一种基于重叠社区发现的软件特征提取方法

刘 春 张国良
(河南大学计算机与信息工程学院 河南 开封 475001)

摘 要 近年来从软件产品的文本描述中提取软件特征获得了大量关注。考虑到产品文本描述中的句子能够更加清晰地表达一个特征的含义,并且文本描述中的每个句子可能会涉及多个软件特征,文中提出了一种通过发现软件产品文本描述中重叠的句子聚簇来提取软件特征的方法。基于复杂网络中的 LMF 重叠社区发现算法,所提方法通过自定义文本描述中句子之间的相识性度量,构建句子之间的相似性网络,然后发现句子相似性网络中的句子社区,实现对句子的聚类。每个句子社区蕴含一个软件特征,包含了所有潜在描述该软件特征的文本句子。所发现的句子社区可能存在重叠的句子,这些重叠句子同时涉及多个句子社区所蕴含的软件特征。进一步,为了帮助人们更好地理解句子社区所蕴含的特征,所提方法设计了相应的算法来从所有句子社区中依次选择熵最小的社区,并从所选社区中挑选最有代表性的、且其他社区还未选择的句子来作为一个社区所蕴含特征的描述符。文中爬取 Softpedia.com 网站的软件产品文本描述信息作为实验数据。实验结果表明,所提方法与现有代表性方法相比在准确性与时间方面具有更好的表现。

关键词 自然语言,特征提取,重叠社区发现
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/jsjkx.181001856

Software Feature Extraction Method Based on Overlapping Community Detection

LIU Chun ZHANG Guo-liang
(School of Computer and Information Engineering, Henan University, Kaifeng, Henan 475001, China)

Abstract Extracting software features from natural language of product descriptions has gained a lot of attentions in recent years. In light that the sentences in the descriptions can describe the semantics of software features more precisely and one sentence may be concerned about more than one software feature, this paper proposed a feature identification method by detecting the overlapping clusters of these sentences in the natural language descriptions. Based on the overlapping community detection algorithm (LMF), the proposed method defines a metric to measure the similarity between each pair of sentences in the descriptions, builds a sentence similarity network accordingly, and then detects the overlapping sentence communities in such network. Each sentence community is a cluster which implies one software feature, and contains all the sentences potentially describing the implied feature. Further, in order to help people better understand the characteristics of sentence communities, the proposed method designs corresponding algorithms to select the communities with the lowest entropy from all sentence communities in turn, and to select the most representative sentences from the selected communities that have not been selected by other communities as descriptors of the features contained in the community. The natural language product descriptions from Soft pedia.com were crawled as experimental data. Experimental results show that the proposed method has better performance in accuracy and time consumption.

Keywords Natural language, Feature extraction, Overlapping community detection

1 引言

实际上包括软件系统在内的每种产品往往都会包含一组特征。这些特征一般指产品所具有的显著特征或具有区分作用的、用户可见的特点[1],比如产品的功能或者外形所具有的特点。根据卡诺模型[2],一种产品所具有的特征可以分为 3 类:基本型、期望型和魅力型。基本型特征是用户认为产品理所当然应该具有的;期望型特征是用户在选择产品时所关注

到稿日期:2018-10-08 返修日期:2019-03-25

刘 春(1982—),男,副教授,主要研究方向为软件需求工程;张国良(1993—),男,硕士生,主要研究方向为软件需求工程,E-mail:glzhang@vip.henu.edu.cn(通信作者)。

的,它们在产品中实现得越好,用户就会越满意;魅力型特征是用户从未预料到的,如果产品实现了该特征,用户将会非常满意。显然,基本型和期望型特征一般是一个产品应该具备的,也是同一个领域内相关产品所共有的特征。在这种情况下,对于想要进入一个现有领域的产品设计者来说,分析市场上同一领域内产品的共有特征,了解必须继承和改进的特征,无疑具有重要意义。

对于诸如移动设备之类的大众便宜产品,可以通过购买产品和人工分析的方式来了解领域内产品的共有特征。然而,对于那些昂贵、依赖于特定操作环境以及领域内相似产品众多的软件系统来说,人工分析将费时费力[3]。此时,利用自然语言处理技术,分析各个软件产品的文本描述信息,自动提取产品特征,将具有重要价值。一方面,这些产品的文本描述往往会详细阐述产品的特征,以吸引用户选择该产品。另一方面,可以利用爬虫很容易地从 Softpedia.com 等在线网站或诸如 Google Play 的 App 商店上获取一个领域内的诸多软件系统的文本描述信息。

