面向用户查询意图的句子相似度分层计算

李景玉 张仰森 陈若愚
(北京信息科技大学智能处理信息研究所 北京 100192)

摘 要 为进一步提高句子相似度算法的准确性并提升其在复杂语境中的适用性,综合编辑距离、关键词及同义词语义方法,设计出面向用户查询意图的句子相似度分层算法。在充分分析实验数据用途的基础上,研究数据的特征分布,借助自然标注将句子相似度计算建模为多层次优化问题。经仿真实验证实该算法是有效的,F值可达到0.6019。

关键词 句子相似度计算,语义一致,编辑距离,关键词特征,用户查询意图
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.050

User Query Intention Oriented Hierarchical Sentence Similarity Computation

LI Jing-yu ZHANG Yang-sen CHEN Ruo-yu
(Institute of Intelligence Information Processing, Beijing Information Science and Technology University, Beijing 100192, China)

Abstract In order to improve the accuracy of sentence similarity computation algorithm and further enhance its applicability in complex context, a hierarchical sentence similarity algorithm for user-oriented query intention was designed, integrating technologies such as edit distance, keyword and synonyms semantic method, and natural annotation. With thorough analyzing of the experimental data and its feature distribution, a multi-level optimization strategy was put forward. The experimental results confirm the algorithm in this paper is effective and achieves F-value of 0.6019.

Keywords Sentence similarity computation, Semantic conformity, Edit distance, Keyword feature, User query intention

1 引言

相似度计算是自然语言处理领域的基础工作,应用背景广泛。根据处理对象的不同,它可以分为词相似度计算、句子相似度计算以及篇章相似度计算。其中句子相似度计算在信息检索、机器翻译、问答系统以及自动文摘等应用领域中的效能影响着系统的整体性能,因此目前仍有众多学者热衷于不断改进句子相似度的计算方法。

目前的句子相似度计算方法主要有基于编辑距离的句子相似度计算方法[1]、基于词的句子相似度计算方法[2-4]、基于本体词典或知网语义的句子相似度计算方法[5,6]、基于语义依存的句子相似度计算方法[7,8]、基于框架语义分析的句子相似度计算方法[9]等。编辑距离代表一个字符串转换成另一个字符串所需要的最小编辑操作次数,文献[1]基于编辑距离思想,以及汉语中单个字往往不具备意义的特点,提出改进的编辑距离计算句子相似度算法,它以词为单位计算编辑距离,同时融入Hownet和《同义词词林》两个语义资源。但是,由于目前中文分词的效果并不理想,不同的语境下一个词有时可以识别出来,有时却不能,分词的不准确势必会带来编辑距离结果的误差。文献[2]从词形、词序两方面计算句子相似度;文献[3]在文献[2]的基础上对关键词的抽取方法进行改进,考虑了同义词的情况,其准确率有所提高,但是对于长度较长、结构较复杂的句子效果仍不理想。

针对上述问题,本文将数据分为3类,即构成句子的字符基本一致、关键词基本一致、句中关键词相差较大,采用分层次方法计算句子相似度,对不同类别的数据提出不同算法,具体如下:

1. 提出去除句末标点的编辑距离句子相似度算法,以字为处理单元快速识别基本一致的句子,从而克服了中文分词效果不理想的问题。

2. 统计文本中的关键词、同义词和否定词并分析其基本特征,据此提出基于关键词特征和语义特征的句子相似度算法,识别关键词基本一致的句子。

3. 在将句子相似度算法应用到信息检索时,重点分析用户查询请求和用户真实意图之间的关系,提出面向用户意图的句子相似度算法,一并解决关键词差异大、长度长、句子结构复杂等问题。

2 句子相似度计算

句子相似度是指两个句子的语义匹配程度,其结果值为[0,1]之间的一个实数。在实际应用中往往设定一个相似度阈值来判断句子语义是否一致。例如,在问答系统中给定一

到稿日期:2014-01-23 返修日期:2014-04-19 本文受国家自然科学基金(61070119,61370139),北京市属高等学校创新团队建设与教师职业发展计划项目(IDHT20130519),北京市教委专项(PXM2013_014224_000042,PXM2014_014224_000067)资助。
李景玉(1989—),女,硕士生,主要研究方向为中文信息处理、数据挖掘、机器学习,E-mail:lijingyu_emily@126.com;张仰森(1962—),男,博士后,教授,主要研究方向为中文信息处理、人工智能、Web内容安全、智能立体仓储;陈若愚(1982—),男,博士,讲师,主要研究方向为面向服务的计算、自然语言处理,E-mail:chenruoyu@bistu.edu.cn。

