基于组件的大数据分析服务平台

赵 薇[1,2] 刘 杰[2] 叶 丹[2]
(中国科学院大学 北京 100190)[1] (中国科学院软件研究所 北京 100190)[2]

摘 要 随着数据规模的快速增长,单机的数据分析工具已经无法满足需求。针对大数据的分析问题,设计并实现了一种基于组件的大数据分析服务平台Haflow。Haflow自定义了业务流程模型和可扩展的组件接口,组件接口支持各种异构工具的集成。系统接收用户定义的业务流程,将其翻译成执行流程实例,提交到Hadoop分布式集群上执行。Haflow是一个可扩展的、分布式的、支持异构分析工具的、面向服务的大数据分析服务平台。提出该平台有两重意义:一方面平台将与数据分析业务无关的工作封装起来,支持各种异构组件,以加快分析应用的开发速度;另一方面,平台后端使用Hadoop分布式系统来实现多任务的并发,从而提高应用的平均执行速度。

关键词 大数据,数据分析,数据挖掘,组件,分布式,服务,平台
中图法分类号 TP311.56 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.09.013

Module Based Big Data Analysis Platform

ZHAO Wei[1,2] LIU Jie[2] YE Dan[2]
(University of Chinese Academy of Sciences, Beijing 100190, China)[1]
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract As the expansion of data size, the data analysis tools that run only on stand-alone computers are no longer sufficient. We designed and implemented a module based big data analysis platform named Haflow to solve this problem. In Haflow, we defined an analysis business model and an extensible module interface, which supports integration of heterogeneous tools. Users submit their analysis flows, and system will interpret them, and then commit to the Hadoop. Haflow is an extensible, distributed, heterogeneous supported, service oriented big data platform. The goal of the platform is twofold. First, it provides an platform that encapsulates the dummy jobs that have nothing to do with the business itself, improving the development speed of analysis applications. Second, by submitting jobs to Hadoop which can execute jobs concurrently, the platform decreases the mean time of the analysis applications.

Keywords Big data, Data analysis, Data mining, Module, Distributed, Service, Platform

1 引言

随着互联网的快速发展,目前我们已处于数据、信息过载的海量信息时代。用户面对海量信息却很难找到自己真正感兴趣的内容。如何有效利用这些海量数据,从而产生实际的价值,已成为迫切需要解决的问题。数据挖掘就是一个能够把海量数据变成可被人类直接利用的信息、"把冰冷信息人性化"的强有力的工具。

目前市场上已经有一些很成熟的数据挖掘平台,如Weka、Clementine、R等。它们都集成了大量成熟、优秀的挖掘算法。但是随着输入数据量的增加,这些工具显得力不从心。对于大数据量的挖掘,它们可以采用先采样后挖掘的方法,但是这明显降低了挖掘结果的准确度。随着Hadoop掀起分布式计算热潮,分布式数据挖掘算法库Mahout[4]应运而生。虽然目前来看Mahout尚未十分成熟,但它却已经引起了业界足够的关注。结合MapReduce分布式计算在数据预处理上显示出的强大优势,分布式的数据处理、分析和挖掘已经成为一种趋势。Hadoop生态系统中的分布式存储和计算能力,结合Weka[3]、R[5]、Mahout等开源数据挖掘的分析能力,使得它们已成为很多公司和个人数据分析的有力工具。

为了解决上面的种种问题,本文设计并实现了一个基于组件的大数据分析服务平台Haflow。该平台主要以组件为设计单位,具有拖拽式流程设计界面,可以协同异构数据分析工具,具有分布式、服务化、易用性高、可扩展性强、开放式等特点。从前端来看,开发人员可以通过拖拽和关联组件来完成整个分析流程的设计。开发人员还可以通过组件管理界面发布自定义的组件,或者导入一组面向领域的可复用组件。从后端来看,Haflow构建在分布式集群上,并且集成多种数

到稿日期:2013-09-20 返修日期:2013-10-30 本文受国家自然科学基金(61202065,61170074),国家863计划(2012AA011204),国家科技支撑计划(2012BAH05F02)资助。

赵 薇(1988—),女,硕士生,主要研究方向为大数据分析、软件工程,E-mail:zhaowei11@otcaix.iscas.ac.cn;刘 杰(1982—),男,博士生,主要研究方向为软件工程、数据集成、数据质量;叶 丹(1971—),博士,正高级工程师,CCF高级会员,主要研究方向为分布式计算、中间件技术、企业应用集成。