由于文本特征提取技术不仅可以应用于软件特征分析[4-5]和软件产品线建立[6],还可以应用于从各种产品的评论中提取有效建议[7-8],因此近年来从公开文本信息中自动提取软件特征已经获得了大量关注。传统的文本特征提取方法主要通过提取单个关键词来表示各个特征[9-10],但是单个关键词的表达能力有限,只能承载简略的特征信息,并可能造成歧义,比如安全类软件的文本描述中,"remove"可能表达移除病毒,也可能表达移除某个感染的文件。在这种情况下,当前的文本特征提取研究主要关注的是如何从文本中提取词组或者句子来表达特征,因为一个词组或者句子要比单个关键词能够表达更多的特征信息。

在基于词组和句子的文本特征提取方面,当前的典型工作大致可以分为两类。一类认为特征蕴含于文本描述的句子中,因此该类工作是将文本信息进行分句和句子聚类,每一类句子表达某一个特征,进而从聚类中挑选代表性的句子来表示特征[4]。另一类认为特征可以通过词组来表达,因此该类工作根据句子的词性结构等信息选择特征描述的常见模板,从文本描述中筛选符合模板、由不同词性单词所构成的词组来表示特征[11],或者直接考虑词语搭配,从文本描述中提取常见的固定长度的词语搭配词组来表示特征[12]。由于目前缺乏比较上述两类工作的优劣的实践研究,并且考虑到一个完整的句子更能准确表达一个特征的含义,本文将主要关注如何基于句子来从文本中提取特征。

基于句子的文本特征提取方法的代表性工作是由 Hariri 等提出的递增扩散聚类方法[4]。该方法首先基于词袋以及 TF-IDF 模型[13]将软件产品描述中的每个句子表示成向量,然后考虑到软件产品描述中的一个句子可能涉及多个软件特征,该方法迭代地对句子进行聚类,每次挑选出一个最优聚簇,识别该聚簇的关键词,并将该词从所有的句子向量中删除,最后开始新的迭代。聚簇中具有代表性的句子将被选出,用于表示一个特征。一方面,由于实际中每个句子并不是很长,且句子中每个单词往往只出现一次,这将导致单词的 TF 值一般为 1;另一方面,由于文本描述涉及的单词数量众多,这将导致高维稀疏的句子向量以及较高的算法时间复杂度。为此,本文在该方法的基础上,考虑到文本描述中每个句子可能与多个特征相关,提出了基于重叠社区发现算法[14]的文本特征检测技术,来从公开的软件产品描述中提取软件特征。

本文爬取 Softpedia.com 网站上的软件产品描述信息作为实验数据。实验结果表明,本文所提出的方法比 Hariri 等提出的递增扩散聚类方法在准确性和性能方面都有更好的表现。本文第 2 节简要介绍本文所采用的重叠社区发现算法;第 3 节详细介绍了本文方法;第 4 节描述了实验设计、评估结果以及可能存在的对实验结果有效性的影响因素;第 5 节描述了相关工作;最后总结全文并展望未来。

2 重叠社区发现算法

社区发现的目标是在以社交网络为代表的复杂网络中发现网络的模块结构特性,一个社区代表了一个节点的聚簇。由于实际中不同社区之间可能共享某些节点,重叠社区发现算法的目标是能够在复杂网络中发现重叠的社区。

重叠社区发现算法的一个典型代表是 LMF 方法[14]。LMF 的基本思想是迭代地从不同的种子节点出发,以贪婪的方式检测社区。LMF 在一次迭代过程中从不属于任何社区的节点中随机选择种子,并且为了确定某一节点是否属于当前社区,使用式(1)来度量社区中节点相对于社区的适应度之和。

$$f=\frac{k_{\text{in}}}{(k_{\text{in}}+k_{\text{out}})^{\alpha}} \tag{1}$$

其中,k_{in} 和 k_{out} 是社区中节点的内部度(与社区内部节点的连接数量)和外部度(与社区外部节点的连接数量);α 是正实数,用于控制社区大小,α 越小,社区越大。

基于该度量方法,LMF 在检测过程中迭代地评估已经在社区中的节点的所有邻居节点,找到其中加入社区时能够增加社区适应度的节点,并且选择其中的最佳贡献者加入到社区中。一旦添加新节点,LMF 再次评估社区中已经存在的节点对社区适应度的贡献,并移除贡献度为负的节点,即排除该节点能够增加社区适应度。当所有邻居的贡献度为负时,对一个社区的贪婪发现就此结束。当网络中的所有节点都已分配社区时,整个社区发现过程将停止。

3 基于重叠社区发现的特征提取

本文目标是从诸如 Softpedia.com 网站上的产品描述文本信息中,有效地提取某个领域内产品的共有特征。围绕该目标,本文提出了基于重叠社区发现算法 LMF 来从文本中提取特征的方法。该方法的基本思路是将文本描述中的每个句子视为复杂网络中的一个节点,将句子之间的相似性视为网络中连接节点的边,进而构造句子相似性网络,然后在 LMF 算法的基础上进行改进,来发现句子相似性网络中重叠的句子社区,每个句子社区包含了与某个软件特征相关的所有句子。在这些描述同一个特征的句子中,选择其中最有代