个相似度阈值，当用户提问和数据库中存储的问题句子相似度大于给定相似度阈值时，则将该数据库中存储的问题认为是用户提问的真正意图，即用户提问和数据库中存储的问题语义一致，从而给出用户所需的答案。为简便起见，本文将句子相似度计算结果定义为是布尔型，分别为语义一致和语义不一致两种，也即句子相似和句子不相似。

本文句子相似度分层计算流程如图 1 所示。

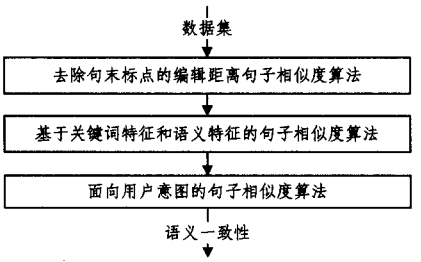

图 1　本文句子相似度分层计算流程

分层计算包括 3 个步骤，分别采用 3 种算法：去除句末标点的编辑距离句子相似度算法、基于关键词特征和语义特征的句子相似度算法、面向用户意图的句子相似度算法。采用 3 种不同的算法来处理不同类型的数据，从而逐步提升本方案的实用价值。

待处理文本集总是存在 3 类句子：构成句子的字符基本一致、构成句子的关键词基本一致、构成句子的关键词相差较大。经分析发现：1）在字符基本一致的句子中，句末的标点基本不具备意义，通过采用去除句末标点的编辑距离算法可以较准确地计算出这类数据的句子相似度；2）针对关键词基本一致的句子，采用基于关键词特征和语义特征的句子相似度算法，这不但解决了关键词在句中词序相对不同的问题，而且还将语义特征融入计算过程；3）根据构成句子的关键词相差较大的数据表述特点，以及具体应用环境，采用面向用户意图的句子相似度算法挖掘语义一致的句子。

本节余下内容将具体阐述上述 3 个方面。

2.1　去除句末标点的编辑距离句子相似度算法

编辑距离，也称 Levenshtein distance，是常用的句子相似度计算方法之一。通过计算一个句子转换成另一个句子的最小编辑操作数，来判定两个句子是否相似。其中编辑操作包括“删除”、“插入”和“替换”。若两个句子的编辑距离越小，则两个句子越相似。当编辑距离等于 0 时，说明两个句子的字符完全相同，语义一致。

经统计发现，人们在用自然语言描述问题时往往在句末加上标点符号，以表达某种语气。但是句末标点对于句子语义一致的判别是不具有意义的，甚至可能是杂音。例如，“母爱是什么”和“母爱是什么?”两个句子，第二个句子的句末标点问号不能决定两个句子的语义是否一致，它仅能说明第一个句子是陈述句，而第二个句子是疑问句。

基于以上观点，提出去除句末标点的编辑距离句子相似度算法。该方法能够识别出字符基本一致的句子是语义一致的，同时该方法采用字符为单位计算编辑距离可以忽略中文分词不准确的问题。因为人们在描述问题时往往偏向使用一些无关真正需求的介词，或者对于同一事物存在不同表述，所以将编辑距离小于等于 2 的句子对视为语义一致，即允许有两次或两次以下的编辑操作次数。例如，“玉兰花”和“关于玉兰花?”的编辑距离为 2，则判定两个句子是语义一致的。

2.2　基于关键词特征和语义特征的句子相似度算法

编辑距离算法的提出，最早是为了解决西文字符匹配问题，而在汉语中一个字往往不具有意义，词才是汉语中最小的能够独立运用的语言单位。例如，构成词“苹果”的两个字“苹”和“果”，任何一个都不能反映出“苹果”的含义。基于上述观点，可以将编辑距离算法改进成以词为单位的编辑距离算法。然而，不论是编辑距离算法，还是以词为单位的编辑距离算法，在计算句子相似度时其计算对象都是一个序列，这说明编辑距离算法的特性之一是在计算句子相似度时考虑了字或词语的顺序。但往往两个句子中能够代表其含义的关键词相对顺序是不同的。例如，“体重与身高的算法”和“身高与体重的算法”两个句子语义一致，但是句子中的关键词“体重”和“身高”的相对位置不同，若采用以词为单位的编辑距离算法，则需要至少 2 次的编辑操作，两个句子才能表述一致。而从语义角度来说，两个句子的“语义编辑距离”应该为 0。

