则由人工进行。基于需求聚类的结果,本文提出了新的需求优选模型,其目标函数为

$$PS_r = Pr(C_i \mid r) \times RC_i \times w_r,\ r \in C_i \quad (6)$$

其中,PS_r 表示需求 r 的优先级得分;C 表示需求聚类集合$\{C_1, C_2, \cdots, C_{|C|}\}$;$Pr(C_i|r)$表示需求 r 与聚类中心的距离;RC_i 表示第 i 个需求聚类的优先级分数(由雇主给定),w_r 表示不同聚类特征的重要性(例如词特征、功能性特征等),也由雇主给定。此公式说明,需求 r 的优先级分数等于需求 r 与所在聚类的中心点的欧氏距离乘以该聚类的重要性分数。也就是说需求 r 的优先级有三方面决定:一是根据什么特征进行聚类且该特征重要性多大;二是需求 r 所在聚类类别的优先级;三是需求 r 与聚类中心的距离。如果聚类特征的重要性较高,聚类类别的优先级较高,且需求 r 处在聚类中心位置,则需求 r 将会得到较高的优先级得分。

2.4 需求优先级排序

需求优选目标函数可以为每个需求进行优先级打分,最终的需求优选结果就是根据该打分由高到低对需求进行排序。

3 实验设置

3.1 实验数据

为了验证算法的正确性,本文采用真实的需求数据对算法进行了测试。实验中,采用了“车友说”网站收集的需求作为实验数据。“车友说”网站是一个可以进行交互的论坛网站,用户包括普通车友、会长、管理员等角色。在人工收集得到的数据中,随机选择了 100 条需求作为实验数据,每条需求都是采用自然语言的形式进行描述的。实验数据经过人工分类,确定了理想划分结果,将需求数据划分为 8 类,其中每一个类中的需求表述的语义是相同的,因此同一个类中的数据认为是冗余数据,应该只保留一个。

聚类实验中需要设置参数,即相似度阈值,相似度阈值越大,同一个类中需求的相似度越高。相似度阈值取 0.5 ~ 1 之间,以 0.25 为间隔进行了 20 组实验,将实验产生的分类结果与理想分类结果进行对比,并通过计算准确率、召回率和 F 值,得到了算法的正确性度量。

对于需求优选的最终排序,本文采用人工标注结果,根据其重要性和开发代价人工进行标注。本研究首先邀请两位需求分析师对需求进行独立排序,对于排序不一致的地方,则由第三位需求分析专家进行最终的裁定。

3.2 评价方法

对于需求聚类的实验结果,本文采用加权准确率(*Precision*,*P*),加权召回率(*Recall*,*R*),加权 F 值(*F-Measure*,*F*)和纯度(*Purity*)来衡量每个事件类别的效果。具体衡量指标计算定义如下:

$$p(i, r) = n(i, r)/n_r \quad (7)$$

$$r(i, r) = n(i, r)/n_i \quad (8)$$

$$f(i, r) = \frac{2 \times p(i, r) \times r(i, r)}{p(i, r) + r(i, r)} \quad (9)$$

$$F = \sum_i (n_i/n) \max\{f(i, r)\} \quad (10)$$

$$Purity = \sum_r (n_r/n) \max\{p(i, r)\} \quad (11)$$

其中,i 是标准类别,r 是实验得到的类别,且该类别与标准类别 i 有最多的公共触发词。n_i 是类别 i 中的触发词个数,n_r 是类别 r 中的触发词个数,n 是所有触发词数目。对于每一个类别,首先分别计算其 $p(i,r)$、$r(i,r)$和$f(i,r)$,然后根据这个计算结果算出最终整个聚类结果的 *F-Measure* 和 *Purity*。

本文将需求优选的评价看成排序问题,将需求根据其实施的先后顺序以及重要性进行排序。对于排序问题,本文采用的评价指标是平均准确度均值(mean average precision,MAP),计算公式如下:

$$MAP = \exp \frac{1}{n} \sum_{i=1}^{n} \log AP_i \quad (12)$$

其中,*AP* 在本文中采用 P@ 10(前十排序)的结果(例如,前 10 个排序结果中,有 8 个与标准结果相同,则 $AP = 8/10 = 0.8$)。*MAP* 取值是[0, 1]区间任意实数。

3.3 基线实验

为了验证本文提出的方法的有效性,本研究设计了几组基线方法用作对比实验。由于基于自然语言处理的需求优选任务是由本文首次提出的,没有前人提出的方法可以用来直接作为基线方法,因此,

本文提出了以下两种基线方法。

基线方法 1：为了评估本文首次提出的基于自然语言处理技术去除数据噪声对最终的需求优选方法有效性有多大影响，该方法将去掉该模块，也就是将原始的需求不经过自然语言处理去除数据噪声直接进行需求聚类和优选。