表性的句子作为特征的描述符。本文方法的具体步骤如下。

3.1 预处理

本文方法的输入是从 Softpedia.com 网站上爬取的通过自然语言描述的产品文本信息。该文本描述信息包括有关产品功能的概述和产品特征列表,其中 Softpedia.com 网站上某些产品的特征列表是缺失的。我们首先从上述文本描述信息中提取文本描述中的每个句子,然后应用 Python NLTK 软件包将每个句子解析为一组单词,删除停用词,并将剩余的词进行词干化。产品描述中的每个句子最终变为一个包含若干关键词的单词集合。

由于文本描述中高频词往往不能有效区分描述不同特征的句子,而低频词往往是噪声数据,因此我们删除了每个句子中 $sf_i<\delta\times N$ 的单词和 $sf_i>(1-\delta)\times N$ 的单词。其中,sf_i 为包含单词 i 的句子数,N 是句子的总数,δ 是可以调节的阈值。另外,由于一些产品描述的书写可能比较随意,比如缺少合理的句子分隔符,导致所提取的某些句子可能过长或者过短,为此在预处理过程中进一步过滤了仅具有一个单词和太多单词(单词数>70)的句子。

3.2 构建句子相似性网络

为了应用社区发现算法来检测文本描述中不同句子之间存在的句子社区,需要构造句子相似性网络。在该网络中,节点实际上是由单词集合所代表的句子,节点之间的边代表了句子之间的相似性。考虑到实际中两个句子重叠的单词越多,重叠的单词越重要,且这些重叠的单词在句子中具有越大的影响力,句子之间的相似性就越大,本文采用式(2)来度量句子相似性。

$$sim(s_i,s_j)=\frac{(\sum_{w_k\in s_i\cap s_j} idf(w_k))^2}{\sum_{w_k\in s_i} idf(w_k)\times\sum_{w_k\in s_j} idf(w_k)} \tag{2}$$

idf 是单词的逆文档频率[13]。本文通过将文本描述中的每个句子视为文档来计算单词的 IDF 值,并通过该值来衡量单词的影响力。

基于式(1),如果每对句子共享一些单词,那么构造的相似性网络是全连通的网络。在这种情况下,所构造的句子相似性网络将具有大量的连接边,比如具有 n 个节点的相似性网络将具有 $n(n-1)/2$ 条边,此时社区发现的过程将耗费大量的时间。并且由于相似性较小的两个句子属于一个社区的可能性较小,因此为了减少相似性网络中边的数量,本文将相似性网络中共享单词少于两个的句子之间的边删除。这一策略同时也有助于隔离相似性网络中属于噪声的句子。

3.3 发现句子社区

由于句子相似性网络是加权网络,我们改造式(1)来度量社区的节点适应度之和。

$$f=\frac{\sum_{e\in E_{in}} 2w_e}{(\sum_{e\in E_{in}} 2w_e+\sum_{e\in E_{out}} w_e)^{\alpha}} \tag{3}$$

其中,E_{in} 是社区中的节点之间边的集合,E_{out} 是当前社区节点与社区之外节点之间边的集合,w_e 是每个边的权重,即句子之间的相似性。通过实验发现,将 α 的值设置在[1,2.5]范围内是合适的。当 $\alpha<1$ 时,由于每个社区的规模增加,检测过程会变得非常缓慢。

此外,LMF 在社区发现过程中随机选择一个还未分配社区的节点作为种子,这种方式有可能会受到噪声数据的影响。本文在基于 LMF 的句子社区发现过程中,选择当前与权重最大的边相连且还未分配社区的节点作为种子节点,因为通过具有最大权重的边连接的两个句子更有可能属于一个新的社区,这可以减弱噪声数据的影响。而噪声数据通常没有边或者只有权重较小的边将它们与其他节点连接。

算法 1 给出了通过改进 LMF 算法来发现句子社区的详细过程。最初句子相似性网络中的所有节点都是种子的候选者,一旦找到种子,算法将基于 LMF 的贪婪策略检测新社区的所有成员,即算法 1 中的第 8 行。在检测到社区之后,将从候选种子集中删除其所有成员,即算法 1 中的第 11 行,这意味着它们不能被选为种子。在开始贪婪检测之前删除种子,即算法 1 中的第 6 行,以避免算法 1 中的 while 循环陷入死循环,这是因为在第 8 行的社区发现过程中,种子节点可能从当前社区被剔除,在这种情况下,该种子节点将在下一个 while 循环被重新选中,导致 while 循环陷入死循环。

经过多次迭代之后,候选种子的数量将减少。当没有种子可选择时,整个检测过程,即算法 1 中的 while 循环将停止。但是候选种子集合,即算法 1 中的集合 U_{node},在 while 循环结束时可能不为空。如果句子相似性网络中存在一些孤立的节点,即噪声数据,则将它们留在 U_{node} 集合中。