在计算句子相似度时，只是单纯地比较两个句子所含有的关键词集合是否相等，并不能够表达其在语义上的一致性。例如，“一什么列车”和“一什么火车”两个句子是语义一致的，但是句子形成的关键词集合并不相等。剩余关键词分别为“列车”和“火车”，从字符匹配角度来说并不相同，而从语义角度上讲，“列车”和“火车”是同义词，是语义一致的。经过统计实验数据发现，在 800 对句子中，包含同义词的句子对有 105 对，其中 30 对是语义一致的，约占含有同义词句子对总数的 28.6%，因此可以得出语义特征对于句子相似度计算而言是必要的。

基于以上观点提出基于关键词特征和语义特征的句子相似度算法。该算法采用关键词特征，将句子转换成关键词列表形式，再兼顾同义词特征、否定词特征，总结分析经过对比消解后的剩余关键词词数与已标注语义一致性之间呈现的规律，最后判定符合规律的句子是否为语义一致。其中，剩余关键词是指一个句子经过分词、去除停用词、修改错别字、大写字母转换成小写字母、中文数字转换成阿拉伯数字、关键词对比消减、同义词对比消减和否定词对比消减后余留的关键词集合。

设 S 为句子的集合，$U\subset S$ 为用户提问的集合，$Q\subset S$ 为数据库中问题的集合，K 是词的集合，任一句子 $s_i\in S$ 的关键词集合 $K_{s_i}\subset K$，定义函数 $f:S\times S\to 2^K$ 表示两个句子经过关键词消减后得到的剩余关键词列表：

$$f(s_1,s_2)=\{i\mid i\in K_{s_1}\text{ 且 }i\notin K_{s_2}\}$$

算法 1　剩余关键词列表的计算

输入：句子 s_1，句子 s_2

输出：经过关键词消解后，存在于句子 s_1 中且不存在于句子 s_2 中的剩余关键词集

1. List k_1=Seg(s_1);
2. List k_2=Seg(s_2);//Seg()为分词函数
3. k_1=Stopwords(k_1);
4. k_2=Stopwords(k_2);//Stopwords()为去除停用词函数
5. k_1=MisusedWords(k_1);
6. k_2=MisusedWords(k_2);//MisusedWords()为修改错别字函数
7. k_1=ConvertToLower(k_1);
8. k_2=ConvertToLower(k_2);// ConvertToLower()为大写字母转换成小写字母函数
9. k_1=ConvertToNumber(k_1);

```
10. k2=ConvertToNumber(k2);//ConvertToNumber()为中文数字转换成阿拉伯数字函数
11. for each word1 in list k1//关键词对比消减
12. for each word2 in list k2
13.         if(word1== word2)
14.                 {
15.                 k1.Remove(word1);
16.                 k2.Remove(word2);
17.                 }
18.     else if(IsTongyici(word1,word2)) //如果两个关键词为同义词
19. {
20.                 k1.Remove(word1);
21.                 k2.Remove(word2);
22. }
23.         else if(IsFoudingci(word1) && IsFoudingci(word2))
            //如果两个关键词为否定词
24. {
25.                 k1.Remove(word1);
26.                 k2.Remove(word2);
27. }
28. return k1,k2.
```

表 1　剩余关键词总词数与句子对语义一致性之间的关系分析

句子2剩余关键词总数 \ 句子1剩余关键词总数	0		1		2		≥3	
	语义一致性	句子对总数	语义一致性	句子对总数	语义一致性	句子对总数	语义一致性	句子对总数
0	1	58	1	8	1	4	1	1
	0	3	0	8	0	5	0	7
1	1	22	1	8	1	2	1	6
	0	30	0	32	0	23	0	15
2	1	19	1	5	1	7	1	3
	0	26	0	26	0	16	0	24
3	1	8	1	4	1	2	1	0
	0	31	0	25	0	20	0	18
≥4	1	19	1	20	1	3	1	4
	0	81	0	93	0	53	0	61