据处理、分析和挖掘工具，并发地调度执行各个分析任务。前端通过调用后端系统提供的 REST API 来完成前后台的通信。

从数据分析工程师的角度来看，该平台可以屏蔽各个系统和工具的异构性，降低数据分析的难度，加快信息提取的速度。从企业成本的角度来看，首先大数据分析平台可供多个用户同时向集群提交多个流程，通过对这些多个流程的调度，不仅可以降低流程的平均执行时间，而且可以极大地提高集群的利用率。其次，拖拽式的开发极大地降低了对分析人员专业技能的要求，使得分析人员可以从算法编写、实验脚本编写等繁杂工作中抽离出来，帮助企业缩短决策的时间，改进决策的过程。总之，大数据分析服务平台不仅可以缩短公司投资回报周期，而且可以提高公司投资回报率。

本文首先介绍大数据分析服务平台的业务模型和系统框架，然后介绍系统模块间的交互方式以及实现中的关键技术，包括流程模型的映射、可扩展组件模型的定义以及中间数据的管理，最后做出总结并介绍下一步工作。

2 相关工作

目前有很多的可视化的数据挖掘工具，如 DMVisualMiner[8]、Weka 等。这类软件的主要问题是，它们主要面向的是小数据量的数据挖掘任务，在大数据量情况下存在执行时间长、精度低等缺点。

大数据挖掘的解决方案是采用分布式计算框架[9]，如 MapReduce、Pregel、Dryad 等。目前存在一些基于云计算的数据挖掘平台。文献[6]中的平台是基于 Google App Engine 的，数据通过网络传送给云服务器，这样的架构并不适合大数据量的情况。其次数据存储在开放的服务平台上，这对于很多公司和个人来说是不能接受的。本文描述的平台可以部署在公司内部，数据导入导出方式灵活，数据的私密性可以得到保证。文献[7]中提到的平台利用 BPEL 流程执行引擎。比较来说，本文中使用的 Oozie 流程管理引擎是专门针对 Hadoop 平台开发的流程管理引擎，具有更高的并发性和灵活性。

3 大数据分析服务平台

大数据分析服务平台 Haflow 为数据分析人员提供开放式的、可视化的分析应用开发平台。平台自定义了业务流程模型和可扩展的组件接口。每个业务流程由组件和有向数据流组成。平台接收用户定义的业务流程，翻译成执行流程实例，提交到 Hadoop 分布式集群上执行。

本节主要介绍数据分析业务流程模型的定义和平台的整体架构设计。

3.1 业务模型定义

一般来说，一个完整的数据分析任务可以划分为多个子任务，子任务之间会存在数据的传输和依赖。数据分析任务一般是针对特定应用场景和数据设计的，因此一般不具有可复用性。但是不同分析任务的子任务往往是可以复用的。本文定义了组件和流程的概念，组件对应分析任务的子任务，一个流程对应整个分析任务。我们定义一个可复用的子任务为一个组件，组件接收来自其它子任务的输入，对数据进行处理和转换之后，传输给其它子任务。每个组件可以有零到多个输入和零到多个输出。数据流程由一到多个组件组成，组件之间通过有向路径连接，表示数据的传输方向。

图 1 中显示的是一个组件，该组件包含了一个输入端点 IN1，两个输出端点 01，02。其中输入端点接收其他输出端点的输出，多个组件由有向路径连接。按照功能来划分，数据分析流程图的组件一般分为 4 种类型。

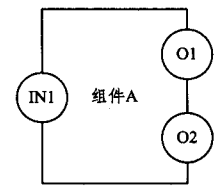

图 1 组件定义

数据资源组件：该组件为数据资源的一个抽象，数据资源可以是文本文件、二进制文件、文件夹、数据库等。一般来说该组件没有入端点，有多个出端点。

数据处理组件：该组件主要负责数据的抽取、转换、过滤等工作。通常来说数据处理组件接收来自数据资源组件的数据，并将处理后的数据输出给数据分析组件。

数据分析组件：该组件是核心功能组件，负责从数据中抽取知识。通用的分析组件一般包括聚类、分类、关联规则挖掘等。

数据展示组件：该组件负责将数据绘制成图像，使得用户可以更加直观地分析结果。

3.2 系统架构设计

平台整体遵循 MVC 框架，分为控制器(Controller)、视图(View)、模型(Model)。视图负责展示数据分析流程和系统中的组件。控制器负责调用基础服务响应用户请求及修改模型中的数据。系统服务的体系结构如图 2 所示。

