- редактор категорий, позволяющий создавать/редактировать категории и устанавливать связи между ними;
- редактор метаданных, позволяющих включать документы в нужные категории;
- визуализатор семантической карты категорий;
- и прототип семантического браузера, позволяющего извлекать метаинформацию и автоматически находить документы, близкие по смыслу.

Таким образом, в данной работе рассмотрены основные достоинства и недостатки семантической сети как метода представления связанных понятий, проанализировано использование семантических технологий в сети Интернет, показано, что интеграция информационной сети связанных документов (например, сети Интернет) и семантической сети понятий позволит получить синергетический эффект. Предложена технология организации семантической сети документов на базе языков разметки HTML и XML. Представлен проект информационной системы работы с научной информацией с возможностью ассоциативного поиска. Опытную эксплуатацию и тестирование системы планируется осуществить на базе кафедры общего, стратегического, информационного менеджмента и бизнес-процессов КубГУ в начале 2011 г.

Литература

1. Егоров А. Искусственное сознание (Сети: нейронные или семантические?). URL: http://www.robo-homo.ru/robo-lenta/robo-debates/136.html.

2. Бернерс-Ли Т., Хендлер Дж., Лассила О. Семантическая Сеть: пер. с англ. URL: http://ezolin.pisem.net/logic/semantic_web_rus.html.

3. Volkel M., Krotzsch M., Vrandecic D., Haller H., Studer R. Semantic Wikipedia. URL: http://www2006.org/programme/files/xhtml/4039/xhtml/fp4039-voelkel.html.

СИСТЕМА СОПРОВОЖДЕНИЯ ИГРОВОГО ОБУЧЕНИЯ

И.С. Игнатьев
Московский государственный институт электроники и математики (технический университет)
E-mail: ivan.ignatyev@auditory.ru

Общая концепция системы

Даже при наличии возможности использования аппаратных и программных средств, прежде доступных только на производстве, многие студенты ею не пользуются в силу сложности реальных и более высокой ответственности за их выполнение. Данные проблемы решаются современными методами обучения: активными методами и игровыми методами [1, 2]. На базе этих методов для решения этой проблемы было решено создать компьютерную систему сопровождения игрового обучения для обучения ИКТ (СИО). Для выполнения задания при использовании инструментария проектирования игровых проектов была разработана методика игрового обучения ИКТ на инженерных специальностях. В ходе обучения студенты развиваются в рамках своих ролей – будущих профессий – на основе постоянной автоматической – при использовании анализа поступающих в систему данных о прогрессе студента в определенной игре – оценки различных действий студентов. В системе возможны, как компьютерные игры – в этом случае данные собираются также автоматически при помощи клиентского программного обеспечения системы, так и другие виды игровой деятельности – в этом случае требуется ввод данных в систему ручным путем. Благодаря постоянной оценке и развитию студент чувствует себя постоянно задействованным и стремится максимально раскрыть свой потенциал [3]. Гибкость методике придает ориентация не на непосредственно на оценку студента в качестве актора какой-либо профессии, а оценку навыков студента, которые в совокупности показывают общий уровень развития студента, что оценивается в уровне развития в профессии как в агрегированном параметре. Также разработанная методика позволяет на основе навыков студента автоматизированно составлять его резюме, что повышает ее полезность для студента.

Система позволяет наблюдать обучение (прежде всего, в группе) и выявлять неявные его закономерности. Общий алгоритм работы системы следующий: пользующиеся системой в обучении используют специально модифицированные для системы игры, которые периодически отправляют отчеты о прогрессе обучающегося в единую базу данных, по материалам которой и отслеживается различными интеллектуальными методами анализа данных общий прогресс обучения, и строятся и проверяются различные предположения об их обучении.

Архитектура СИО

С архитектурной точки зрения СИО представляет собой систему с многозвенной (трехзвенной) клиент-серверной архитектурой. Серверами системы являются:

- сервера сбора результатов обучающих игр S_g;
- сервера анализа результата обучающих игр S_a;

- сервера хранения результатов обучающих игр S_m;
- сервера представления результатов анализа S_w.

