УДК 004:896;031.42;622;623;624

Т. Шатовская, И. Каменева, А. Гуд
Харьковский национальный университет радиоэлектроники

РЕПОЗИТАРИЙ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Используя агентные технологии, и, в частности, технологии агентно-ориентированного моделирования, предоставляется возможность исследовать и создать интернет-приложение нового типа – Репозиторий интеллектуального анализа данных (Data Mining Repository).

Using agent technology, and in particular, agent-based technology model, the opportunity to explore and create a new type of Internet application - Data Mining Repository.

Введение

Хранилище научных и статистических наборов данных из различных предметных областей всегда представляли большую ценность для научных работников, преподавателей, студентов. Хранилище может помочь ученым поддерживать и обосновывать свои эксперименты в области интеллектуальной обработки данных. Для информационной поддержки исследований и разработок в области Data Mining было предложено разработать тематический репозиторий «Data Mining Repository». Мы надеемся, что создание такого репозитория будет объединять людей вокруг одной любимого дела, профессии, увлечения, позволит делиться мнениями, давать и получать советы, рекомендации в области Data Mining и Maching Learning, а также в области статистических исследований.

Основная идея нашей работы состоит в том, чтобы используя мультиагентную технологию и онтологическое описание предметной области создать хранилище научных наборов данных, которое будет позволять хранить данные, осуществлять поиск, обработку данных в предметной области интеллектуального анализа данных. На основе сочетания онтологических моделей и агентних технологий мы получили новую модель хранилища научных наборов данных, что позволило в десятки раз увеличить эффективность работы с данными.

Онтологические модели репозитория

Важным моментом при написании Semantic web приложений является выбор способа хранения онтологий, так как от этого зависят такие важные характеристики программной системы как производительность, расширяемость, способность взаимодействия с другими системами. Следует понимать, что онтологии могут быть очень большими - в некоторых из них помимо сложной иерархии с множеством классов и свойств, могут храниться миллионы объектов. В совокупности с ситуацией, когда в одной базе хранится множество онтологий, скорость выполнения запросов к базе данных сильно уменьшается. Обычно база данных является составной частью большого проекта, в котором обращение к базе происходит довольно часто, и от скорости выполнения запросов к базе данных напрямую зависит скорость работы программы в целом.

Используя агентные технологии, и, в частности, технологии агентно-ориентированного моделирования, предоставляется возможность исследовать и создать интернет приложение нового типа. Агентно-ориентированное моделирование обеспечивает средства для концептуализации и понимания процессов, происходящих в Интернет.

Онтологический анализ обычно начинается с составления словаря терминов, который используется при обсуждении и исследовании характеристик объектов и процессов, составляющих рассматриваемую систему, а также создания системы точных определений этих терминов. Кроме

того, документируются основные логические взаимосвязи между соответствующими введенным терминам понятиями. Т.е. нужно делать различия между понятиями и терминами. Результатом такого анализа есть онтология системы, или же совокупность словаря терминов, точных определений и их взаимосвязей между ними.

Таким образом, онтология включает в себя совокупность терминов и правила, согласно которым они могут быть скомбинированы для построения достоверных утверждений о данной системе в некоторый момент времени. Кроме того, на основе этих утверждений, могут быть сделаны соответствующие выводы, позволяющие вносить изменения в систему для повышения эффективности ее функционирования.

Проанализировав термины предметной области было виделено три класса и набор слотов в онтологичской моделе ресурса: DataSet; DataSetFile; Judge. В табл. 1 представлено описание слотов класса DataSet – это описание непосредственно файла с набором данных.

Таблица 1

Слоты класса DataSet

Атрибут	Тип	Потужність	Наявність	Опис
Abstract	String	Single	Mandatory	Введення (короткий опис)
AnalysisMethod	String	Multiple	Mandatory	Метод аналізу (посилання на елементи онтології методів аналізу даних)
Area	String	Single	Mandatory	Предметна область
AttributeAmount	Integer	Single	Mandatory	Кількість атрибутів
AttributeInfo	String	Single	Mandatory	Інформація про атрибути
AttributeType	String	Multiple	Mandatory	Тип атрибутів
CitedPaper	String	Multiple	Optional	Статті, що посилаються на набір даних.
Creators	String	Multiple	Mandatory	Список розробників вибірки: посилання на елементи онтології користувача
DataSetInfo	String	Single	Optional	Опис набору даних
DataType	String	Multiple	Mandatory	Тип даних
DateDonated	String	Single	Mandatory	Дата завантаження даних або остання дата обновлення
DownloadAmount	Integer	Single	Mandatory	Кількість скачувань
DSFiles	Instance of DataSetFile	Multiple	Optional	Файли вибірок даних
InstanceAmount	Integer	Single	Mandatory	Кількість елементів у вибірці
KeyWord	String	Multiple	Optional	Ключеві слова
RelevantPapers	String	Multiple	Optional	Релевантні статті
SolutionMethods	String	Single	Optional	Метод рішення
Status	String	Single	Mandatory	Статус набору даних (нова, низький, середній, високий)
DatasetMark	Float	Single	Optional	Середня оцінка набору даних
Title	String	Single	Mandatory	Назва