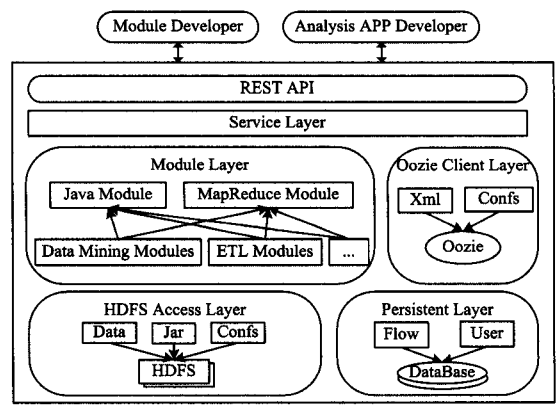

图 2 系统架构图

组件开发者和分析应用开发者通过系统提供的 Web 界面或者 REST API 访问系统服务。系统提供的服务主要基于 4 个基础服务层：组件管理层(Module Layer)、Oozie 访问层(Oozie Client Layer)、HDFS 访问层(HDFS Access Layer)、持久层(Persistent Layer)。控制器调用服务层(Service Layer)接口，并响应用户请求。服务层主要提供如下服务：

数据分析流程的翻译和执行。用户使用系统提供的数据分析组件，设计自己的数据分析流程，提交到 Haflow。

Haflow负责将用户定义的数据分析流程翻译成 Oozie，并通过 Oozie 的 Client 接口提交到 Hadoop 中执行。该系统支持多用户、多任务。每个用户可以连续提交多个任务，多个任务可在后台并发执行。

自定义组件(User Defined Module)。用户可以遵循一定的接口和规范，开发自定义的组件，提交到平台中。平台将扫描组件的元数据，将组件加载到系统中，之后该组件便可以被分析应用开发者看到并使用。

为了方便平台的开发和使用，系统还提供了一个基于 Web 的可视化流程设计界面。界面主要分为 3 部分：流程管理面板、流程设计面板、组件模块面板。用户通过将组件模板拖拽到流程设计面板形成分析流程，这里的流程称为用户定义的数据分析流程。用户提交该分析流程之后，系统会将该流程翻译称为 Oozie 的任务流程。具体的细节会在本文的第 4 节介绍。

4 系统实现及关键技术研究

在本小节中，主要研究系统的具体实现和关键技术。首先介绍系统模块交互，然后介绍流程模型映射，接下来介绍可扩展组件模型，最后介绍中间数据管理。

我们的系统中用的流程执行调度引擎是 Hadoop 开源生态系统中的 Oozie[1,2]。这是一款专门为 Hadoop 开发的可扩展的、基于多租户的、安全的、易用的工作流管理系统。

4.1 系统模块交互

大数据分析服务平台最主要的功能是将用户设计的分析流程翻译成 Oozie 执行任务，并提交到分布式集群上运行。Haflow 的系统组成及各个模块之间的交互关系如图 3 所示。从图中可以看到，系统后端由分布式集群构成，集群上装有 Hadoop 系统、流程执行引擎 Oozie、共享库、工具集等。Mahout 是基于 MapReduce 编程模型的分布式的数据挖掘算法工具包。Weka 是一款开源的单机版的数据挖掘软件，提供了丰富的数据挖掘算法。流程提交引擎负责将执行模型实例提交到集群中。执行模型实例由模型转换引擎从数据分析业务模型转换而来。数据分析业务模型是指用户在界面上设计的数据分析流程。执行模型实例指的是 Oozie 的一个工作流实例。组件库提供了流程模型中需要的元数据信息和必要的文件。

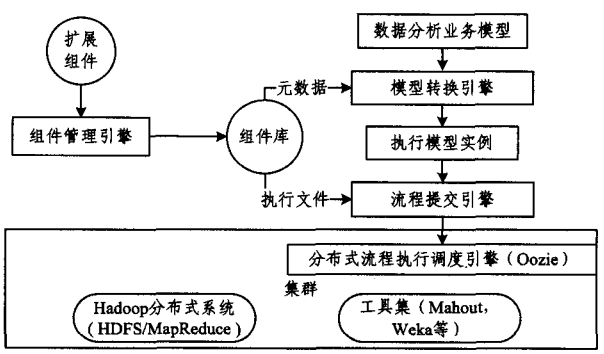

图 3 系统组成图

图 4 展示了系统的运作流程图。用户通过我们提供的可视化 Web 界面设计数据分析业务流程，发出“执行”请求。系统接收到请求后，将会完成接下来的各项工作。首先对任务进行有效性检查，流程合法则进行流程映射，映射成功之后生成配置文件。流程提交引擎收集流程需要的配置文件、数据和 Jar 包，提交到 HDFS 上面。最后由 Oozie 提交并监控任务执行。