Сервера хранения результатов обучающих игр S_m являются серверами баз данных, остальные сервера являются серверами приложений, так как несут различную функциональную нагрузку в рамках СИО.

Клиентами СИО являются клиент на машине пользователя С, который обеспечивает сбор результатов игрового обучения, и интернет-браузер, с помощью которого пользователь получает доступ к аналитической информации, полученной по результатам игрового обучения.

Серверная часть системы

Сервер получает данные, расшифровывает их и упорядоченно помещает в соответствующие таблицы базы данных. По мере накопления данных появляется возможность их анализировать, для этого сервер предоставляет свои инструменты анализа данных. Результат анализа должен быть представлен пользователю.

Получение и проверка данных

Данные в целях ускорения передаются по ненадежным каналам и могут исказиться, и соответственно требуется не только их принять, но и как можно быстрее сохранить. Для этого после получения пакета требуется как можно скорее его проверить на соответствие формальным требованиям со стороны протокола и, опять же с наименьшими затратами по возможности, далее сохранить в постоянное хранилище данных.

Для того чтобы реализовать проверки, которые позволят отбрасывать поврежденные пакеты, следует предусмотреть нужный механизм. Так, для этого можно использовать проверяющие парсеры xml-кода. Также это позволит прозрачно перейти от представления данных в передаваемом через сеть пакете к представлению данных в их хранилище [4]. Для устранения возможности блокировки приема информации прием является максимально асинхронным. Для этого для каждого пакета следует не обрабатывать его в основной программе приема, а порождать дочернюю программу, которая уже будет обрабатывать пакет, проверять его корректность и сохранять в базе данных [5]. Также для того, чтобы сохранение в базу данных не стало узким местом, используется асинхронное сохранение.

Сервер сбора результатов обучающих игр Sg реализован при помощи фреймворка Twisted, который позволяет организовать асинхронную обработку пакетов с сохранением их в СУБД [6]. Был создан запускающийся независимо от основного комплекса демон. Созданы скрипты для его запуска в ОС Debian Linux, подготовлен установочный файл. Конфигурация демона производится при помощи файла конфигурации, находящегося в специально выделенном для этого каталоге.

Хранение данных

Данные необходимо хранить таким образом, чтобы их можно было достаточно легко считать, и в то же время они не занимали на диске много места. В связи с этим следует отойти от традиционного хранения данных в достаточно быстрых, но неэкономных в смысле места хранения типов хранилищ, таких как MyISAM и InnoDB, в пользу специализированного хранилища, которое предназначено для долговременного хранения больших объемов данных – Archive [7].

Сервер хранения данных представляет собой структуру таблиц, основанную на нескольких различных типах хранилищ. В хранилищах типа InnoDB хранятся данные, которые требуют оперативного доступа и работы с ними: конечные данные анализа, исходные данные для визуализации, учетные данные пользователей, различного вида описания для игр, алгоритмов и т.п., которые в основном обслуживают сервер представления результатов анализа Sw. В хранилище типа Archive хранятся исходные данные (результаты игр), предназначенные для анализа. Их выборка происходит сразу большим объемом, что позволяет пренебречь оперативностью в пользу массовости.

Анализ данных

Анализ данных – это требовательный к таким ресурсам, как оперативная память и вычислительная мощность, процесс. Он должен запускаться над выбранными данными отдельно от всех остальных процессов, чтобы не вызвать никаких перегрузок системы. Более того, это сложный и долго выполняющийся процесс, а потому его запуск как веб-скрипта сопряжен с большими сложностями. В силу этих причин анализ данных физически выделен из общего функционала серверов в качестве отдельного сервиса, тем более что вполне можно выделить как входные, так и выходные данные для этого сервиса. В качестве входных данных выступают данные о том, какой алгоритм, с какими параметрами и ограничениями и над какими данными следует выполнить. В качестве выходных данных – запись в таблице результатов анализа игр и визуальное представление результата в виде диаграмм Вена и других визуальных средств.