Для построения данной онтологической модели была использована система Protege, основанная на фреймовой модели представления знанй OKBC (Open Knowledge Base Connectivity) и оснащенная рядом плагинов, что позволяет адаптировать его для редактирования моделей в разных форматах (стандартный текстовый, базы данных JDBC, UML, языков XML, XOL, SHOE, RDF і RDFS, DAML + OIL, OWL). После определения онтологической модели ресурса Protege позволяет конвертировать проект в RDF-модель. При конвертации было задано пространство имен http://dmr.kture.ua/dataset/. Часть полученной RDF модели показана на рисунке 1.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
    <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
    <!ENTITY dataset 'http://dmr.kture.ua/dataset/'>
    <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
    xmlns:dataset="&dataset;"
    xmlns:rdfs="&rdfs;">
<rdf:Property rdf:about="&dataset;Abstract"
    rdfs:label="Abstract">
   <rdfs:domain rdf:resource="&dataset;DataSet"/>
   <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&dataset;AnalysisMethod"
    rdfs:label="AnalysisMethod">
   <rdfs:domain rdf:resource="&dataset;DataSet"/>
   <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

Рис. 1. Часть RDF модели ресурса

Также для проектируемой системы были разработаны онтологии пользователя системы и онтологии методов анализа данных. Онтология методов представляет собой токсономию всех методов анализа данных. Онтология пользователя имеет два абстрактных класса: Account и Person. Класс Account представляет пользователя как логическую сущность пользователя системы. Класс Person представляет пользователя как человека, который использует данную систему. Реальное значение RDF невозможно оценить, пока оно используется для внутренних целей отдельно взятой программы. Польза от внедрения RDF будет тогда, когда это станет средством межпрограммного взаимодействия, обмена данными, когда машины получат способность комбинировать информацию, полученную из разных источников, тем самым, получая какую-то новую информацию.

Полученная нами RDF-модель представляет собой метаданные экспериментальных выборок данных, что позволило нам далее разрабатывать мультиагентную систему, опираясь на работу с метаданными выборок.

Следует понимать, что онтологии могут быть очень большими – в некоторых из них помимо сложной иерархии с множеством классов и свойств, могут храниться миллионы объектов. В совокупности с ситуацией, когда в одной базе хранится множество онтологий, скорость выполнения запросов к базе данных сильно уменьшается. Так как обычно база данных является составной частью большого проекта, в котором обращение к базе происходит довольно часто, то от скорости выполнения запросов к базе напрямую зависит скорость работы программы в целом. В связи с этим можно выделить несколько типов запросов, скорость которых должна быть высокой, независимо от сложности структуры онтологии и количества объектов в ней: определение степени сравнения классов (необходимо определить для двух классов, является ли один из них предком / потомком другого); получение всех объектов класса; отбор объектов по значению свойства; проверка на нужное значение свойств.

СУБД Oracle 10g включает поддержку RDF / RDFS, давая возможность разработчикам программного обеспечения использовать преимущества платформы семантически организации данных. Прикладные разработчики могут дополнять значения в данных и метаданных, определяя новые наборы условий и отношений между ними. Эти наборы условий ("онтологии") более приспособлены для осуществления запросов и анализа, основанного на семантическом подходе, чем обычные наборы данных. Онтологические наборы данных часто содержат миллионы элементов данных и отношений между ними, которые могут быть сгруппированы в триплеты, используя новую RDF модель данных. Oracle допускает расширение биллионов триплетов для удовлетворения

требований большинства приложений: RDF данные хранятся как направленный, логический граф; субъекты и объекты отображаются как узлы, а предикаты как связи, в которых субъект является начальным узлом, а объект конечным; связи представляют из себя полный RDF триплет; RDF модель данных поддерживает три типа объектов базы данных; модель (RDF граф, состоящий из набора триплетов).

