# 第四章 对需求引起的变更进行审查利用

# 文本相似度算法的模型验证

## 4.1 数据准备及预处理

软件生命周期数据的获取的过程如图17所示，其主要流程可描述为：



图17 软件生命周期数据获取

(1)数据收集和标注，由相关专业人员对软件生命周期数据的收集和标注，其中用到的软件生命周期数据包括软件需求文档，设计文档，代码文件等。

(2)脱密处理，对于航空发动机软件相关文档，有必要的话需进行脱密处理，表5列举了一些常用的脱密技术。其中为了更好的配合本文中提出的相似度计算方法，关键词可以表示字母加数字。

(3)文本处理，对于软件数据，可以使用一些需求抽取工具如信息抽取工具如SuperTracePlus[14]等来辅助相关人员行抽取，也可以编写脚本根据word文档的目录结构和标题对文本进行抽取，自动解析为结构化数据。

(4)最后需要专业人员对数据分类情况进行检查和修改，确认无误后，存入数据库。

表5 常用脱密方法

|  |  |
| --- | --- |
| **脱密方法** | **详情** |
| 彻底删除法 | 对一些工艺技术和关键数据，采用彻底删除的方法 |
| 模糊处理法 | 对于特定的装备，省去装备具体的名称，使用“xx”等或其他模糊名称；可使用虚数表示具体数字 |
| 替换取代法 | 将保密的某个装备表示为特定名词或符号 |
| 数据示意法 | 使用合理取值范围内其他数字表示装备性能参数或分数 |

这种数据获取方式可以应用于领域文本数据的获取，如航空发动机软件开发领域文档。

但由于发动机控制系统软件的各种软件数据有很强的保密性，并不容易获得，同时软件需求间的可追溯性关系需要专业人员根据经验确定，在操作时有很多不便。因此，本文选取在需求追踪任务中常用的五个公开数据集来进行实验。选取这些数据集的原因除了克服上述适航领域数据的缺点之外，这些数据集中，每个单独的文件都是一个不能再进一步分解的数据。这些数据的类型与DO-178B/C所对应的高层需求、低层需求和源代码等所要求的类型相近，它满足实验模型的输入要求。其次这些数据集广泛在需求追踪任务中使用，有利于对比实验的设置，对比结果更加具有说服力，最后选择多个不同数据集的目的是为了说明本文所提出的算法可应用于不同的数据集。

本文选取的五个数据集分别是MODIS，CM1，eTOUR，EasyClinic和iTrust。这五个数据集有两个来源：MODIS是NASA公开项目中需求数据的一个子集，它由Hayes[14]等人整理，包含高层需求和低层需求两种文档，多次被用于需求间的可追溯性的研究当中。CM1同样来自于NASA的公开项目CM1，它也包含软件的高层需求和低层需求两类文档。eTOUR来自于一个专用于旅游的电子导航软件； EasyClinic来自于医疗管理的软件项目，两个项目均由Salerno大学开发，开发语言为Java；iTrust来自于一个用于记录医药信息的开源项目，开发语言同样是Java。这四个数据集均可从CoEST网站上得到。CoEST是一些研究学者为了鼓励和促进软件的可追溯性研究而建立的网站，在该网站上提供了多种已经整理好的软件数据集，这些数据集从实际的项目中整理而来，并经过多位专家学者验证，被广泛使用于软件可追溯性的研究当中。其中，DO-178B/C要求的软件需求的可追溯性所需数据，均可在这些数据集中找到。有一点需要指出的是，这些数据集中不仅包含软件的各种生命周期数据，还包括已经验证的可追踪链，这给本文的实验带来了极大的便利。实验数据集的描述见表6。

可以看到，在五个数据集中除了包括高层需求、低层需求、测试用例等不同类型文本文件，还有源代码等代码文件，除此之外，每个数据集中都提供了经由验证的可追踪关系结果集，并在单独文件中给出了软件制品间正确的追踪关系。

表6 实验数据集描述

|  |  |  |  |
| --- | --- | --- | --- |
| **数据集** | **描述** | **追踪链数量** | **文件总数** |
| MODIS | 19个高层需求，49个低层需求 | 41 | 68 |
| CM-1 | 2个高层需求，53个低层需求 | 45 | 75 |
| eTOUR | 58个用例，116个代码文件 | 308 | 174 |
| EasyClinic | 30个用例，20个UML交互图文件，  63个测试用例，47个代码文件 | 47 | 160 |
| iTrust | 131个用例，367个代码文件 | 534 | 498 |

选好实验数据集后，需要对数据集进行预处理以方面本文后面的实验使用。对于数据集的预处理使用传统的数据处理方法，对于每一个数据集：把数据集中所有文件作为输入，通过空格文本分割为单个单词，去除其中除了下划线和连字符之外所有的非数字、非字母符号，将所有单词转化为小写，并去除停用词，对于文本文件停用词是NLTK[48]所提供的停用词加上数据集中的冗余词，而代码文件中停用词是相应代码中的关键字(keywords)。