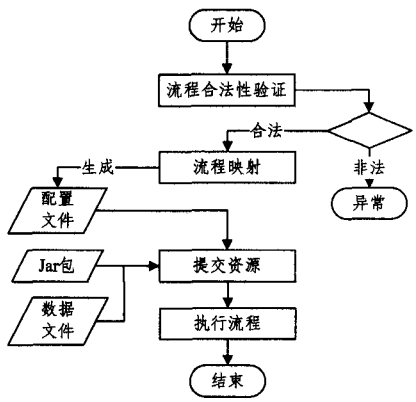

图 4 任务执行流程图

系统中最关键的部分是流程转换引擎和组件管理引擎，我们将在接下来的两个小节中分别对它们进行详细介绍。

4.2 流程模型映射

流程转换引擎的主要任务是将用户的流程分析业务模型转换成 Oozie 的执行实例。3.1 节中介绍了本文定义的业务模型，总体来说它是一个数据传输流，表征了各个组件间的依赖关系。Oozie 工作流是一个表示控制依赖关系的有向无环图(Direct Acyclic Graph，DAG)。每个 Oozie 工作流指定一组动作(例如 Hadoop 的 MapReduce 作业、Java 程序等)及其执行顺序，并使用 XML 描述这个图。

接下来介绍我们的转换算法。转换前重要的一步是检测用户的流程分析业务是否合法。首先判断各个节点的有效性，包括每个组件是否配置了完整的输入输出和配置信息等。然后判断整个图的有效性，主要是检测图中是否有环。

该转换算法有多种实现方式。首先是最简单的方式，也是目前系统原型中已经实现了的方式，我们称之为方法 A。方法 A 首先对业务模型中的图节点进行拓扑排序，得到一个串行的可执行顺序。然后将每个组件封装成一个 Oozie 的一个动作，增加开始节点、结束节点和异常节点。最后生成 Oozie 的配置文件。这种转换算法可以确保有效的业务模型能够转换成 Oozie 执行实例。但是该算法翻译成的任务是串行的。

另外一种翻译方式是对图进行层次遍历，同一层的图节点可以并发执行，我们称之为方法 B。其与方法 A 最大的不同是，方法 B 首先对业务流程图进行广度优先遍历，同时标记每个节点所在的层次。将每个组件封装成 Oozie 的一个动作。对于有多个组件的层，则在这些同一层的动作前增加一个 FORK 节点，同时在这些动作后增加一个 JOIN 节点。理论上，该算法执行速度会优于方法 A。

4.3 可扩展组件模型

可扩展的组件模型是 Haflow 的最主要的特色之一。用

户编写自己的算法，并添加少量的组件元信息便可以提交并集成到平台中。组件提交后用户便可以在分析流程设计时使用该组件。该算法的实现主要依赖于 Java 的反射和注解。

用户开发自己的组件需要完成两步，第一步是继承 AbstractModule 或者其子类，并实现 validate 方法。该方法用来验证用户提供的配置信息是否合法，默认实现是认为用户填入的信息全部合法，用户可以加入自定义逻辑。类的继承关系如图 5 所示。

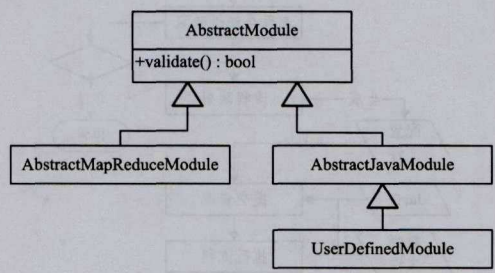

图 5　自定义组件类图

第二步是给自定义的组件添加元信息，元信息的结构如图 6 所示。配置信息包括组件模型的 ID、名称、类别、输入节点、输出节点以及用于显示的配置信息等等。组件的元信息是以注解的方式加到组件中的。因此图 6 中显示的是注解元素之间的关系图。