Традиционная технология процесса организации поиска информации в базах данных информационной системы предусматривает - персональное обращение пользователя по сети Интернет к Data Mining Repository серверу с запросом, формирование из полученных ответов сводного результата и его последующая обработка. Выполнение вообще рутинных операций может отнимать у специалистов достаточно много рабочего времени. В связи с этим возникает еще одна проблема - разработка агентной системы для автоматизации процессов исполнения запросов в информационной системе, которая взяла бы на себя большую часть рутинных операций по организации поиска информации в базах данных нашей системы.

Общая структура системы представлена на рис. 2.

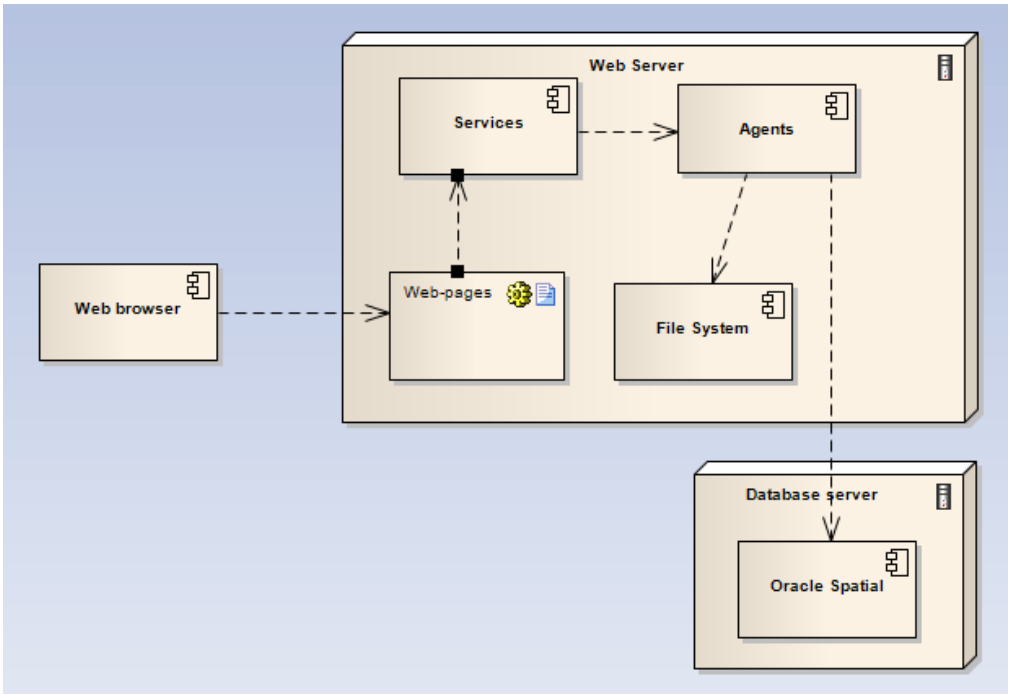

Рис. 2. Структура системы

Предложенная система состоит из следующих уровней: презентационный уровень, сервис-уровень, уровень агентов, база данных. Система «Data Mining Repository» представляет собой сервис-ориентированную архитектуру, которая соответствует принципам многоразового использования функциональных элементов, ликвидации дублирования функциональности в программном обеспечении, унификации типовых операционных процессов, обеспечению перевода операционной модели на централизованные процессы и функциональную организацию на основе промышленной платформы интеграции. Компоненты программы могут быть распределены по разным узлам сети, и предлагаются как независимые, слабо связанные, которые могут заменять сервисы-приложения. Разработанная система реализована как набор веб-сервисов, интегрированных с помощью протоколов SOAP и WSDL. Интерфейс компонентов программы предоставляет инкапсуляцию деталей реализации конкретного компонента от других компонентов. Таким образом, эта архитектура предоставляет гибкий и элегантный способ комбинирования и многократного использования компонентов. Для обработки запросов, поступающих от пользователей с Web интерфейса, были использованы агенты системы. Система поддерживает четыре Web сервиса:

– Administration Service;
– Search Service;
– Profile Service;
– Resource Service.

Для разработки веб-сервисов было применено решение xfire. Xfire – это свободное решение, обеспечивающее проблему совместимости, реализующее различные проблемы промышленных стандартов. Для разработчиков распределенных приложений – это простейший механизм реализации удаленных вызовов.