Общая схема работы сервера анализа заключается в следующем:

- Получив задание на анализ от пользователя системы через сервер представления результатов анализа, сервер анализа проверяет его корректность и достаточность представленных в нем данных.
- После этого сервер запускает на основании полученного задания долговременный процесс, который будет осуществлять сам анализ данных непосредственно.

- В том случае, если для выполнения процесса анализа данных не хватает ресурсов или превышен интервал ожидания ответа, сервер анализа данных возвращает ошибку серверу представления результата.
- По завершении процесса анализа данных сервер анализа данных сохраняет представленные конкретными алгоритмами результаты анализа в стандартизованном виде в БД сервера представления результатов анализа.

В результате работы сервер создает в БД сервера представления результатов анализа структуру, на основании которой сам сервер представления отображает результаты анализа данных.

Для анализа данных предусмотрено несколько методов анализа.

Байесовский классификатор [8] позволяет определять вероятность принадлежности наблюдения или объекта к одному из классов. При этом выдвигается предположение о независимости влияния на эту вероятность различных атрибутов объектов – так называемое предположение об условной независимости классов, которое существенно упрощает сопутствующие вычисления.

Простой байесовский классификатор относит объект x к определенному классу C_i тогда и только тогда, когда выполняется условие:

$$P(C_i|x)>P(C_j|x),$$

где: $P(C_i|x)$ – апостериорная вероятность принадлежности объекта x к классу C_i,

$P(C_j|x)$ – апостериорная вероятность принадлежности объекта x к произвольному классу C_j, отличному от C_i.

Байесовский классификатор относит объект к определенному классу тогда и только тогда, когда апостериорная вероятность принадлежности объекта к этому классу больше апостериорной вероятности принадлежности объекта к любому другому классу.

Классификация деревьями решений – это способ представления совокупности правил в иерархической, последовательностной структуре, пройдя по которому от корня к листу каждому объекту можно сопоставить один единственный узел, соотносящий этот объект и класс, к которому он принадлежит. Алгоритмы построения дерева решений стремятся построить дерево, начиная с корня и далее вниз по узлам дерева до листа, который непосредственно определяет класс, к которому должен принадлежать объект. В частности, так как лист определяет, к какому классу принадлежит объект, зачастую построение делается на основании подсчета наибольшего убывания энтропии обучающей выборки (алгоритм ID3 и его различные модификации) [9]:

$$Gain(x) = \max_T \left(\sum_{r=1}^{k} \frac{Freq(Cr,I)}{I} \log_2 \frac{Freq(Cr,I)}{I} - \sum_{r=1}^{k} \frac{Freq(Cr,T)}{T} \log_2 \frac{Freq(Cr,T)}{T}\right)$$

где: **Freq(Cr, I)** – количество объектов определенного класса **Cr** в исходной выборке **I**,

Freq(Cr,T) – количество объектов определенного класса **Cr** в выборке после разбиения (и образования нового узла) **T**.

Подсчет производится по всем возможным разбиениям и всем классам, а затем выбирается максимальный выигрыш энтропии и на основании этого выбирается наилучшее разбиение на текущем шаге алгоритма.

Кластеризация отличается от классификации тем, что алгоритм не использует обучающую выборку, а делает выводы от принадлежности объектов к определенным классам на основании непосредственно анализируемых данных. Кластеризация позволяет создавать разбиения заданной выборки объектов на непересекающиеся подмножества (кластеры) так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Схожесть и различия определяются при помощи специальной метрики, называемой расстоянием. Расстояние может считаться различными способами, но основные его свойства таковы, что чем более схожи объекты друг с другом, тем меньше должна быть эта метрика. Также эта метрика должна быть строго положительной. Иерархическая кластеризация отличается тем, что строит иерархию кластеров по мере своего выполнения. Непосредственно разбиение исходной выборки на кластеры получают из среза этой иерархии в любой момент работы алгоритма. Обычно критерием останова алгоритма для получения правдоподобной картины распределения объектов по кластерам ставят достижение определенного расстояния между объектами или кластерами, которые на следующем шаге должны быть обработаны[10]:

$$| D(x_j, x_i) | < \varepsilon \text{ , где}$$

- x_i, x_j – n-мерные объекты, которые предложены алгоритмом к объединению в один кластер на данном шаге,
- $D(x_i, x_j)$ – функция расстояния между ними,
- **e** – предварительно выбранное максимальное расстояние, при котором объединения еще будут происходить.