进一步地,即使选择了不同的种子节点来探索句子社区,基于 LMF 算法的社区节点发现过程仍然可能返回相同的社区。因此,一旦检测到新社区,算法检查它是否已经存在,即算法 1 中的第 9 行。此外,由于检测到的社区有重叠情况,它们可能比较相似,因此有必要合并最相似的社区。假设 $C=\{c_1,c_2,\cdots,c_i\}$ 是检测到的句子社区集合,本文使用式(4)来计算两个社区之间的相似性,并在它们的相似性大于指定阈值时将它们合并。

算法 1　基于 LMF 的句子社区发现算法

```
Data: ssn: sentence similarity network; U_node: the set of nodes in the
      network
Result: C_comm: the set of identified communities
1. begin
2.    while True do
3.    c_comm ← Ø //a new community;
4.    seed ← findSeed(U_node)//find the node which is linked to the
      edge with maximum weight from U_node;
5.    if seed is found then
6.       U_node ← U_node − {seed};
7.       c_comm ← C_comm ∪ {seed};
8.       c_comm ← findCommMembers(ssn, C_comm);
9.          if c_comm ∉ C_comm then
10.      C_comm ← C_comm ∪ {c_comm};
11.      U_node ← U_node − c_comm;
12.   end
13.   else
```

```
14.     break//break from while loop;
15.   end
16.   end
17.   C_comm←mergeSimComm(C_comm);
18.   return C_comm;
19. end
```

$$sim(c_i,c_j)=\frac{|c_i\bigcap c_j|}{\min(|c_i|,|c_j|)} \tag{4}$$

3.4 选择特征描述符

为了从句子社区中选择最有代表性的句子作为社区所蕴含特征的描述符，本文提出的方法如算法 2 所示。

算法 2 选择特征描述符的算法

```
Data: C_comm: The set of sentence communities
Result: D: The set of feature descriptors
1. begin
2.   k←|C_comm|//number of communities;
3.     for i=1;i<k;i++ do
4.     //compute the entropy of each community;
5.     E_comm_entropy←computeCommEntrope(C_comm);
6.     c_min_enyroy←selectComm(C_comm,E_comm_entropy);
7.       //treat each community as a document to compute tfidf values of the words in c_min_enyroy;
8.     V_word_tfidf←TFIDFComputing(C_comm,c_min_entroy);
9.     V_sent←sentToVector(c_min_entroy,V_word_tfidf);
10.    centroid←computeCentroid(V_sentences);
11.    descriptor←selectClosetSent(c_min_entroy,centroid,V_sent);
12.      D←D∪{descriptor};
13.      C_comm←C_comm−{c_min_entroy};
14.  for each comm∈C_comm do
15.      comm←comm−c_min_entroy;
16.      end
17.    end
18.    return D
19. end
```

首先，由于所发现的各个句子社区之间可能彼此重叠，为了避免不同的句子社区选择同一个句子作为特征描述符，每次挑选还未选择特征描述符且具有最小熵的句子社区（算法 2 中的第 5－6 行）进行特征描述符的选择。每个社区的熵计算如下：

$$H(c_k)=-\sum_{s_i\in c_k}\frac{1}{n_{s_i}}\log(\frac{1}{n_{s_i}}) \tag{5}$$

其中，n_{s_i} 是包括句子 s_i 的句子社区的数量，$1/n_{s_i}$ 表示句子 s_i 属于一个特定社区的概率。基于 Beil 等的工作[15]，式(5)可以衡量社区与其他社区之间的重叠度。具有最小熵的社区意味着该社区与其他社区具有较小的重叠部分，社区内聚度高，社区中的句子作为其他社区的特征描述符的概率低，因此该社区应具有高优先级来进行特征描述符的选择。

选择当前具有最小熵的社区后，计算所选社区中每个单词的 *TF-IDF* 值[13]（见算法 2 的第 8 行）。此时，我们将还未选择特征描述符的每个社区视为一个文档，即每个社区是一个包含社区中所有句子的文档。在这种情况下，社区中经常出现的且能够标识该社区的词语将具有更高的 *TF-IDF* 值，因此对特征描述符的选择具有更大的影响。这也可以避免将每个句子作为文档来计算单词 *TF-IDF* 值时，句子中大多数单词的 *TF* 值为 1 的情况。在计算所选社区中的单词 *TF-IDF* 值之后，将选定社区中的每个句子转换为 *TF-IDF* 向量（见算法 2 中的第 9 行），并选择最接近社区质心的句子作为特征描述符（见算法 2 中的第 10－11 行）。最后，从社区集合（见算法 2 中的第 13－15 行）中删除所选社区及其所有句子。这意味着所选社区与其余社区重叠的句子将从其它社区中删除（见算法 2 中的第 15 行）。该步骤旨在消除它们对剩余社区特征描述符选择的影响，并避免在不同的社区中选择相同的描述符。上述过程迭代运行，直到为每个社区选择出特征描述符。