Агентная часть системы

Для реализации поставленной концепции было создано несколько агентов из основных функций хранилища - агент поиска, персональный агент, агента ресурса, что позволит получить систему, которая обладает следующими свойствами: данные, механизмы управления, знания и ресурсы распределены; система естественным образом представляется как совокупность автономно сотрудничающих компонентов; система содержит унаследованные компоненты, которые должны взаимодействовать с другими, возможно новыми программными компонентами. Каждый агент представляет BDI (Знание-Желание-Намерения) модель с определенным планом убеждений, целей, мероприятий и т.д. В системе Data Mining Repository все агенты, входящие в мультиагентной систему относятся к одному из следующих типов: Агент-менеджер, который работает на сервере и координирует работу пользователей; Агент пользователя, который осуществляет взаимодействие с пользователями; Агент ресурса, отвечающего за операции с наборами данных; Агент поиска, который осуществляет поиск информации.

Таким образом, даже если агенты будут размещены на разных серверах, то возможно их взаимодействие при запросах пользователей. К серверной части мультиагентной системы входят агенты ManagerAgent, ProfileAgent, ResourceAgent, SearchAgent. Обмен сообщениями между агентами базируется на протоколе HTTP, а работа с базой данных осуществляется по протоколу JDBC.

Рассмотрим более подробно агента ресурса. Основными функциями агента ресурса являются: добавление научных наборов данных; взаимодействие с агентом пользователя для отображения пользователю недавно добавленных выборок в зависимости от интересов пользователя. После добавления нового набора в хранилище агент ресурса информирует агента пользователя о добавлении научном наборе данных для отображения пользователям информации об этом; редактировании метаданных научных наборов данных, редактировать предоставляется возможность пользователям, которые являются создателями или администратору; выбор метаданных всех наборов данных из хранилища; выбор всей информации о конкретном наборе данных, подробную информацию могут просматривать только зарегистрированные пользователи; установление статуса выборки в зависимости от оценок, число загрузок скачивания выборки. Оценка может быть выставлена каждому набору данных. Оценка присваивается с учетом коэффициента профессионализма пользователя, который ее устанавливает. В момент выставления оценки его оценка умножается на коэффициент. Эту функцию выполняет агент ресурса. Агент ресурса должен получить у агента пользователя коэффициент, высчитать результат и сохранить его в базе данных.

Статус выборки может повышаться в зависимости от количества загрузок.

Взаимодействие с агентом пользователя для модификации коэффициента профессионализма пользователя в зависимости от статуса научных наборов данных, которым он ставил оценку или добавлял в хранилище;

– Фильтрация наборов данных по определенному параметру с мета данных выборок; добавление новых данных в репозиторий научных наборов данных, которые нашел в Интернете агент поиска.

В качестве примера рассмотрим последовательность добавления нового набора данных в систему. Зарегистрированный пользователь должен зайти на страницу Create New Dataset. Заполнив все необходимые поля формы, нужно нажать на кнопку Insert, Web страница вызовет функцию

сервиса ресурса на придание нового статистического набора данных до репозитория (рис. 2). Сервис вызывает агента ресурса на выполнение плана добавления нового набора данных в репозиторий. После добавлвения агентом данных в базу данных агент вызывает агента менеджера для поиска всех пользователей с предпочтениями, которым соответствует только что добавленная выборка.

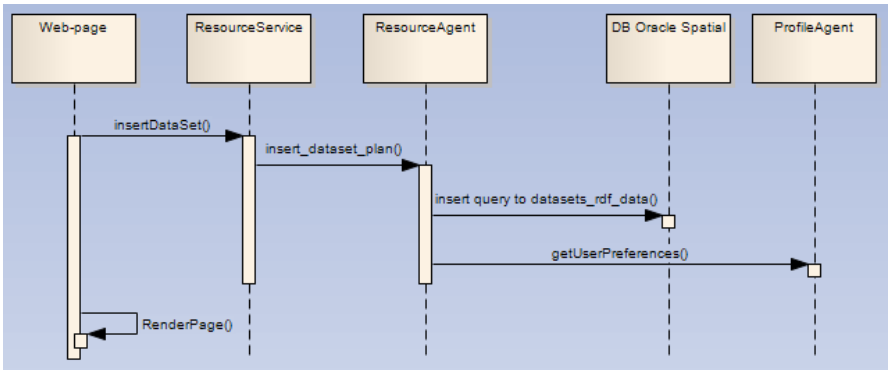

Рис. 3.