表 1 所列为剩余关键词总词数与句子对语义一致之间的关系分析。如表 1 所列，将句子 1 剩余关键词总数分为 4 类：等于 0、等于 1、等于 2 和大于等于 3，句子 2 剩余关键词总数分为 5 类：等于 0、等于 1、等于 2、等于 3 和大于等于 4。当已知句子 1 剩余关键词总数与句子 2 剩余关键词总数时，统计满足已知条件的句子对的语义一致性分布，从而得到表 1。例如，从表 1 中可以得出，当句子 1 剩余关键词总数等于 0，同时句子 2 剩余关键词总数也等于 0 时，在所采用的实验数据中，共有 61 对句子满足条件，其中有 58 对句子是语义一致的，3 对句子是语义不一致的。通过分析表 1 中数据，本文提出如下规则来有效提高 F 值，其中，句子 $u\in U$ 为用户提出的问句，句子 $q\in Q$ 为数据库中的问题句。

设 $Size(f(s_1,s_2))$ 表示经过关键词消解后存在于句子 s_1 中且不存在于句子 s_2 中的剩余关键词总词数。

规则 1　当 $Size(f(u,q))+Size(f(q,u))\leqslant 1, u\in U, q\in Q$，且剩余关键词不含有否定词时，则两个句子是语义一致的。

规则 2　当 $Size(f(u,q))=0, Size(f(q,u))=2, u\in U, q\in Q$，且剩余关键词不含有否定词时，则两个句子是语义一致的。

规则 3　当 $Size(f(u,q))=2, Size(f(q,u))=0, u\in U, q\in Q$，且剩余关键词不含有否定词时，则两个句子是语义一致的。

规则 4　当句子 u 和句子 q 的剩余关键词中含有的否定词为奇数时，则两个句子是语义不一致的。

采用基于关键词特征和语义特征的句子相似度算法判别句子语义一致性的具体做法为：首先利用算法 1 计算剩余关键词列表，再根据上述规则即可以判定哪些句子对是语义一致的。

2.3　面向用户意图的句子相似度算法

上述基于关键词特征和语义特征的句子相似度算法中，由于经关键词对比消减后剩余关键词总词数相差较大的句子往往很多是语义不一致的，因此判断句子语义一致性的规则主要考虑剩余关键词总词数相差较小的句子。

然而，当两个句子中句子 1 是长句，且语义上包含句子 2 时，两个句子也有可能是语义一致的。例如，在用户使用搜索引擎寻找答案时，句子 1 是用自然语言描述的用户提问，而自然语言并不能被计算机理解，需要将用户提问转换成计算机可以理解并能快速搜索到答案的问句，转换后的句子称为句子 2，句子 2 代表用户的真实需求。若句子 2 是句子 1 的子句，那么搜索到的答案将会是用户提问的一部分或者全部答案。此时，可以认为句子 1 和句子 2 是语义一致的。也即，搜索引擎发展到一定程度后，需要解决用户提问句和用户真实需求之间对应的问题。

自然标注是用户无意中对自然语言处理资源所做的义务“标注”[10]，标点符号也是“显式自然标注”的一种。虽然标点符号在自然语言处理中往往作为停用词被去掉，但是本文认为标点符号在一定程度上可以反映出一个长句子包含不同子含义。若将一个长句子看成若干个语义句的集合，那么标点符号就是语义句的分割符。

基于以上因素的考虑，提出面向用户意图的句子相似度算法。利用用户提问中的标点符号，将用户提问转换成用户提问子句集，计算子句集中每一个元素与计算机存储的问题集中问题的语义一致性。若存在一个子句集中元素与问题集中某个问题是语义一致的，则认为用户提问与这个问题是语义一致的。在判断用户提问的一个子句与问题集中某个问题是否是语义一致时，采用算法 1 计算剩余关键词词数，再根据规则判定其语义一致性。由于子句所包含的关键词个数相对完整问题句的关键词个数较少，因此提出规则 5 来判断子句和问题集中问题的语义一致性。

规则 5　当 $Size(f(q,u_i))=0, Size(f(u_i,q))\leqslant 1, u_i\in U, q\in Q, i=1,2,\cdots,n$，且剩余关键词不含有否定词时，则用户提问 U 与问题集中问题 q 是语义一致的。