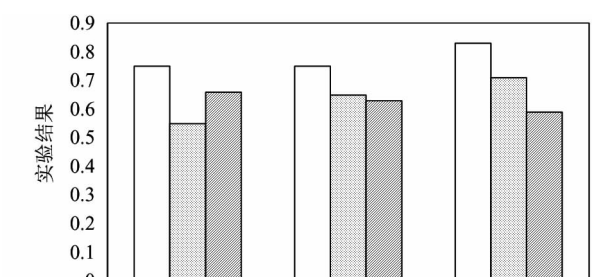
基线方法 2：为了验证本文首次提出的基于超图的需求聚类方法的有效性，该方法采用 K-means 方法作为聚类的对比方法，这是目前被广泛使用的聚类方法之一。

4　实验结果与分析

本节首先介绍实验的总体评价，包括三个不同方法（本文方法、基线方法 1 和基线方法 2）在三个评价指标（*F*，*Purity*，*MAP*）下的对比实验结果。本文在随后的实验分析中详细分析了各个方法的优势与不足，并给出了本文方法在不同参数条件下的实验结果分析。

4.1　实验结果

本研究采用 4.2 节中给出的评价标准进行了详细的实验测试，整体的实验结果如图 5 所示。通过对实验结果的观察我们发现：



	F	Purity	MAP
本文方法	0.75	0.75	0.83
基线方法1	0.55	0.65	0.71
基线方法2	0.66	0.63	0.59

图 5　实验结果

（1）本文方法在三个评价指标上都明显高于两个基线方法，这证明了本文方法的有效性，对于需求聚类和需求优选都有很好的帮助。

（2）本文方法明显高于基线方法 1。主要原因在于，基线方法 1 没有对需求数据去噪，大量的相同或相似需求直接被用来聚类和优选，这样导致聚类和优选的算法效果明显下降。而且冗余的需求对于最终的需求优选也会有很大的干扰，相同或相似的两个需求都被排到前面是没有任何意义的，并且还会浪费需求开发者的分析和开发时间，这对需求工程来讲是一种浪费。

（3）本文方法明显高于基线方法 2，主要原因在于本文采用了较为先进的超图分割聚类算法，而基线方法 2 采用的是 K-means 聚类算法。超图分割聚类算法将聚类问题转换为组合优化问题，并利用图论和相关启发式算法来解决，构造数据集的最小生成数，再逐步删除最长边，其优点在于不需要进行相似度的计算。而 K-means 聚类算法结果好坏依赖于对初始聚类中心的选择、容易陷入局部最优解，对 K 值的选择没有准则可依循，对异常数据较为敏感，只能处理数值属性的数据，聚类结构可能不平衡。

4.2　实验分析

本文除了对实验结果进行整体评价之外，还进行了一些较为细致的实验分析。对于需求聚类，本文首先进行了需求去冗余操作。该操作中需要计算两个需求的相似度，对于相似度高于某一阈值的，即认为这两个需求相似度极大，如果同时进行后续操作会造成数据冗余，因此，认为存在需求冗余现象。本文分析认为，该阈值的选取对于去冗余以及后续的需求聚类结果都会有比较明显的影响，因此，本文对该参数进行的详细的实验分析，分析结果如图 6 所示。

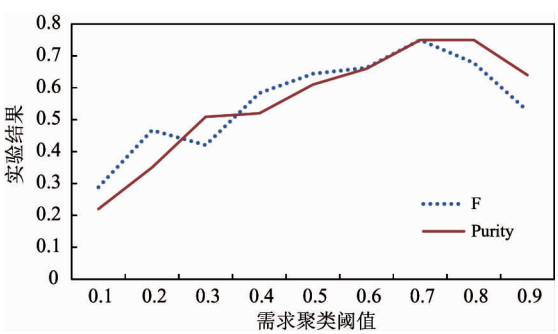


图 6　不同需求聚类阈值结果

从图 6 中我们可以看到，对于不同的阈值取值，需求聚类的结果会有比较大的波动。当需求阈值在 0.1 到 0.7 之间时，随着需求阈值的增加，实验结果

也有明显的提高。这说明随着阈值的加大,大量的冗余需求被剔除出去,因此实验结果会有明显的提高。但同时我们也发现,当需求阈值在 0.7 到 0.9 之间时,随着需求阈值的增大,实验结果不升反降。这一结果也不出乎预料,因为随着需求阈值的增大,会有更多的需求被剔除出去,而这难免会有非冗余需求被剔除出去,这对后续的实验结果就会造成非常差的影响。因此,本文认为在该项阈值选择上,0.7是一个比较合适的取值。