4 实验

4.1 实验数据

我们于 2017 年 12 月从 Softpedia.com 上抓取了防病毒和多媒体两类软件产品的文本描述信息。经手动删除重复的软件产品和缺少描述的软件产品后，获得了 610 个防病毒软件产品和 629 个多媒体软件产品。爬取的文本信息包括软件产品的描述和产品所具有的特征列表，其中某些产品未提供特征列表。

最终我们从 610 种防病毒产品中获得 10 079 个有效句子，从 629 种多媒体产品中获得 10 220 个有效句子。为了评估本文的文本特征提取方法，需要人工来对这些句子进行聚类，以了解其中隐含的软件特征。然而，通过人工的方式对这么多句子进行聚类并不容易，因此，我们从爬取的数据中随机选择部分软件产品的文本描述来进行实验。如表 1 所列，分别随机选择了 50 个防病毒软件产品和多媒体软件产品，防病毒软件产品的文本描述总共包含了 1038 个句子，多媒体软件产品的文本描述则总共包含了 989 个句子。针对每个数据集，首先由两名研究员分别进行独立分析，标识噪声数据，识别表达特征的句子的合理聚簇，然后一起讨论他们的结果以产生最终作为答案的聚类结果。表 1 列出了每个数据集的聚类簇数。

表 1 实验数据集

Table 1 Experiment datasets

Dataset	Num. valid sent.	Num. clusters
Antivirus-50	1038	23
Multimedia-50	989	22

4.2 本文特征提取方法的准确度分析

为了分析本文特征提取方法的准确度，选择当前基于句子聚类的典型特征提取方法递增扩散聚类（IDC）[4]进行比较分析。我们选择了纯度和标准互信息（NMI）两个指标，这两个指标也被 IDC 方法用于与 LDA[16]，SphericalK-means[17]，K-means 和 FuzzyK-means[18]等典型聚类方法进行比较。

(1)纯度。纯度可以通过将提取的簇与答案簇进行比较来进行计算。设 w_k 是手动分析生成的答案簇集合,c_j 是提取的句子集合,纯度的计算式如下:

$$purity(W,C)=\frac{1}{N}\sum_j \max_k |w_k \cap c_j| \tag{6}$$

对于每个提取的社区,首先找到与其共享最多成员的答案簇,然后计算正确分配的成员的总数,并将该数量除以提取的社区中的成员总数来计算纯度。纯度可以精确地评估特征提取算法的准确性,其取值范围为 0 到 1,其值越高表示性能越好。此外,当提取的社区中的成员较少时,纯度将表现出更高的值。在极端情况下,当提取的社区数量等于这些社区中的成员数量时,纯度的最大值为 1,因此 NMI 经常作为纯度的补充。

(2)NMI。NMI 是标识两个对象之间相关强度的度量。给定提取的句子社区集合 C 和答案簇集合 W,NMI 的计算式如下:

$$NMI(W,C)=\frac{I(W,C)}{(H(W)+H(C))/2} \tag{7}$$

其中,$I(W,C)$ 表示提取的社区集和答案集之间的互信息,$H(W)$ 和 $H(C)$ 分别代表两个集合的熵。其计算方法如下:

$$I(W,C)=\sum_k \sum_j \frac{|w_k \cap c_j|}{N} \log\left(\frac{N(|w_k \cap c_j|)}{|w_k||c_j|}\right) \tag{8}$$

$$H(W)=-\sum_k \frac{|w_k|}{N} \log\left(\frac{|w_k|}{N}\right) \tag{9}$$

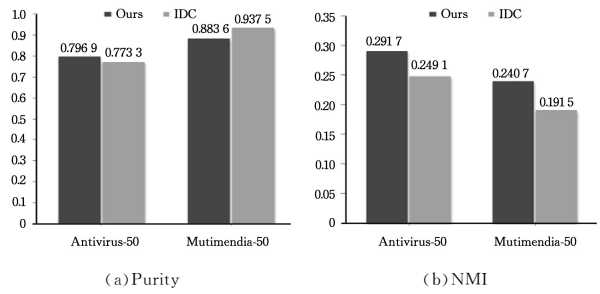
其中,N 是数据集中句子的总数量。与纯度相似,NMI 值越高,准确性就越高。此外,IDC 旨在提取指定数目的句子集合,但本文方法没有指定要提取的句子社区数量,并且多个参数会影响到发现的社区数量。鉴于此,我们决定从发现的社区中选择与 IDC 相同数量的句子簇进行比较,以比较两种方法在提取相同数量的软件特征时的准确性。