Конечно при большом количестве выборок запрос будет долго выполняться и соответственно будет долго выполняться рендеринг страницы. Поэтому для оптимизации этого процесса необходимо применить пейджинг как на странице, так и с помощью запроса. Рассмотрим один из планов. Последовательность действий в ReadDatasetPlan следующая: из целей устанавливается параметр плана «имя набора данных». При старте плана эти параметры используются для построения запроса к базе данных ресурса. Если набор не был найден, то план устанавливает соответствующий результат в исходный параметр. Если набор данных найден, то в представлении агента добавляется онтологически модель ресурса для дальнейшего ее использования. Для передачи данных между сервером и ResourceAgent используется сетевое соединение - сокет. Интерфейс сокетов позволяет передавать данные между двумя приложениями, работающими на одном или разных узлах сети. Сокет создается как объект класса Socket, с указанием хоста серверного приложения и номера порта, используемого сервером.

server_socket = new Socket (server_name, server_port);

Далее создаются входящий и исходящий потоки для обмена информацией. На стороне клиента эта операция выполняется точно также как и на стороне сервера.

server_receive = new BufferedReader (new InputStreamReader (server_socket.getInputStream ()));
server_send = new PrintStream (server_socket.getOutputStream ());

При успешном соединении ResourceAgent передает серверу данные и команды sql-запрос. Эта команда означает.

Если передача информации серверу происходит нормально, то агент сообщает пользователю о том, что его запрос принят системой для обработки, т.е. выдает результат. Затем он завершает свою работу, закрывая сетевое соединение.

finally (if (server_receive! = null) server_receive.close ();
if (server_send! = null) server_send. close ();
if (server_socket! = null) server_socket. close ();)

Как уже было отмечено, серверная часть мультиагентной системы реализована на Java. При этом ResourceAgent осуществляет связь с интерфейсом, с другими агентами репозитория и другими серверами системы.

Такая четырех уровневая архитектура репозитория дает возможность взаимодействовать с разработанным репозитории на уровне сервисов, является очень актуальной практикой в наше время, и на уровне агентов, то есть возможным взаимодействие агентов различных систем.

Вывод

Была разработана система «Data Mining Repository», которая представляет собой сервис-ориентированную архитектуру. Она состоит из следующих уровней: презентационный уровень, сервис-уровень, уровень агентов, база данных.

Было создано несколько агентов из основных функций хранилища. Данная система актуальна и дает возможность взаимодействовать с различными системами с помощью интеллектуальных агентов.

1. Web Ontology Language (OWL) OWL Web Ontology Language Overview. W3C Recommendation. – February 2004. 2. Resource Description Framework (RDF) Model and Syntax Specification. W3C Proposed Recommendation. – January 1999. 3. Dubes R.C. and Jain A.K. Algorithms for Clustering Data / Prentice Hall, 1988. 4. Blake, C.L. & Merz, C.J., 2000. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html].

УДК 004.75+519.612.2

Д. Федасюк, П. Сердюк, Ю. Семчишин
Національний університет "Львівська політехніка",
кафедра програмного забезпечення

АДАПТИВНИЙ АЛГОРИТМ ІЄРАРХІЧНО РОЗПОДІЛЕНОГО РОЗВ'ЯЗУВАННЯ СЛАР ВЕЛИКОЇ РОЗМІРНОСТІ

У контексті систем розподілення обчислень адаптивністю називають здатність таких систем продовжувати працювати та повністю використовувати обчислювальні ресурси після зміни конфігурації мережі чи швидкодії окремих її вузлів. Предметом цього дослідження є реалізація адаптивного підходу до ієрархічно розподіленого розв'язування систем лінійних алгебраїчних рівнянь великої розмірності. У роботі сформульовано задачу адаптивного підходу, розроблено та досліджено алгоритм та програмну реалізацію адаптивного підходу. Результати серії експериментів підтвердили ефективність адаптивного підходу.

In the context of the distributed computing systems, adaptability is an ability of such systems to continue functioning and fully utilizing computing resources even after changing network configuration or performance of its nodes. The subject of this study is to implement an adaptive approach to hierarchically-distributed solving of high dimensional linear algebraic equations. The problem of an adaptive approach is stated, the algorithm and the software implementation of an adaptive approach are researched and developed in the work below. The experimental results proved the effectiveness of an adaptive approach.

Вступ

Значну кількість задач математичної фізики можна розв'язати за допомогою систем диференційних рівнянь у часткових похідних. Розв'язування таких систем після їхньої лінеаризації числовими методами зводиться до розв'язування систем лінійних алгебраїчних рівнянь (СЛАР).