Кластеризация методом k-среднего является быстрым методом кластеризации, не относящимся к иерархическим, хотя схожесть элементов и здесь определяется на основании метрики расстояния.

Основным недостатком данного метода является то, что для его запуска необходимо выдвинуть гипотезу о числе кластеров, содержащихся в исходных данных. На основании этой гипотезы алгоритм образует заданное число кластеров и начнет итеративно перераспределять между ними до тех пор, пока не будет подобрано такое распределение, которое минимизирует расстояние между воображаемыми центрами кластеров (центроидами) и объектами в кластерах. Фактически, данный алгоритм является алгоритмом поиска минимума функции J [11]:

$$J = \sum_{i=1}^{N}\sum_{j=1}^{K} u_{ij} * d(x_i, c_j), \text{ где}$$

- N – число объектов в выборке,
- K – число кластеров,
- x_i – i-ый объект в выборке,
- c_j – центроид кластера Cj,
- $u_{ij} = 1$, если x_i принадлежит кластеру C_j, иначе $u_{ij} = 0$,
- $d(x_i, c_j)$ – метрика расстояния между объектом в выборке и центроидом соответствующего кластера.

После выбора на основе метрики расстояния наиболее близкой к объекту центроиды для всех объектов выборки центроиды перевычисляются на основании метода нахождения центра масс, после чего процесс соотнесения повторяется. Этот метод создает плотные кластеры, однако останов алгоритма на любом этапе до его конца не дает верной картины распределения объектов по кластерам. Критерием окончания здесь чаще всего ставится отсутствие достаточно большого сдвига между центроидами на текущем и предыдущем шаге или достижение определенного достаточно большого числа шагов алгоритма (это ограничение необходимо для того, чтобы избежать зацикливания в том случае, когда какой-либо объект находится на границе двух кластеров):

$$\sum_{i=1}^{K}\left|c_i^{\,t} - c_i^{\,t-1}\right| < \varepsilon \text{ , где}$$

- c_i^t, c_i^{t-1} – центроиды i-ых кластеров на t-ом и t-1-ом шагах соответственно,
- K – общее число кластеров,
- e – предварительно выбранный минимальный сдвиг центроиды, при котором достигнута нужная нам точность распределения и дальнейшее выполнение алгоритма считается бессмысленным.

Представление данных

Традиционное представление данных – это их табличная форма. Однако она ненаглядна, особенно в случае большого объема данных. Поэтому для большей наглядности применяют визуализацию этого объема данных. Существуют различные методы визуализации, однако не все из них подходят для конкретных алгоритмов. Для системы же необходимо иметь достаточно универсальные методы визуализации, так как она применяет различные методы анализа и соответственно для этих методов анализа должны применяться различные методы визуализации. Так, например, для анализа деревьями решений следует использовать визуализатор этого дерева, а для любой другой классификации этот визуализатор не подойдет, скорее всего. Таким образом, возникает необходимость применять определенный визуализатор в зависимости от того, какой именно анализ производился над данными.

Сервер представления результатов анализа реализован на платформе веб-фрэймфорка Django. Выполняемые сервером функции:

- Сервер дает возможность пользователю зарегистрироваться для использования базы игр в системе.
- После регистрации и ее подтверждения через электронную почту пользователь получает доступ к своему кабинету.
- Через кабинет пользователь может скачать необходимое для игры программное обеспечение и начать играть.
- В кабинете отражается накопление данных об играх и, как только данных будет достаточно для анализа, доступна функция анализа.
- При выполнении этой функции можно выбрать алгоритм и его параметры, доступны умолчания для всех параметров.
- Выбранный набор алгоритма и параметров можно сохранить и запустить на исполнение.
- По окончании анализа полученные результаты визуализуются как в виде таблицы, так и в индивидуальных для алгоритмов более наглядных видах.

