2412 绪论前自己写的文字 主要是需要写摘要和翻译英文

7283 绪论 2天 全天 10个西红柿

1. 6000 相关理论和技术研究 2天 全天 8个西红柿
2. 6600 模型的建立 3天 工作日 20个西红柿
3. 4.1 文本相似度算法的设计 1972 1+1天 工作日 5个西红柿

4.2 算法的验证 2238 5296 1+1天 工作日 20个西红柿

4.3 怎么实现功能 1天 工作日

1. 设计与实现 4288 带设计 20 天
2. 对目标进行分解，呈列出要实现的“小目标”，再根据与生命周期数据间的关系
3. 根据Do178c规定的生命周期数据类型，参考Do178c实验给出的数据元，
4. 确认进行建模使用的方法：安全UML的精简版本
5. 选择安全UML作为**建模的方法，**（使用到了哪些相互间关系）

**二、配置管理过程相关的元数据，**

三、分析配置管理目标、过程、证据间的关系，（利用GSN）,

介绍：

对于航空机载软件的研制和适航审定，FAA确定了Do-178c、Order 8110.49chg1标准，其中，从适航符合性角度，Do-178c是基础标准，它定义了开发过程需要满足的与相应安全等级相适应的目标；Order8110.49 chg1则属于操作标准，定义了如何实际操作来审查一个项目的研制过程是否满足Do-178c的相应目标要求。Do-178c标准从软件安全性角度，按照机载软件的安全级别对开发过程和活动提出了一系列目标要求；后者则从符合性审查角度规定了具体的审查阶段、活动、涉及的数据和相应的审查规则等。

## 配置管理过程建模方法的选取

由于所建立模型的过程、数据与目标间的复杂关系，选择一种适用于此次建模的方法是一个难点。

由于过程建模是对软件过程活动的再工程,涉及到软件产品的开发与维护、软件项目管理、过程管理与过程改善等各个方面,涉及到过程活动、角色、产品、资源和约束等各种过程成分,还涉及到建模所用的形式化方法,加之软件过程本身具有的复杂性,使得构造过程模型的方法也是多种多样的。

* 以**过程所涉及的各种成分**(如活动、角色、产品、资源和约束等)为出发点来考虑过程建模。主要有四种不同方法：

1. 以活动为中心的建模方法
2. 以角色为中心的建模方法
3. 以产品为中心的建模方法
4. 基于过程模板的建模方法

* 以软件过程模型侧重于描述**功能目标**还是侧重于描述**活动行为**。主要考虑过程模型是否支持软件过程的动态性进行分类

1. 面向活动的过程建模方法
2. 面向目标的建模方法
3. 面向活动和面向目标的建模方法的结合。

* 按照过程建模所采用的不同形式化方法的分类。

1. 过程程序设计方法
2. 多功能分解方法
3. 基于计划的建模方法
4. 基于Petri网的建模方法
5. 基于规则的建模方法
6. 基于逻辑的建模方法
7. 基于知识的建模方法
8. 混合风格的建模方法

基于建模目的以及配置管理过程特点的考虑，本研究所准备采用基于过程模板、面向活动和面向目标结合、基于Petri网与UML集合的建模方法，并在研究过程中跟进实际情况加以考虑变更或改动。