为了选择一定数量的句子社区进行比较,本文通过计算句子社区权重 CW 来对提取的句子社区进行排序。考虑到一个句子社区的凝聚力越大,社区的规模越大,就越能代表一个软件特征,句子社区权重的计算方式如下:

$$CW(c_i)=\gamma \times cohesion(c_i)+(1-\gamma)\times \frac{|c_i|}{N} \tag{10}$$

$$cohesion(c_i)=\frac{1}{2|c_i|}\sum_{e_k \in c_i} \sum_{e_h \in c_i, e_h \neq e_k} similarity(e_k,e_h) \tag{11}$$

在应用表 1 中的两个数据集进行比较分析的过程中,由于不同的参数会影响两种方法的准确性,本文采用固定的步长来改变不同的参数值以尝试不同的参数值组合,从而找到纯度和 NMI 的最佳结果。图 1 给出了比较的结果。对于纯度,可以看出在数据集 Antivirus-50 上,本文方法与 IDC 相比表现略好,而 IDC 在数据集 Multimedia-50 上表现更好。但就 NMI 而言,本文方法在两个数据集上表现出了更好的准确性。其原因是 IDC 倾向于选择内聚更高的聚簇作为每次迭代的最佳聚簇,因此,IDC 提取的聚簇中将包含较少的句子,其在纯度方面表现更优。不同的是,本文采用的重叠社区发现方法往往在贪婪社区发现过程中将更多的句子包含在社区中,因此本文方法在 NMI 中具有更好的表现。



(a)Purity　　(b)NMI

图 1　算法准确性的比较

Fig. 1　Comparison results in accuracy

IDC 方法主要包含两个步骤:代表特征的 k 个聚簇的提取和对提取的聚簇进行补充等操作的后处理。在比较分析过程中发现,IDC 算法的时间开销主要耗费在基于 Spherical K-means[17] 的 K 个聚簇的提取上。而在此步骤中,IDC 只有一个参数会影响算法的时间性能,即每个聚簇的代表性关键词的最小 $TF\text{-}IDF$ 值。与此相似,本文方法的时间开销主要耗费在社区发现上。而在此过程中也只有一个参数会影响算法的时间性能,即能够控制社区大小的参数 α。因此,为了比较两种方法在时间方面的表现,我们固定两种方法的其他参数,然后在上述两个参数的合理取值范围内分别取 10 个参数值,然后记录两种方法的运行时间,最终取各种方法的平均运行时间进行比较。

本文在一台 3.3GB CPU 和 8GB 内存的 PC 机上进行实验,表 2 列出了在两个数据集上最终的时间性能的比较结果。从表 2 可以看出,本文方法在时间性能方面具有更好的表现。一方面,由于 IDC 方法基于 Spherical K-means 算法在迭代过程中实现句子聚类,其时间复杂度为 $O(n^3)$,而本文方法基于 LMF 的重叠社区发现算法,其时间复杂度为 $O(k*n^2)$,其中 k 为最终检测社区的个数。另一方面,本文采用重叠社区发现算法来提取表达特征的句子簇,在此过程中本文方法在理想情况下只需要遍历一次数据集,而 IDC 方法由于基于 Spherical K-means[17] 方法来进行句子聚类,需要多次遍历整个数据集,因此 IDC 方法的时间性能与数据集的规模严格相关。从表 2 还可以看出,IDC 方法在两个数据集中几乎耗费了同样的时间,这是因为两个数据集具有大致相同的规模。不同的是,本文方法的时间消耗不仅依赖数据集的规模,还依赖数据集本身所具有的特征。当数据集本身具有很高的聚集性时,就会得到一个稀疏的句子相似性网络,这时的句子社区发现过程将会耗费较短的时间,这也是本文方法在多媒体数据集上耗费更短时间的原因。这种多媒体数据集上所体现的稀疏特征也能够解释为什么两种方法在多媒体数据集上能表现出更好的纯度(见图 1)。

表 2　时间性能比较

Table 2　Comparison in time performance

(单位:s)

数据集	本文算法	IDC
Antivirus-50	1725	33661
Multimedia-50	892	32731

4.3 对特征描述符选择方法的评估

为了评估特征描述符的选择算法，基于数据集 Antivirus-50 和 Multimedia-50，本文首先通过算法 1 得到相应的句子社区（设定控制社区大小的关键参数 α 为 1），然后利用本文提出的特征描述符选择算法为每个社区选择特征描述符。之后，两名研究人员独立核实所选择的句子是否能代表每个社区所蕴含的特征，并计算相应的准确度，最后以两名研究人员给出的准确度的均值为最终的准确度。表 3 列出了实验中所发现的句子社区数量和本文特征描述符选择算法的准确度。

表 3 特征描述符选择算法的评估结果

Table 3 Evaluation results of feature descriptors selection algorithm

Dataset	Num. detected communities	*Accuracy*/%
Antivirus-50	68	67.7
Multimedia-50	82	65.2