Клиентская часть системы

Клиент реализует функциональность сбора данных и их отправки на сервер сбора данных.

Сбор данных

Для сбора данных предусмотрено несколько методов:

- Встроенный клиент СИО: при помощи встраивания в непосредственно в обучающую программу. В этом случае сбор данных проходит прозрачно для пользователя. Пользователь использует модифицированную клиентом СИО обучающую программу, и по ходу обучения происходит сбор данных.
- Отдельный клиент СИО: при помощи сбора данных, остающихся по результатам выполнения обучающей программы. В этом случае сбор данных происходит периодически. Для сбора данных клиент должен быть запущен вместе с обучающей программой и настроен на сбор данных именно этой программы. Также должна быть определена структура данных для дальнейшей их интерпретации.

Передача данных

Клиент системы игрового обучения должен производить замеры и отсылать данные о прогрессе учащегося в игре достаточно часто. В связи с этим возникают требования к пропускной способности канала и производительности конечного устройства учащегося. Исходя из этих требований, а также статистической незначимости потери какого-либо отдельного результата учащегося, для передачи данных был выбран протокол UDP. Это позволяет ему гораздо быстрее и эффективнее доставлять данные для приложений, которым требуется большая пропускная способность линий связи, либо требуется малое время доставки данных.

Протокол взаимодействия клиента и сервера

Для реализации модели СИО был разработан основанный на xml протокол обмена информацией о успехах клиента в игре. Протокол формулирует передаваемые сведения, которые затем могут быть использованы для анализа интересов играющих, их действий, их ошибок, для отладки самих программ, для сбора данных об играющих и оценки их успехов в игре.

Основное сообщение протокола имеет следующий вид:

```
<? xml version="1.0" encoding="utf-8" ?>
    <game = gid>
          <data userid = uid, userphash = hash>
                <sensor id=id1> sensor1_data </sensor>
                <sensor id=id2> sensor2_data </sensor>
...
                <sensor id=idn> sensorn_data </sensor>
          </data>
    </game>
```

Протокол используется для обмена между клиентом на машине пользователя и сервером сбора результатов S_g. В основном сообщении указывается уникальный идентификатор игры, получаемый при занесении игры в систему. Сами данные о результатах обучения пользователя систематизированы в соответствии с идентификаторами, которые показывают, что эти данные значат в системе. Область уникальности этих идентификаторов ограничивается каждой конкретной игрой. Для отделения данных разных пользователей друг от друга используется уникальный идентификатор пользователя и хеш-функция его пароля.

Выводы

На основании нескольких фреймворков была реализована сложная система, направленная на внедрение новых методов обучения и их сопровождение, а также улучшение учета, контроля и анализа успеваемости. Преподавателю представлены гибкие интеллектуальные средства анализа различных собираемых данных об успеваемости учеников.

Литература

1. Ю.М. Порховник. Активные методы в дистанционном обучении. // Журнал «Дистанционное образование», №1 от 1997 г.
2. Б. Зельцерман и др. Время Эксперимента. Юбилейный выпуск. – Рига: Педагогический центр "Эксперимент", 2002.
3. Brad Paras, Jim Bizzocchi. Game, Motivation, and Effective Learning: An Integrated Model for Educational Game Design. // Proceedings of DiGRA 2005 Conference: Changing Views – Worlds in Play.
4. Документация по библиотеке lxml. lxml.objectify. // Доступно по http://codespeak.net/lxml/objectify.html .
5. Документация по фреймворку Twisted. Using Threads in Twisted. // Доступно по http://twistedmatrix.com/documents/current/core/howto/threading.html
6. Документация по фреймворку Twisted. Twisted RDBMS support. // Доступно по http://twistedmatrix.com/documents/current/core/howto/rdbms.html .
7. Документация по MySQL 5.0. The MySQL 5.0 Archive Storage Engine // Доступно по http://dev.mysql.com/tech-resources/articles/storage-engine.html .
8. George H. John, Pat Langley: Estimating Continuous Distributions in Bayesian Classifiers. // Eleventh Conference on Uncertainty in Artificial Intelligence, 1995.