根据最佳实践提出的数据项管理的要求，将管理的配置项分为了数据元与数据项，并且识别了以下数据项（表 3）：

|  |  |
| --- | --- |
| 数据项编号、名称 | 数据项编号名称 |
| D-01 项目进度计划 | D-29软件需求数据（条目化） |
| D-02 分配到软件的系统需求 | D-30高层需求与系统需求的追踪数据 |
| D-03软件等级 | D-31软件架构（非条目化） |
| D-04项目估算数据 | D-32底层需求层次结构 |
| D-05项目软件生命周期 | D-33底层需求（条目化） |
| D-06项目管理计划 | D-34软件设计说明（非条目化） |
| D-07软件合格审定计划 | D-35底层需求与高层需求的追踪数据 |
| D-08软件开发计划 | D-36源代码 |
| D-09软件验证计划 | D-37源代码与底层需求的追踪数据 |
| D-10软件配置管理计划 | D-38测试用例 |
| D-11软件质量保证计划 | D-39测试用例与高层需求的追踪数据 |
| D-12软件需求标准 | D-40测试用例与底层需求的追踪数据 |
| D-13软件设计标准 | D-41高层需求测试覆盖数据 |
| D-14软件编码标准 | D-42底层需求测试覆盖数据 |
| D-15核查检查单 | D-43测试规程 |
| D-16核查记录 | D-44测试规程与测试用例的追踪数据 |
| D-17问题报告 | D-45软件生命周期环境 |
| D-18评审检查单 | D-46目标代码 |
| D-19评审报告 | D-47可执行目标代码 |
| D-20软件配置索引 | D-48编译、链接、加载日志 |
| D-21基线 | D-49测试结果 |
| D-22变更请求 | D-50测试结果与测试规程的追踪数据 |
| D-23软件配置状态报告 | D-51测试结构覆盖数据 |
| D-24软件配置管理记录 | D-52目标代码与源代码的追踪分析数据 |
| D-25软件生命周期环境配置索引 | D-53加载控制记录 |
| D-26软件质量保证记录 | D-54软件完成综述 |
| D-27高层需求层次结构 | D-55软件符合性评审报告 |
| D-28高层需求（条目化） |  |

这与在Do178c标准中，目标、过程、与数据间的关系是相似的，通过适当的变换，所以可以利用GSN，对do178c标准进行具体分析，找出相关目标的过程要求、子过程及存在的证据间的对应关系。

在Do178c标准的71个目标中，以下6个是与配置管理相对应的目标。

1. 标识构型项（A-8.1）
2. 建立基线和可追溯性（A-8.2）
3. 建立问题报告，变更控制，变更评审和构型状态纪实机制（A-8.3）
4. 建立归档、检索和发布机制（A-8.4）
5. 建立软件加载控制机制（A-8.5）
6. 建立软件生命周期环境控制（A-8.6）

将Do178c标准所要求的目标对应于GSN中的安全目标，生命周期数据或数据元对应GSN中的安全证据，而标准中对过程要求目标可以作为论证策略的一部分。其中将Do178c提供的过程指导作为策略或是、论点的一部分在分析过程中得到证明，能够减轻对整个论证过程进行分析的复杂性。

|  |  |
| --- | --- |
| 数据项内容 | 原子性数据项 |
| a.1软件配置索引 |  |
| a.2需要变更的基线 | a.2.1基线的受控软件库  a.2.2基线.pre  a.2.3基线.next |
| a.3变更完成后的基线 | a.3.1基线的受控软件库  a.3.2基线.pre |
| a.4配置项库 |  |
| a.5变更申请单 | a.5.1变更影响分析（包含受变化影响的生命周期数据）  a.5.2受影响的配置项或数据元列表（变更对象）  a.5.3软件发生错误的起始处（如果是需求变更则需求处）  a.5.4变更和行动 ） |
| a.6变更审批单 | a.6.1变更影响分析 （包含受变化影响的生命周期数据）  a.6.2受影响的配置项或数据元列表（变更对象）  a.6.3软件发生错误的起始处（如果是需求变更则需求处）  a.6.4变更和行动 ） |
| a.7问题报告 | a.7.1变更影响分析 （包含受变化影响的生命周期数据）  a.7.2受影响的配置项或数据元列表（变更对象）  a.7.3软件发生错误的起始处（如果是需求变更则需求处）  a.7.4变更和行动 ）  a.7.5问题描述  a.7.6问题报告的批准、关闭情况。 |

软件质量评估的方法，主要是针对研发方的，对于审定方的，间“证据”论文，（大多都没有太强的实用性）

选择建模的方法，实际情况，国外的研发审定工作1.目前证据管理的情况 2.国外成立的专门进行研发审定的研究、统一工作的传统，其中，1论文提到了一种安全UML，该UML的目的是， 某种程度上以后的相关工作人员都应参照此项的情况，