4.4 有效性影响

虽然本文爬取了 Softpedia.com 上的数据并进行了相关实验，结果证实了本文方法的可行性，但仍然存在诸多潜在因素影响本文实验结果的有效性。

首先，虽然本文通过人工分析的方式对数据集中的句子进行了聚类，但是所应用的数据集中仍然可能存在噪声数据，经过人工分析所得到的答案聚簇也可能存在缺陷，因为人工进行句子聚类的过程不仅枯燥，而且费时费力，所有这些因素都可能影响本文实验结果的有效性。其次，由于需要通过人工分析来得到作为答案的句子聚簇，因此本文只选择了两个数据集，并且数据集的规模都较小，这也可能影响本文结果的有效性。

5 相关工作

本文中的研究工作实际上与领域分析[1]、需求重用[19]、需求挖掘[20]和特征模型挖掘[21]等方向的研究工作相关。本节将重点关注和概述与软件特征提取相关的研究工作。

如引言所述，传统的特征提取方法主要是基于关键词的方法，比如 MARK 方法通过提取评论中经常出现的负面关键词来发现评论中隐含的重要信息[10]；AR-Miner 方法则主要应用 LDA 等技术来对评论进行聚类，并获取每一类的关键词[11]。不同于基于关键词的方法，当前软件特征提取的研究工作主要关注于如何通过提取词组或者句子来更好地获得具有更完整语义的特征描述。

基于句子的方法假设每个句子描述一个或多个特征，因此这种方法倾向于通过不同的方法将文本描述中的句子聚到不同的类别中，每个句簇代表一个软件特征。在这方面，Hariri 等[4]提出使用 TF-IDF 向量空间模型（VSM）来表示每个句子，并提出一种递增扩散聚类算法来迭代地聚类文本描述中的句子。在一个迭代过程中，它首先采用 Spherical K-means 方法进行句子聚类，然后选择最佳聚簇，从聚簇中选择一个最有代表性的句子来对聚簇进行重命名，最后从所有句子中删除最佳聚簇的关键词，以消除它们对进一步聚类的影响。同样，Bakar 等[7]也使用 TF-IDF 和 VSM 模型将每个句子变成向量，但他们使用 SVD 方法得到聚簇。上述工作所提出的方法均需要提供或估计潜在的特征数量，即需要生成的句子聚簇的数量。例如，Hariri[4]提供了一个公式来估计潜在特征的数量。Yu 没有使用 TF-IDF 模型，而是采用 LDA 方法将句子转化为向量，然后应用层次聚类方法对句子进行聚类，以挖掘特征之间的语义结构[22]。

与基于句子的方法不同，基于词组的方法认为特征通常是以短语的形式表达的，即通过一组具有不同词性的词组来表达，因此这种方法倾向于首先定义词组的模板，然后从每个句子中提取这些词组，最终通过聚类合并相似的词组。在这方面，Vu 等[11]提出了一种基于短语的方法来从软件的用户评论中提取用户意见。为了获得短语的模板，他们首先从语料库中挖掘词性的序列模式。Liu 等[5]也提出了类似的特征提取方法。但不同的是，他们是从一个样本数据集中通过人工分析的方式获取短语模板。Johann 等[8]专注于提取和匹配软件描述和评论中的特征，以让设计人员知道用户提到了哪些功能特征，从而提出了 SAFE 算法。同样，他们也手动构建了几个词组模板和句型，并将它们统一应用于 App 描述和用户评论。为了识别软件特征并分析应用商店中软件特征的生命周期，Sarro 等[23]首先提取原始的特征模式。在他们的工作中，如果软件的 HTML 描述中与单词“包括，新，最新，密钥，免费，改进，下载，选项，功能”相关联，这样的描述将被保存为原始特征模式。该方面的另一个典型工作是 Guzman 等[12]的工作，不同于上述工作，他们不依赖于词组模板而是通过找到常见的词语搭配来提取用户评论中的软件特征。

基于词组或者短语的方法的准确性受文本描述的表达风格和语法正确性的影响。而在实际中，这些描述通常表达自由，并且可能在语法上有很多错误[10]。因此，获得准确的词组和短语模板并非易事。虽然通过找到单词搭配来提取特征并不需要词组模板，但是如何确定用于识别词语搭配的最小频率以及如何理解所识别的搭配仍然是问题。因此，本文遵循基于句子的方式来提取软件的特征，但与现有的基于句子的方法不同，本文采用基于重叠社区发现的方法，该方法与现有基于句子的文本特征典型方法相比在准确性和时间方面有更好的表现。

结束语 考虑到软件产品描述中的句子更能清晰地表达一个软件特征的含义，本文提出了一种基于重叠社区发现的软件特征提取方法。该方法能够自动地从公开的软件产品文本描述中挖掘所有隐含的软件特征，不需要用户设定潜在的软件特征的数目，并且实验结果表明该方法与现有相似的方法相比在准确性和时间消耗方面具有更好的表现。

虽然本文所提方法能够提取软件特征，但是在实验中发现实际中提取的软件特征可能很多。在这种情况下，如何将提取的特征分类为基本型、期望型和魅力型，是我们下一步需要解决的问题。

参 考 文 献

[1] KANG K. Feature-Oriented Domain Analysis (FODA) Feasibi-

lity Study[J]. Technical Report Software Engineering Institute Carnegie Mellon University,1990,4(4):206-207.