除了实验所需的软件相关数据集，由于计算文本相似度需要使用事先训练好的词向量，因此，词向量的训练需要一个大型并且权威的语料库，并且包含大量无标记的语料。对于语料库的选择，Siwei L.等人[49]研究并指出，训练词向量的最终质量与语料库的领域和大小相关，用领域相关语料训练效果比使用领域无关的训练效果要好；在同领域的情况下，训练所使用的语料库越大越好。因此，本文在选取语料时，选择了文本质量较高的维基百科词条作为训练语料。通过编写爬虫获得维基百科中技术分类下相关词条，解析维基百科网页，对网页信息进行清洗，从而获得的所有词条及其描述来获得领域相关的语料。其流程如图18所示。最终得到的语料库大小为326M。需要注意的一点是在处理维基百科词条时要尽量保证每个词条的完整性。Gensim是一个开源工具，它由python实现，集成了词向量模型，是一款很强大的工具。本实验中使用它来训练词向量，在训练过程中词向量的训练模型选择的Skip-gram，预测窗口的值设置为5，并将最终的向量的维度设置为200；在训练过程中选择开启softmax来加速训练速度。



图18 维基百科数据获取流程

## 4.2 实验设置

在第三章中，本文提出的面向航空发动机适航领域需求追踪算法可以分为两个阶段：文本相似度计算和学习排序模型训练。因此，本节针对每个阶段设置了对比实验，对于CWI算法，除了与计算文本相似度过程中所用到基于统计的VSM(向量空间模型)算法和只考虑词向量的WE(隐语义模型)算法，还和目前需求追踪领域较为成熟的LSI算法进行了对比实验，设置这组对比实验的目的是验证CWI算法在计算领域文本相似时的有效性。

下面详细介绍每组实验设置，实验对比方法如表7所示。

表7 设计的对比实验

|  |  |
| --- | --- |
| **实验** | **对比实验方法** |
| 实验一 | VSM、WE、LSI |
|  |  |
|  |  |

由于在训练好的词向量中，已经包含了单词的各种词形，因此不需词干化，但在使用VSM、LSI算法时对软件文本的处理略有不同，还需要对单词进行词干化。VSM和LSI具体算法已经在第二章介绍。这两个算法可以用来验证在的算法在考虑语义之后对最终结果的影响。

WE方法即利用公式(3.4)所计算出的语义相似度，这种方法直接忽略计算过程中的缺失词，使用这种方法是为了验证软件制品中缺失词对相似度计算的影响。

### 4.4.1 文本相似度对比实验

本组实验选用了第二小节介绍的VSM、WE、LSI和本文所提出的文本相似度算法进行了对比实验，此次实验所使用的数据集为第一节所介绍的全部五组数据集，详细描述如表6所示。为了简便表示，可以将所使用数据集中数据名称的首字母大写代替，即HL表示高层需求文档(high-level requirements)，LL表示低层需求文档(low-level requirements)，UC表示用例文档(user cases)，CC表示代码文档(code classes)，ID表示UML交互图文档(interaction diagrams)，TC表示测试用例文档(test cases)。实验结果见表8， “->”表示可追溯性链，PRE代表precision，REC代表recall。为了更好的展示最终结果，本文计算了不同算法下的F1值，其直方图如图19所示。

表8 四种算法的precison和recall值

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DataSets\Metrics | |  | **VSM** |  | **WE** |  | **CWI** |  | **LSI** |  |
|  | |  | **PRE** | **REC** | **PRE** | **REC** | **PRE** | **REC** | **PRE** | **REC** |
| MODIS | HL->LL | | 0.215 | 0.288 | 0.319 | 0.428 | 0.315 | 0.563 | 0.279 | 0.357 |
| CM1 | HL->LL | | 0.158 | 0.187 | 0.249 | 0.226 | 0.413 | 0.396 | 0.127 | 0.41 |
| eTOUR | UC->CC | | 0.07 | 0.301 | 0.104 | 0.302 | 0.135 | 0.563 | 0.077 | 0.221 |
| iTrust | UC->CC | | 0.006 | 0.245 | 0.227 | 0.404 | 0.243 | 0.422 | 0.009 | 0.45 |
| EasyClinic | UC->ID | | 0.185 | 0.764 | 0.324 | 0.732 | 0.342 | 0.806 | 0.259 | 0.833 |
| UC->TC | | 0.36 | 0.685 | 0.542 | 0.832 | 0.553 | 0.865 | 0.45 | 0.755 |
| UC->CC | | 0.208 | 0.487 | 0.215 | 0.677 | 0.247 | 0.764 | 0.317 | 0.503 |

这里将实验结果分为两组进行对比，第一组是VSM、WE和CWI方法间的对比，可以从表8及图19中看到，CWI对比其他两个算法精确率，召回率及F1值均有一定提升；由第三章可以知道CWI是在计算文本相似度时，既考虑了语义相似度，又考虑了在计算 语义相似度过程中的缺失词，通过CWI与这两个算法单独作用的对比。可以充分说明，本文提出的CWI算法由于考虑了文本词语的特征。

第二组的对比是WE、CWI算法和目前需求追踪领域成熟的LSI算法进行比较发现CWI算法总是比直接使用词向量的WE效果要好，这是因为当为1时，两者等价。即WE算法是CWI的下限。可以看WE、CWI算法在各个指标上对比LSI均有所提高。产生这种结果的一个主要原因是LSI算法在计算文本相似度时，将文本映射到一个较低维的空间，具有相同主题的词语被映射到同一维度，而这种变幻是通过数学方法得到，并没有考虑词语的语义特性，忽略了词语之间的关系，CWI方法恰好弥补了这一缺点。

图19 不同算法下的F1值

综合以上两组对比，可以看到CWI算法在计算文本相似度时，既考虑了语义的相似度，同时结合领域文本的特性，在运行过程有效的处理了文本数据中缺失词，在计算了文本相似度时具有一定优势。