采用面向用户意图的句子相似度算法判别句子语义一致性的详细描述如算法 2 所示。

算法 2　面向用户意图的句子相似度算法

```
输入:用户问题 U,问题集中某个问题 q
输出:语义一致性
1. string[] clauses=U. Split(punc);
2. for (int x=0;x < clauses. Length;x++)
3. {
4. for (int y=x;y < clauses. Length;y++)
5. {
6.     string clause="";
7.     for (int i=x;i <= y;i++)
```

```
8.    {
9.        clause += clasuse[i];
10.   }
11.   算法 1(clause,q);
12.   if (f(clause,q)和 f(q,clause)符合规则 5)
13.   {
14.       用户问题 U 与问题 q 是语义一致的;
15.   }
16. }
17. }
```

3 实验结果及分析

3.1 实验数据

本文实验所采用的数据来自百度开放研究社区举办的 Q-T 语义一致识别大赛[11]中所提供的标注语料。此语料中共有 800 对句子,即 800 句 query 和 800 句 title,其中 query 指用户的搜索语句,代表用户真实的需求,即上文提到的句子 Q;title 指用户提问语句,是用户提交的自然语言,即上文提到的句子 U。800 对句子已经标注过,语义一致表示为 1,也称为正例;语义不一致表示为 0,也称为反例。表 2 是数据样例。

表 2 数据样例

语义一致性	Query	Url	Title
1	玉兰花	http://zhidao.baidu.com/question/5406833.html	关于玉兰花?
0	同学之间作文	http://zhidao.baidu.com/question/193945642.html	同学之间的友谊作文 300 字

3.2 实验环境、工具、语义资源

实验计算机软件配置:Windows 8 操作系统,Microsoft Visual Studio 2010 C#。

实验使用工具:NLPIR 汉语分词系统[12] 2013 版。

语义资源:知网错别字表[13]、知网同义词集[13]、哈尔滨工业大学信息检索实验室同义词词林[14]扩展版、中文停用词表、否定词表。

3.3 评价指标

本实验主要评估语义一致句子(也就是标识为正例的句子)的准确率、召回率和 F 值,并以最终的 F 值作为衡量各个方法的指标。

$$\text{准确率 } P=\frac{\text{正确标识为正例的句子数量}}{\text{标识为正例的句子数量}}$$

$$\text{召回率 } R=\frac{\text{正确标识为正例的句子数量}}{\text{应该被标识为正例的句子数量}}$$

$$F\text{ 值}=\frac{P*R*2}{P+R}$$

3.4 实验结果及分析

将 800 句 query 和 800 句 title 采用本文提出的方案计算其相似度以判断语义一致性,并将结果与数据集中已给出的标注结果比较,得出本方案的准确率、召回率和 F 值。表 3 是本方案各步骤采用不同算法计算得出的准确率、召回率和 F 值。

表 3 本方案各步骤采用不同算法计算结果

步骤	算法	准确率 P	召回率 R	F 值
1	去除句末标点的编辑距离句子相似度算法	0.803	0.2611	0.3941
2	基于关键词特征和语义特征的句子相似度算法	0.6073	0.5714	0.5888
3	面向用户意图的句子相似度算法	0.5933	0.6108	0.6019

通过表 3 可以得出,去除句末标点的编辑距离算法的召回率仅为 0.2611,说明该算法只识别出相对较少的一部分语义一致句子,但是其准确率较高,作为本方案第一步的算法拥有较高的正确率,可以保证后续算法的可行性。同时该算法的 F 值为 0.3941,说明该算法具有扩展性。在第二步中提取关键词特征和语义特征,采用基于关键词特征和语义特征的句子相似度算法使得 F 值提升到 0.5888,算法综合性能提升明显。虽然牺牲了一部分的准确率,但是召回率提升了一倍多,这说明该算法采用关键词特征和语义特征时可以识别出更多的本应为语义一致的句子。在第三步中通过分析用户意图,总结用户提问和问题集中问题的特性,采用面向用户意图的句子相似度算法提高了 F 值,最终达到 0.6019。作为综合评估算法效果的标准,F 值越高说明算法越有效。