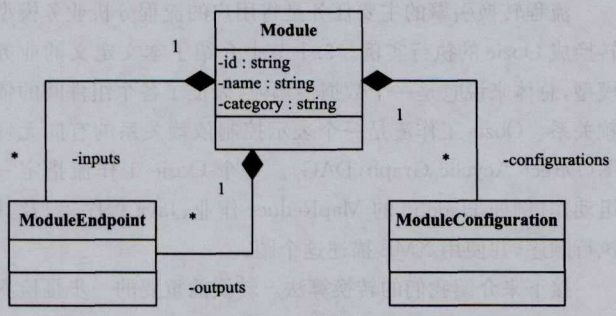

图 6　组件元信息结构

用户完成前面的两步之后就可以通过系统提供的接口将自定义的组件提交到平台中。平台收到组件之后，利用反射机制扫描组件中提供的元数据，将元数据记忆起来。当前端请求组件列表时，服务器会将新的组件列表返回。

4.4　中间数据管理

中间数据指的是除了用户输入的数据和流程结束后用户希望保留的数据之外的其它数据。大数据的分析任务会在流程执行期间产生大量的中间数据，而这些中间数据在流程设计和调试过程结束之后便失去存在的价值。平台需要对这些中间数据的生命周期进行管理。目前系统采用的策略是记录流程产生中间数据的存储位置，并在下一次同一流程执行之前删除之前的中间数据。

对中间数据进行管理还有一个很重要的步骤就是可以在流程执行前判断数据传递的有效性。本系统中将中间数据分为了多种类型。在流程验证步骤中，还需要验证一个数据传输路径的两端是否匹配。

5　案例分析

本节通过一个实际案例介绍大数据分析服务平台的基本功能和使用方法。

首先，我们利用 Mahout 工具包开发了 4 个组件，将其提交到平台中，界面效果如图 7 所示。

图 7　自定义 Mahout 组件

图 7 中一共展示了 4 个组件。CanopyCluster 封装了 Canopy 聚类算法、生成聚类模型。Describe 组件为数据列过滤组件，该组件会给数据的每一列打上一个标记，标记列中的数据类型和是否被过滤。BuildForest 封装了随机森林算法，接收来自数据源和 Describe 的数据，训练分类模型。TestForest 封装了随机森林的验证算法，接收来自 BuildForest 的模型数据和测试数据，产生测试数据。

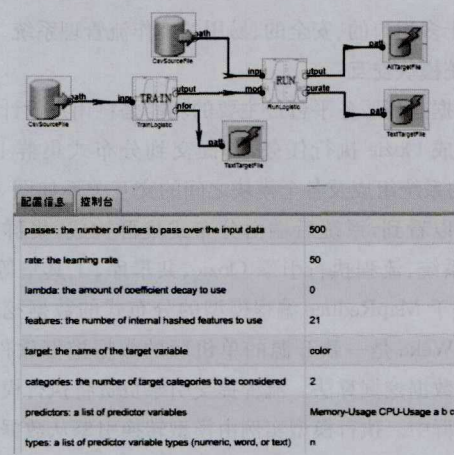

图 8　分类应用的界面效果

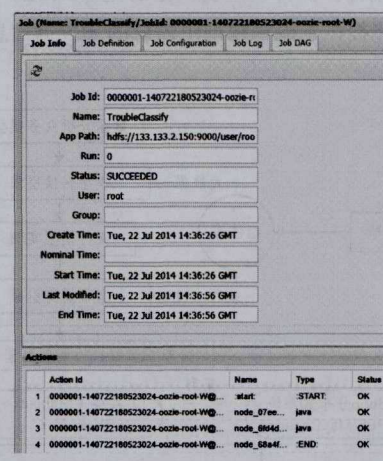

图 9　流程执行成功界面效果

图 8 展示了一个完整的分类应用。该应用使用到了上图

中的3个组件，并用有向路径连接了这些组件。点击“运行”，分析流程提交到系统翻译执行，执行成功后的状态如图9所示。

大数据分析服务平台将数据分析应用开发变成了两阶段的任务。首先是编写自定义组件，提交到系统中，然后利用这些组件快速开发数据分析应用。系统集成了很多共享开发组件，用户只需要开发应用特定的组件即可，极大地提高了分析应用的开发速度。

结束语 本文为了解决大数据分析问题，设计并实现了一个可扩展的、分布式的、支持异构分析工具的、面向服务的大数据分析服务平台的原型。着重介绍了该平台的设计与实现，包括系统的整体架构以及流程映射、可扩展组件接口定义和中间数据的管理。在4.2小节中介绍了两种模型转换算法，方法A得到的是串行的执行序列，方法B对方法A进行了优化。但是由于JOIN的语义，每层的执行速度取决于执行时间最长的任务，因此方法B并不是最优的方法。进一步的优化方案将会在我们后续的工作中进行研究。