但此UML专注于研发，同时考虑到了研发过程中为审定过程提供需要的生命周期数据，而我国在机载软件研发审定方面的实际情况是，审定基本靠专业人员经验，研发为了进度，人员对标准理解不多，研发主要还是按照传统软件的研制流程，最后生成的生命周期数据的形式与可审定的程度参差不齐，未来必然越来越规范化，但是一个逐步的过程，另外，由于研发方要考虑的到具体的设计等必然要精细的多，而审定工作不可能将每次审定一个软件都编写一次，故必然应该只专注于标准提到的统一性的要求，必须抽取出标准中共同的目标。

考虑这两点因素，本人此次建模参考安全UML标准，特别是n.n章对各个对象间关系的分类、精简与总结；同时在数据项的选择上，设定为可粗可细，（不归这里管），选取了实际工作中最关键的相关的要素，结合标准规定的数据项要素， 在软件的实际设计开发过程中，为能够适应不同格式标准情况的生命周期数据，可以选择审定项的细致程度，当然可由该工具审定的结果也根据生命周期数据情况的不同而不同。对于无法标准化或由于复杂程度只能有专家进行分析的部分，展现为易于得到的文本以利于审查人员管理证据

相关的要素包括\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_，来源于\_\_\_\_\_\_\_\_\_\_\_\_

分析各个要素之间的关系,通过Do178c标准的原文、实践、咨询相关专家人员。

将需要达到的目标分为Ji类。1.追溯性。2。需满足的时间、人员、归属、已被验证、顺序关系、并发、 一致性等关系。先后关系、协作关系、数据的传递关系乃至资源的共享关系等。还有多次变更间的关系，3.对于充分性的要求，无法进行判断，只能在初审（根据审定指南相关部分，由审查人员人工提前审查）。

建立 对 配置管理进行审定的 软件工作的UML图

对于标准中对可追溯性的要求，提出了一种。。。的方法，

对追溯性进行判定的方法研究，方法是利用 改进软件工程领域 文本搜索的 算法。

分析 存在哪些特点：在变更管理或是基线改变中，关键字加权？，缺失词处理？

变更：主要是需求变更，也可能是错误引起的变更 所以，在变更阶段主要是需求追溯性的改变。

对于一些难以判定的，留给人工判定，

而系统则是可以提供（可选择的数据项）的追溯性的计算，审查人员根据追溯性计算 的结果，进行进一步的检查。

涉及到配置管理的数据项和数据元

1.

涉及到追溯性的

每次变更（两条基线间中的关键要素：变更申请时间，数据项，变更申请人，变更审核人员，基线，受控软件库，基线间的衍生关系，配置项的追溯关系？ 变更应记录、批准、并进行跟踪，变更的源头。

建模和设计软件的关系：

1. 若是建变更管理的模型。通过最后的数据进行验证。变更管理中的要求怎么弄，变更管理是有很多步骤的，（验证） （执行）

软件需求变更管理的研究与实践 赵海英 第三章 优化的统一变更管理

1. 统一的变更管理流程
2. 分析do178c目标，
3. 根据1、2挑选出 需要的配置项。 2、3考虑顺序。 3应是在1、2的基础上，2，可能需要根据3进行微调
4. 方法：用安全UML对统一的变更流程进行建模，并分析do178c要求的证据应在哪一步进行验证，并用安全UML表示出来。

优化的统一变更管理 Rational的统一变更管理为基础（UCM）,同时考虑Do178C标准的相关目标和要求，加上其他促进变更管理的措施，建立的“优化的统一变更管理”

变更处理流程：1.变更评估处理流程 2.软件配置管理流程（变更实施流程）（分为两个部分，1.变更评估 问题报告、变更申请、变更审核 2. 新的基线的建立、配置项改变，、软件受控库的改变，更改的配置项的识别等，还涉及到可追溯性的判断、）