表 4 对比了本文提出的句子相似度分层计算方法与文献[1]提出的基于改进编辑距离的句子相似度计算方法。其中,基于改进编辑距离的句子相似度计算方法采用的阈值为3.9。由表 4 可见,本文提出的分层计算方法明显优于改进编辑距离计算方法,准确率提高了 9%,F 值提高了 15%。这主要得益于本文利用编辑距离算法克服了中文分词尚存在错误的问题,再利用关键词特征和语义特征计算句子相似度克服了编辑距离算法具有的词序限制,最后结合用户查询意图的特点,利用自然标注计算包含长句的句子对的句子相似度。综上所述,本文提出的句子相似度分层计算方法符合实际情况,具备一定的实用性。

表 4 分层计算句子相似度方法与改进编辑距离计算句子相似度方法的结果比较

方法	准确率 P	召回率 R	F 值
基于改进编辑距离的句子相似度计算方法	0.5092	0.4089	0.4536
句子相似度分层计算方法	0.5933	0.6108	0.6019

结束语 句子相似度计算作为自然语言处理领域的基础工作,具有重要的意义。本文通过将数据分为 3 种类别:基于编辑距离、关键词和同义词语义方法,利用信息检索领域中用户查询意图匹配问题的特点,提出 3 种算法来分别处理不同类型的数据。实验表明,本方案可有效提高 F 值,F 值达到 0.6019。

参 考 文 献

[1] 车万翔,刘挺,秦兵,等. 基于改进编辑距离的中文相似句子检索[J]. 高级技术通讯,2004,14(7):15-19

[2] 吕学强,任飞亮,黄志丹,等. 句子相似模型和最相似句子查找算法[J]. 东北大学学报:自然科学版,2003,24(6):531-534

[3] 杨思春,等. 一种改进的句子相似度计算模型[J]. 电子科技大学学报,2006,35(6):956-959

[4] 赵臻,吴宁,等. 基于多特征融合的句子语义相似度计算[J]. 计算机工程,2012,38(1):171-173

[5] 程传鹏,吴志刚. 一种基于知网的句子相似度计算方法[J]. 计算机工程与科学,2012,34(2):172-175

[6] 刘宏哲. 一种基于本体的句子相似度计算方法[J]. 计算机科学,2013,40(1):251-256

[7] 李彬,刘挺,秦兵,等. 基于语义依存的汉语句子相似度计算[J]. 计算机应用研究,2003(12):15-17

[8] 王品,黄广君. 信息检索中句子相似度计算[J]. 计算机工程,2011,37(12):38-40
[9] 李茹,王智强,等. 基于框架语义分析的汉语句子相似度计算[J]. 计算机研究与发展,2013,50(8):1728-1736
[10] 孙茂松. 基于互联网自然标注资源的自然语言处理[J]. 中文信息学报,2011,25(6):26-32
[11] 百度 Q-T 语义一致大赛 [DB/OL]. http://openresearch.baidu.com/activityindex.jhtml? channelId=452
[12] NLPIR 汉语分词系统[DB/OL]. http://ictclas.nlpir.org/
[13] 董振东,董强. 知网[DB/OL]. http://www.keenage.com
[14] 梅家驹,竺一鸣,高蕴琦,等. 同义词词林[M]. 上海:上海辞书出版社,1993:106-108

(上接第 200 页)

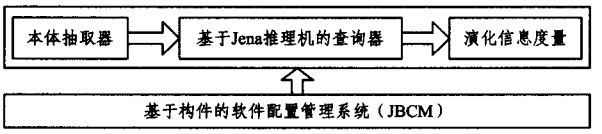

图 3 构件化软件演化信息获取及度量系统原型

结束语 在构件化软件开发中,利用演化信息能够更好地辅助软件的开发。传统的软件演化信息通常以文件或者项目作为软件变化的基本单元,不能有效地支持构件化软件演化信息的存储和检索。为此,本文在基于构件的软件配置管理模型的基础上,对构件化软件演化信息进行本体建模,借助描述逻辑 DL Query 以及基于属性相关性的 SPARQL 查询等机制实现直接演化信息查询;同时,通过定义规则和在 Jena 推理机中实现构件化软件蕴含演化信息的获取;最后提出了一种预测演化趋势的构件化软件演化度量的方法,为后期的构件化软件维护工作提供了基础。