5 结 论

本文首次提出了基于自然语言处理技术对需求进行深层次语言理解,找到相同语义的需求并只保留一条,以期达到去冗余的目的。以往的需求优选模型很少考虑需求的冗余性问题。同样的一个需求用两个不同的句子描述就被看成了两个不同的需求,这大大增加了需求优选模型的负担。当需求规模较小时,这一问题往往会被忽略掉,但是当需求规模巨大时,需求的冗余性问题就凸显了出来。此外,本文采用基于超图分割的需求聚类方法,其相对于在需求聚类领域常用的基于成对比较的算法(如层次聚类等)的优势在于:(1)可以满足类别内部的高度一致性,成对比较的需求聚类在计算需求相似性时,只考虑两两需求之间的信息,而超图则考虑全局的需求信息。(2)可以满足类别外部的高度不一致性,成对比较的需求聚类只考虑了类别内部的相似性,而没有考虑类别之间的高度不一致性。(3)传统的需求聚类算法一般需要事先指定或预判聚类数量,而这个数量的设定对结果的影响非常大,基于超图的需求聚类则不需预先指定类别数量。最后,本文提出了需求优选目标函数,该目标函数针对不同标准的需求聚类结果给需求进行优先级打分,综合考虑了需求与类别的相似性、类别的权重以及聚类标准的权重,最终给出一个最合理的需求优选结果。通过对大规模的需求进行实验,结果表明本文提出的方法对于需求聚类问题起到明显的帮助作用。

参考文献

[1] Honsing M. Mit Vollgas in die Krise: Die auto-industrie, eingekeilt zwischen kosten-und innovationsdruck, braucht neue konzepte. *Technology Review*, *Ausgabe*, 2005, 5: 42-55

[2] Möller K H. Ausgangsdaten fur Qualitatsmetriken-Eine Fundgrube fur Analysen. Software-Metriken in der Praxis, Springer Berlin Herdelberg, 1996. 105-116

[3] Boehm B W, Ross R. Theory-W software project management principles and examples. *IEEE Transactions on Software Engineering*, 1989, 15(7): 902-916

[4] Davis A M. Software Requirements: Objects, Functions, and States. Prentice-Hall, Inc. Upper Saddle River, 1993

[5] Cheng B, Atlee J. From state of the art to the future of requirements engineering. In L. Briand and A. Wolf, editors, Future of Software Engineering 2007, Los Alamitos, California, USA. IEEE Computer Society Press

[6] Rana A, Singh S P, Soni R, et al. Challenges of global stakeholder's in software release. In: Proceedings of the 2014 International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2014. 551-555

[7] Alrashoud M, Abhari A. Perception-Based Software Release Planning. *Intelligent Automation & Soft Computing*, 2014 (ahead-of-print): 1-21

[8] Miller W, Spooner D L. Automatic Generation of Floating-Point Test Data. *IEEE Transactions on Software Engineering*, 1976, 2(3): 223-226

[9] Saaty T L. How to make a decision: the analytic hierarchy process. *European journal of operational research*, 1990, 48(1): 9-26

[10] Regnell B, Nattoch H M, Dag J, et al. An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software. *Requirements Engineering*, 2001, 6(1): 51-62

[11] Leffingwell D W D. Managing Software Requirements: A Unified Approach. Boston: Addison-Wesley Longman Publishing Lo. 2000

[12] Sommerville I, Sawyer P. Requirements Engineering: A Good Practice Guide. New York: John Wiley & Sons, 1997

[13] Karlsson J W C, Regnell B. An Evaluation of Methods for Prioritizing Software Requirements. *Information and Software Technology*, 1998, 39(14): 939-947

[14] S, L. Software Requirements: Styles and Techniques. New York City: Pearson Education, 2002

[15] Lehtola L, Kauppinen M. Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects. Software Process Improvement, 2004

[16] Berander P W C. Differences in Views between Development Roles in Software Process Improvement - A Quantitative Comparison. In: Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering, Edinburgh, UK, 2004. 57-66

[17] Karlsson J K. A Systematic Approach for Prioritizing Software Requirements. Linköpings Universitet, Sweden, 1998

[18] Xu B W, Xie X Y, Shi L, et al. Application of genetic algorithms in software testing. *Advances in Machine Learning Application in Software Engineering*, 2007, 287-317. Doi: 10.4018/978-1-59140-941-1.ch012

[19] Bagnall A J, Rayward-Smith V J, Whittley I M. The next release problem. *Information and Software Technology*, 2001, 43(14): 883-890

[20] Greer D, Ruhe G. Software release planning: An evolutionary and iterative approach. Information Software Technology, 2004, 46, 4, 243-253

[21] Ruhe G, Greer D. Quantitative studies in software release planning under risk and resource constraints. In Proceedings of the International Symposium on Empirical Software Engineering (ISESE'03), Roman Castles, Italy, 2003. 262-270