强调的是过程而非方法，不需要借助复杂的变更管理工具。

变更申请阶段：变更申请编号，变更描述，变更原因，不实施的后果 变更申请单

变更评估阶段：影响和风险评估 变更评估单

变更审批阶段：批准签名，批准日期。 变更审批结果（单？）

（三个共同要有的变更对配置项的影响。）哪些配置项

变更行动阶段的属性: 变更负责人。变更 开始执行和完成时间期限。

配置管理阶段：

变更执行完成后：变更核查日期 核查结果

变更管理流程图 放在第二章

时序图 有了，基本可以用

变更流程 基本确定

涉及的配置项 一会儿进一步明确

状态迁移图， 感觉好像没有用

用例图 需要重新画

变更管理中子活动与产品间的输入输出关系 改一改可以用，有助于理解

变更管理及子活动聚集关系图 说明用

活动代理的聚集关系 写程序时用

配置项进行分类（归总），结合 安全UML分析目标 未完成

UML图需要做几个？

提出文本语义相似度计算的方法，用于对变更管理中，对变更是否合理进行一定程度的判断，

对于由于需求的改变而进行的变更，通过对相似度的的计算来判断变更的合理性。

对于需求变化而引起的变更是否基本符合变更管理中对于追溯性的要求。具体就是检查

例如 变更的起始项是 A1，经过更改新生成里a2, 受到影响的配置项包括 b1,b2,b3,b4等等，变更管理完成后，在新建立的基线中，b1,b2.,b3,b4….已不在该基线中，而该基线中应包含新生成的配置项 c1,c2,c3,c4…

1. 计算a1与b…..的相似度
2. 计算a1 与c。。。的相似度 是否存在相似度极低的情况，根据实际情况设置阈值，对于相似度低于该阈值的方法进行人工审查

## 文本语义相似度计算

本节主要介绍了通过引入

策略： 通过DF-id值，确定单词的权重

单词权重大的理论上讲应该是专业性词汇

方法：在0.5左右进行测算，看哪个个效果最好，在计算词与词之间的相似度时，若其中一方为 idf词，则计算出的相似度根据公式进行加成。

对于df-idf进行加权计算，对于缺失词，建立词汇表，他的相似度，等于“相关词汇”的相似度的平均值。但计算词汇间的相似度时，先确定是否为同一词。

基于DF-IDF的词与相似度加成的方法。

基于缺失词相似度计算方法的改进。

对于权重较大的词，若该词与被对比的文本相似对较大（大于0.5），则在计算时，将该词与被对比的句子的相似对继续加大。若该词与被对比的文本相似对较小，则在计算时，将该词与被对比的句子的相似对继续减小。

词与词？词与句子？控制词与词的比较方便。

单词加权策略和查询扩展方法来改进的文本语义相似度计算方法

并且为了满足3.1节中提到的适航领域软件变更中的文本特点，在使用单词加权策略时，用到的数据集合随着软件生命周期的不同而有所调整。

### 单词加权策略

为了区分句子中的单词的重要程度，我们选择对句子中的单词进行加权处理。本文使用TF-IDF的策略来计算，如公式(3.3)所示。TF（Term Frequency，词频）表示某个词在当前文档中的出现频率，如公式(3.1)所示；IDF（Inverse Document Frequency，逆文档频率）与含有某个单词或短语的文档数量有关，并且包含该单词或短语的文档数量越少，则该值越大，即表示当包含该单词或短语的文档数越少时，该词在当前文档中所占的权重越高，如公式（3.2）所示。当一个词或短语在一个文档中出现的频率较高，而在其他文档中出现频率较低时，则可以认为这个词或短语具有较强的类别区分能力，即能够比其他词更有代表当前文档的能力。

(3.1)

(3.2)

(3.3)

其中指单词在文档中的词频，分子表示在文档中该单词出现的次数，分母表示文档中单词的总数。指单词或短语在文档集合中的逆文档频率，分子表示文档总数，分母表示出现过此单词或短语的文档数，一般为了防止分母为0，会给该文档数加一，最后对比值取对数。