需要指出的是本文的工作与 Christoph Kiefer 的基于本体的软件资产库挖掘[13]相似,不过他们的研究面向对象软件的演化,侧重点是扩展 SPARQL 语言以支持基于相似度的查询,以及通过简单的示例说明在软件演化可视化、度量、本体推理上的应用;而本文的工作针对的是构件化软件演化,着重于基于本体和 Jena 推理机的构件化软件演化信息获取及演化度量。在未来的工作中,将完善目前的系统原型,并将其封装成 Web-Service 的形式。

参 考 文 献

[1] Lehman M, Belady L. Program Evolution: Processes of Software Change [M]. London Academic Press: London, 1985: 538-540
[2] Gîrba T, Ducasse S. Modeling History to Analyze Software [J]. Journal of software maintenance and evolution: research and practice, 2006, 18: 207-236
[3] Gîrba T. Modeling History to Understand Software Evolution [D]. Fakultät der Universität: Berne, 2005: 13-19
[4] Morse T. CVS[J]. Linux Journal, 1996(21es): 3, 1996
[5] Subversion[OL]. [2013-12-15]. http://subversion.tigris.org/
[6] Robbes R, Lanza M. Versioning systems for evolution research [C]// 8th International Workshop on Principles of Software Evolution, 2005 (IWPSE 2005). IEEE Computer Society, 2005: 155-164
[7] Chu-Carroll M C, Wright J, Shields D. Supporting aggregation in fine grained software configuration management [C] // Formal Software Engineering FSE'02. ACM Press, 2002: 99-108
[8] Dig D, Manzoor K, Johnson R, et al. Refactoring-Aware Configuration Management for Object-Oriented Program [C] // the 29th International Conference on Software Engineering (ICSE'07). 2007: 427-436
[9] Robbes R. Mining a Change-Based Software Repository [C] // Proceedings of the Fourth International Workshop on Mining Software Repositories (MSR '07). 2007: 15-22
[10] Xing Z, Stroulia E. Refactoring detection based on umldiff change-facts queries [C] // Proc. WCRE'06. 2006: 263-274
[11] Dig D, Comertoglu C, Marinov D, et al. Automatic detection of refactorings in evolving components [C] // Proc. ECOOP'06. 2006: 404-428
[12] Taneja K, Dig D, Xie Tao. Automated detection of api refactorings in libraries [C] // ASE'07. ACM, 2007: 377-380
[13] Kiefer C, Bernstein A. Mining Software Repositories with iSPARQL and a Software Evolution Ontology [C] // Fourth International Workshop on Mining Software Repositories (MSR'07). 2007
[14] Matthew H. A practical Guide to Building OWL Ontology Using the Protégé-OWL Plugin and Code Tools [Z]. [2013-05-10]
[15] Restol, Jena2. A semantic Web Framework [OL]. [2013-12-20]. http://Jena.Sourceforge.net
[16] OWL Web Ontology Language Overview [OL]. [2013-03-19]. http://www.w3.org/TRlowl-features/
[17] Sager T, Bernstein A, Pinzger M, et al. Detecting Similar Java Classes Using Tree Algorithms [C] // Proc. of the 2006 Int. Ws. on Mining Software Repositories (MRS '06). New York, NY, 2006
[18] Stevens R, De Roover C, Noguera C, et al. A History Querying Tool and its Application to Detect Multi-version Refactorings [C] // 17th European Conference on Software Maintenance and Reengineering. 2013: 335-338
[19] Prete K, Rachatasumrit N, Sudan N, et al. Template-based reconstruction of complex refactorings [C] // Proc. of the 2010 IEEE Int. Conf. on Software Maintenance. 2010: 1-10
[20] 曹居易,石玲. 基于 OWL 的软件工程数据建模[J]. 计算机研究与发展,2009,46(增刊):214-221
[21] 李季,刘春梅. 基于本体的可信软件演化框架模型[J]. 计算机应用研究,2010,27(12):4551-4554
[22] 何文民,沈国华,黄志球. 基于本体的特征模型演化的一致性验证[J]. 计算机应用研究,2013,30(7):2072-2076
[23] 张路,谢冰,梅宏,等. 基于构件的软件配置管理技术研究[J]. 电子学报,2001,29(2):266-268
[24] 钟林辉,谢冰,邵维忠. 扩充 CDL 支持基于构件的系统组装与演化[J]. 计算机研究与发展,2002,39(10):1361-1365