[22] Ruhe G, The A N. Hybrid intelligence in software release planning. *International Journal of Hybrid Intelligent Systems*, 2004, 1(1): 99-110

[23] Feather M S, Menzies T. Converging on the optimal attainment of requirements. In: Proceedings of the 10th IEEE International Conference on Requirements Engineering, Essen, Germany, 2002. 263-270

[24] Baker P, Harman M, Steinhofel K, et al. Search based approaches to component selection and prioritization for the next release problem. In: Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM'06), Philadelphia, USA, 2006. 176-185

[25] Harman M, Skaliotis A, Steinhfel K. Search-based approaches to the component selection and prioritization problem. In: Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation (GECCO '06), Seattle, USA, 2006. 1951-1952

[26] Srinivas N, Deb K. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 1994, 2(3): 221-248

[27] Finkelstein A, Harman M, Mansouri S A, et al. "Fairness analysis" in requirements assignments. In: Proceedings of the 16th IEEE International Requirements Engineering Conference (RE'08), Catalungy, Spain, 2008. 115-124

[28] Finkelstein A, Harman M, Mansouri S A, et al. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making, *Requirements Engineering*, 2009, 14(4): 231-245

[29] Zhang Y, Harman M, Mansouri S A. The multi-objective next release problem. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07), London, England, 2007. 1129-1137

[30] Durillo J J, Zhang Y Y, Alba E, et al. A study of the bi-objective next release problem. *Empirical Software Engineering*, 2011, 16(1): 29-60

[31] Saliu M O, Ruhe G. Bi-Objective release planning for evolving software systems. In: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Dubro Vnik, Croatia, 2007. 105-114

[32] Franch X, Maiden N A M. Modelling Component Dependencies to Inform Their Selection. COTS-Based Software Systems, Springer Berlin Heidelberg, 2003. 81-91

[33] Greer D, Ruhe G. Software release planning: An evolutionary and iterative approach. *Information & Software Technology*, 2004, 46(4): 243-253

[34] Zhang Y, Harman M. Search based optimization of requirements interaction management. In: Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10), Benevento, Italy, 2010. 47-56

[35] Zhang Y Y, Harman M, Lim S L. Empirical evaluation of

search based requirements interaction management. *Information and Software Technology*, 2013, 55(1): 126-152

[36] Carlshamer P. Release Planning in market-driven software product development: Provoking an understanding. *Requirements Engineering*, 2002, 7(3): 139-151

[37] Ruhe G, Saliu M O. The art and science of software release planning. *IEEE Software*, 2005, 22(6): 47-53

[38] Li C, Van Den Akker J M, Brinkkemper S, et al. Integrated Requirement Selection and Scheduling for The Release Planning of A Software Product. Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2007. 93-108

[39] Van Den Akker M, Brinkkemper S, van Diepen G, et al. Flexible release planning using integer linear programming. *REFSQ'*05, 2005

[40] Moisiadis F. The fundamentals of prioritising requirements. In: Proceedings of the Systems Engineering, Test and Evaluation Conference, Sydney, Australia, 2002

[41] Lim S L, Damian D, Finkelstein A. StakeSource 2.0: using social networks of stakeholders to identify and prioritise requirements. In: Proceedings of the 33rd International Conference on Software Engineering, Honolulu, USA, 2011. 1022-1024

[42] Zhang Y Y, Harman M, Finkelstein A, et al. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation. *Information and Software Technology*, 2011, 53(7): 761-773

Research on clustering and prioritizing of software requirements based on natural language understanding

Tong Zhixiang, Ma Peijun, DingXiao, Guo Qi, Chu Kai

(School of Computer Science and Technology Harbin Institute of Technology, Harbin 150001)

Abstract

To realize the optimal selection of great quantities of software requirements, a new requirement prioritization approach based on natural language understanding is presented. The approach is described below. Firstly, it uses the natural language processing technique to deeply understand requirements from the angle of language and find the requirements of same semanteme, with one of them being reserved. Secondly, it adopts a requirement clustering method based on hypergraph partitioning to cluster quantities of requirements. Finally, it presents an object function for optimal selection of requirements. The function gives marks for requirement priorities according to requirement clustering results, and at last, gives a rational requirement clustering results, and at last, gives a rational requirement prioritization result based on the comprehensive consideration of the similarty between requirement and classification, the weight of classification, and the weight of clustering standard. The results of the experiment performed on a real large-scall requirement set show that the performance of the requirement clustering based on the natural language processing is excellent, the requirement clustering is very important to the follow-up requirement prioritization, and the requirement clustering based on hypergraph partitioning and the function for requirement prioritization outperform the base line method.

Key words: natural language understanding, requirement clustering, requirement prioritization, hypergraph partitioning