在当文档数量较少时，计算得到的tfidf值并不准确，而在适航领域，可能在软件需求相关文档尚未充足的情况下建立需求跟踪关系，因此此时需要进行数据补充。使用当前软件生命周期阶段已有的文档进行数据扩充，能够在一定程度上缓解tfidf计算不准确的问题。

### 查询扩展方法

在论文中，计算两个词之间的语义相似度我们首先将词表示为词向量的形式，然后使用cosine相似度计算，如公式(3.4)所示。

(3.4)

其中和分别表示词和的词向量，和分别表示两个词的词向量的长度。

合适的查询扩展方法和加权策略能够提升信息检索任务的性能[42]。对某个文档进行查询扩展，是对文档中的各个单词或词组进行扩展，步骤如图12所示，描述如下：

（1）首先计算TFIDF权重，然后根据TFIDF进行排序，选择前的单词或词组进行扩展，组成集合，其中参数设置为经验值0.3。

（2）对以上选择出的前的单词或词组进行扩展，使用单词语义相似度计算公式，选择出相似度大于阈值的单词或词组，每个被扩展的词组成集合，形式化的表示如公式(3.5)所示。其中参数同样设置为经验值0.7。

(3.5)



图12 查询扩展流程图

### 改进的文本语义相似度算法

在论文[16]中，文本语义相似度的计算有三个步骤：1）计算词之间的语义相似度；2）计算词与文本之间的语义相似度；3）计算文本之间的语义相似度。词与词的相似度计算如公式(3.4)所示，词与文本的相似度计算即为词与文本中所有词相似度的最大值，如公式(3.6)所示。文本与的相似度为文本中的每个词与文本的相似度的和，然后做归一化处理，如公式(3.7)所示。

(3.6)

(3.7)

(3.8)

其中表示文本中与文本的语义相似度不为0的单词或词组集合。

以上即为论文[16]中提出的方法，我们称之为W2V方法。相对于该方法，我们做出如下改进：

（1）在计算词与文本的相似度时，添加上查询扩展，于是重写了公式(3.6)，如公式(3.9)所示。添加上查询扩展后，同时还设置了参数来分配原词的相似度和扩展的词的相似度之间的权重，该值的取值范围为，在实际的计算过程中，该值在0.5到0.9之间，以0.01的步长增长，遍历整个范围，当效果最好时，设定值，因此参数根据数据集的不同而不同。如果当前单词没有扩展词，则使用与论文[16]中相同的方法，即使用公式(3.6)计算。

(3.9)

其中表示查询扩展项，即当计算词与词之间的相似度时，如果该词具有扩展词，则扩展项使用词的扩展词集合中的词与词之间的相似度之和，然后做归一化处理，保证的值的范围在0到1之间。举个例子，单词“technique”的扩展词有“technology”、“method”和“approach”等，当计算单词“technique”与句子“The basic requirement of planes is safety.”的相似度时，不仅仅计算“technique”与句子的相似度，同样会计算其扩展词“technology”、“method”和“approach”与该句子的相似度，最后使用比例参数将两者结合起来。

（2）在计算文本之间的相似度时，原方法的公式是不对称的，即对于同样的一对文本和文本，不同的输入顺序会产生不同的结果。为了保证一致性，以及结果与语句长短的无关性，本方法增加了一个限制，即在输入时使文本的长度长于文本。增加该限制主要有两个原因：第一是使得该公式是对称的，当计算任意一对文本的相似度时，使得结果与二者的前后顺序无关；第二是当较短文本是较长文本的子集时，避免较长文本中的其他词汇被忽视。第二个原因是一种特殊情况，举个例子：文本和文本分别是“Airworthiness is necessary.”、“This document emphasizes that airworthiness is necessary.”，在使用公式(3.9)计算时，文本中的每个词在文本中都出现了，因此在使用公式(3.7)计算词与文本的相似度时，文本中的每个词与文本中的词的相似度的最大值均为1，此时文本中的其他词被忽略掉了。为了避免这种情况，限制两个文本的前后顺序是重要的。

# 第四章 适航领域软件需求追踪算法模型验证

## 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。