参考文献

[1] Islam M, Huang A K, Battisha M, et al. Oozie: towards a scalable workflow management system for Hadoop[C]// Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies (SWEET'12). 2012, 4: 1-10

[2] http://oozie.apache.org/docs/3.3.2/index.html

[3] http://www.cs.waikato.ac.nz/ml/weka/

[4] http://mahout.apache.org/

[5] http://www.r-project.org/

[6] 纪俊. 一种基于云计算的数据挖掘平台架构设计与实现[D]. 青岛：青岛大学，2009

[7] 余永红，向晓军，高阳，等. 面向服务的云数据挖掘引擎的研究[J]. 计算机科学与探索，2012，6(1)：46-57

[8] 钱肖鲁，朱建秋，朱扬勇. DMVisualMiner：一个可视化数据挖掘分析平台[J]. 计算机工程，2003，29：148-150

[9] 丁岩，杨庆平，钱煜明. 基于云计算的数据挖掘平台架构及其关键技术研究[J]. 中兴通讯技术，2013，19(1)：53-60

(上接第51页)

4 结论

4.1 总结

本文首先对主流的i* 建模工具进行了调研，研究了建模工具的基本功能，同时分析了其功能的不足，并在此基础上提出了新工具设计的功能补充点，给出了可视化工具的详细设计和实现。

i* 建模框架作为非常重要的需求工程分析框架，需要一个比较好的编辑工具。而通过我们的调研可以看到现在存在的工具不论是在功能上还是在用户体验性能上都趋于陈旧和粗糙。我们使用了富客户端技术的Flex进行开发，使得工具的用户体验大大提升，同时提供了云存储、自动布局等先进的功能，实现了一个较好的基于i* 框架的建模工具。

4.2 展望

虽然Lerisk已经是比较成型的一个i* 框架建模工具，但它仍然存在一些不足和缺陷，这些不足可以是对之完善的下一步目标。

1. 对于策略推理模型的自动布局功能

由于策略推理模型的主体元素和其它元件可能拥有从属关系，因此在进行布局时要考虑它们的包含关系。可以考虑将整个主体视作单位进行布局，再以对主体内部的元件进行布局的方式来进行实现，这可能是一个比较好的方向。

2. 更加完善的编辑功能

Lerisk虽然可以成功地实现整个流程的编辑功能，但在一些方面还有需要完善的空间，譬如元件的复制和粘贴、连接曲线的弯曲度调整等。

3. 性能的优化和提升

在处理的模型图非常庞大时，我们发现Lerisk效率会变慢，因此关于自动布局的算法及界面的绘制有进一步优化的空间。

希望这些尚未完善的问题能够在以后工作中得到解决，使Lerisk成为一个愈发成熟的建模工具。

参考文献

[1] 金芝，刘璘，金英. 软件需求工程：原理与方法[M]. 北京：科学出版社，2008

[2] Alonso O, Strogen J, Baeza-Yates R, et al. Temporal information retrieval: Challenges and opportunities[C]// Proceedings of the 1st International Temporal Web Analytics Workshop (TWAW) 2011. Hyderabad, India, 2011: 1-8

[3] Alonso O, Gertz M, Baeza-Yates R. On the value of temporal information in information retrieval[J]. ACM SIGIR Forum. ACM, 2007, 41(2): 35-41

[4] Schilder F, Habel C. From temporal expressions to temporal information: Semantic tagging of news messages[C]// Proceedings of the workshop on Temporal and spatial information processing-Volume 13. Association for Computational Linguistics, 2001: 9

[5] Pustejovsky J, Castano J M, Ingria R, et al. TimeML: Robust Specification of Event and Temporal Expressions in Text[J]. New directions in question answering, 2003(3): 28-34

[6] Dias G, Campos R, Jorge A. Future retrieval: What does the future talk about[C]// Proceedings of Workshop on Enriching Information Retrieval of the 34th ACM Annual SIGIR Conference, SIGIR. 2011

[7] Baeza-Yates R. Searching the future [C]// SIGIR Workshop MF/IR. 2005

[8] Kulkarni A, Teevan J, Svore K M, et al. Understanding temporal query dynamics[C]// Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011: 167-176

[9] Verhagen M. TimeML Corpora [DB/OL]. [2013-01-10]. http://www.timeml.org/site/publications/timeMLdocs/timeml_1.2.1.html

[10] Pustejovsky J. Task 15: TempEval Temporal Relation Identification [C/OL]. [2013-01-10]. http://timeml.org/tempeval/