[2] BERGER C. Kano's methods for understanding customer-defined quality[J]. Center for Quality Management Journal,1993,2(4):3-36.

[3] FERRARI A,SPAGNOLO G O,DELL'ORLETTA F. Mining commonalities and variabilities from natural language documents [C]// International Software Product Line Conference. New York:ACM,2013:116-120.

[4] HARIRI N,CASTROHERRERA C,MIRAKHORLI M,et al. Supporting Domain Analysis through Mining and Recommending Features from Online Product Listings[J]. IEEE Transactions on Software Engineering,2013,39(12):1736-1752.

[5] LIU Y,LIU L,LIU H,et al. Mining domain knowledge from app descriptions[J]. Journal of Systems & Software, 2017, 1(23):1-19.

[6] BAKAR N H,KASIRUN Z M,SALLEH N. Feature extraction approaches from natural language requirements for reuse in software product lines:A systematic literature review[J]. Journal of Systems & Software,2015,106(C):132-149.

[7] BAKAR N H,KASIRUN Z M,SALLEH N,et al. Extracting features from online software reviews to aid requirements reuse [J]. Applied Soft Computing,2016,49:1297-1315.

[8] JOHANN T,STANIK C,ALIREZA M A B,et al. SAFE:A Simple Approach for Feature Extraction from App Descriptions and App Reviews[C]// Requirements Engineering Conference. IEEE,2017:21-30.

[9] CHEN N,LIN J,HOI S C H,et al. AR-miner:mining informative reviews for developers from mobile app marketplace[C]// International Conference on Software Engineering. ACM,2014:767-778.

[10] VU P M,NGUYEN T T,PHAM H V,et al. Mining User Opinions in Mobile App Reviews:A Keyword-Based Approach (T) [J]. Computer Science,2015,9(13):749-759.

[11] VU P M,PHAM H V,NGUYEN T T,et al. Phrase-based extraction of user opinions in mobile app reviews[C]// IEEE/ACM International Conference on Automated Software Engineering. IEEE,2016:726-731.

[12] GUZMAN E,MAALEJ W. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews[C]// Requirements Engineering Conference. IEEE,2014:153-162.

[13] MOGOTSI I C,CHRISTOPHER D. Manning,Prabhakar Raghavan,and Hinrich Schütze:Introduction to information retrieval[J]. Information Retrieval,2010,13(2):192-195.

[14] LANCICHINETTI A,FORTUNATO S,KERTéSZ J. Detecting the overlapping and hierarchical community structure of complex networks[J]. New Journal of Physics,2008,11(3):19-44.

[15] BEIL F,ESTER M,XU X. Frequent term-based text clustering [C]// Eighth International Conference on Knowledge Discovery and Data Mining. ACM,2002:436-442.

[16] BLEI D M,NG A Y,JORDAN M I. Latent dirichlet allocation [J]. Journal of Machine Learning Research,2003,3:993-1022.

[17] DHILLON I S,MODHA D S. Concept Decompositions for Large Sparse Text Data Using Clustering[J]. Machine Learning,2000,42(1/2).

[18] BEZDEK J C. Pattern Recognition with Fuzzy Objective Function Algorithms[J]. Advanced Applications in Pattern Recognition,1981,22(1171):203-239.

[19] NIU N,SAVOLAINEN J,NIU Z,et al. A Systems Approach to Product Line Requirements Reuse[J]. IEEE Systems Journal, 2014,8(3):827-836.

[20] LIAN X,CLELAND-HUANG J,ZHANG L. Mining Associations Between Quality Concerns and Functional Requirements [C]// Requirements Engineering Conference. IEEE,2017:292-301.

[21] MEFTEH M,BOUASSIDA N,BENABDALLAH H. Mining Feature Models from Functional Requirements[J]. Computer Journal,2016,59(12).

[22] YU Y,WANG H,YIN G,et al. Mining and recommending software features across multiple web repositories[C]// Asia-Pacific Symposium on Internetware. ACM,2013:1-9.

[23] SARRO F,ALSUBAIHIN A A,HARMAN M,et al. Feature lifecycles as they spread,migrate,remain,and die in App Stores [C]// Requirements Engineering Conference. IEEE,2015:76-85.