9. J. R. Quinlan. Induction of decision trees. // Machine Learning, Volume 1, Issue 1, 1986.
10. Ward, Joe H. Hierarchical Grouping to Optimize an Objective Function.// Journal of the American Statistical Association 58 (301), 1963.
11. MacQueen, J. B. Some Methods for classification and Analysis of Multivariate Observations. // Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967.

ТЕКСТОВАЯ КЛАСТЕРИЗАЦИЯ АЛГОРИТМОМ ROCK

И.И. Савин
Московский государственный институт электроники и математики (технический университет)
E-mail: ivan.savin@2011.auditory.ru

Введение

Большой рост количества информации в электронном виде привел к серьезным проблемам, связанным с ее структуризацией. По данным корпорации EMC [1] на сегодняшний день более 95% цифровой среды состоит из неструктурированной информации. В организациях неструктурированные данные составляют свыше 80% всей информации. Наибольшая часть этих данных содержится в виде текстовых документов.

Структуризация наборов текстовых документов может потребоваться для поиска и сравнения похожих текстов, поиска дубликатов, составления списка документов смежной тематики или просто списка рекомендуемых документов.

Решением этих задач занимается кластерный анализ текстовых документов. Так как документы написаны на человеческом языке, из которого машине трудно выявлять полезную информацию, одной из важных задач кластерного анализа является представление документов в удобном виде для последующей машинной обработки. Текстовые данные практически в любом представлении имеют свою специфику по сравнению с наборами числовых данных, что необходимо учитывать при выборе алгоритма кластеризации.

Целью данной работы является разработка анализатора на основе алгоритма ROCK для последующей кластеризации кафедральной базы знаний, работающей на популярном для использования в образовании движке MediaWiki [2]. Результатом работы является комплекс программного обеспечения, состоящий из программ подготовки текстовых документов и кластеризации, а также расширений для MediaWiki, позволяющих управлять кластеризацией, задавать различные параметры анализа и просматривать результаты анализа.

Подготовка текстовых документов

Теоретическая часть

Текстовый документ с точки зрения машины представляет собой объект, из которого достаточно сложно выявить какие-либо полезные данные кроме размера документа и его метаданных для последующего кластерного анализа. Чтобы это стало возможным, необходимо для начала подготовить текстовый документ, привести его в вид наиболее удобный для анализа машиной.

Существует две наиболее распространенные модели представления текстовых документов: древовидная [3] и векторная модели [4]. Древовидная представляет собой наборы цепочек следующих друг за другом слов. Это позволяет затем машинным анализом выявлять среди нескольких документов похожие цепочки и таким образом выявлять их схожесть. Однако, такой вид представления документа наиболее востребован при проверке на плагиат. В случае же с кластеризацией текстовых документов наилучшим представлением документа является векторная модель.

Векторная модель документа представляет собой таблицу частоты употребления слов в документе. Такое представление позволяет хотя и с некоторой погрешностью, но достаточно быстро определить ключевые слова текста, а значит и его тематику. В особенности такое правило действует для технической литературы, которая и является основным предметом кластеризации данной работы.

В человеческих языках и особенно в русском языке слова имеют достаточно много аффиксов (то есть вспомогательных частей слова, присоединяемых к корню для образования новых однокоренных). Так при составлении векторной модели документа в таблице могут попадаться однокоренные слова с разными окончаниями, которые можно было бы учитывать как одно слово при рассмотрении его тематического значения. Следует заметить, что некоторые аффиксы достаточно сильно меняют смысл слова, поэтому нельзя их рассматривать как незначащие при определении тематического значения (например, «сложный» – «несложный», «использующий» – «используемый»). Значимая часть слова, очищенная от незначащих аффиксов, называется термом. Терм – это основной элемент таблица частоты слов.

Говоря о технической литературе, следует заметить, что она обычно имеет некоторую структуру заголовков и элементов форматирования для удобства использования человеком. Заголовки обычно содержат наиболее значимые слова текстового документа. Также заголовки документа обычно имеют