Для того, чтобы построить дерево товаров, в xml-файле хранится ссылка на первую страницу интернет-магазина, на которой расположен список представленных товаров, в этом же файле указан шаблон ссылки для перехода на следующую страницу, а также XPath-запрос для получения данных по товарам, специфичный для данного сайта. Процесс выглядет следующим образом:

1. По расписанию запускается программа.
2. Программа считывает xml-файл и находит первую ссылку.
3. Если ссылка доступна, то получает данные используя XPath-запрос.
4. Используя шаблон перехода на следующую страницу, осуществляется получение данных со следующей страницы.
5. Данные формируются в коллекцию.
6. Производится сверка коллекции с уже имеющимися данными в системе и, если находится уникальный товар, то он сохраняется в системе.
7. Далее считывается следующая ссылка и повторяются пункты 2-6 до тех пор, пока есть ссылки.

После того как дерево сформировано, необходимо решить вторую проблему, а именно получение детальной информации по каждому из товаров. Для получения данных можно использовать предоставляемый поисковыми системами интерфейс API, однако не все поисковые системы предоставляют данный интерфейс и зачастую имеющийся интерфейс имеет коммерческие ограничения. Таким образом было предусмотрен собственный вариант получение данных из поисковых систем.

Главное особенностью данного вариант является абстрагирование от исходного кода приложения. То есть при изменении формата данных возвращаемых поисковой системой, код приложения не должен переписываться. Наиболее подходящим для решения такой задачи является XSLT. Первоначально необходимо преобразовать полученный HTML в XHTML, далее к XHTML-документу применяется XSL-таблица, выделяющая из XHTML-документа ссылки на найденные ресурсы и их описания в отдельный XML документ. Эта XSL-таблица является частью знаний поискового агента. На основе полученного XML-документа строится объектное дерево, перемещаясь по которому, выстраиваем список найденных ссылок и их описание. Схематично данный процесс представлен на рисунке 2.

Получение огромного количества данных по запросу от различных поисковых систем, их преобразование и анализ требует достаточно производительного сервера. Для уменьшения нагрузки на сервер запланировано обновление дерева товаров и информации по ним в соответствии с заданным графиком – 1 раз в сутки. Такой подход исключит необходимость для каждого отдельного запроса пользователя выполнять полный цикл поиска, что снизит нагрузку на сервер.

Список использованных источников:

1. Michael Wooldridge, *An Introduction to Multiagent Systems* / John Wiley & Sons; 2nd Edition edition, 2009. – 484 p
2. Википедия: Поисковая система. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Поисковая_система. – Дата доступа: 15.03.2015.
3. Портал искусственного интеллекта: Многоагентные системы. [Электронный ресурс]. – Режим доступа: http://www.aiportal.ru/articles/multiagent-systems. – Дата доступа: 15.03.2015.

МОБИЛЬНАЯ АВТОМАТИЗИРОВАННАЯ СИСТЕМА ОБСЛУЖИВАНИЯ КЛИЕНТОВ ЗАВЕДЕНИЙ ПИТАНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Барбасевич А.В., Дубинин А.Ю.

Фролов И.И. – доцент, кандидат технических наук

В наши дни мобильные устройства получают все более широкое применение в повседневной жизни людей. В первую очередь это обусловлено желанием человека постоянно оставаться на связи. Мобильные устройства стали новой ступенью в развитии не только телекоммуникаций между людьми, но и различных других сфер нашей жизни, таких как торговля, маркетинг, экономика, государственное управление и т.д.

В свою очередь одну из лидирующих позиций в мире сейчас занимает компания Apple, которая производит самые популярные смартфоны (iPhone) и планшеты (iPad), предоставляющие неограниченные возможности для пользователей. Данные устройства работают под управлением мобильной операционной системы iOS, которая славится стабильностью, скоростью работы, дизайном и централизованным магазином приложений App Store. С того момента как Apple сделала открытой iOS SDK для разработчиков, App Store начал наполняться различными приложениями: социальными, игровыми, образовательными, медицинскими. За 5 лет существования магазина количество всех приложений достигло цифры в 1 миллион, а число загрузок превысило 60 миллиардов.

Автоматизированная система предназначена для ресторанов, кафе и других заведений питания с целью предоставления возможности посетителю ознакомиться с меню заведения, а также произвести заказ со своего мобильного устройства. Система позволит заведениям увеличить скорость обслуживания клиентов, что должно в свою очередь повысить уровень удовлетворенности посетителей.

Система состоит из серверной части и клиентской, в качестве которой выступает приложение для мобильных телефонов iPhone и планшетов iPad. Основными принципами являются мобильность, удобство, интуитивно-понятный пользовательский интерфейс, сохранность данных. Система позволит заведениям увеличить скорость обслуживания клиентов.

Основные модули автоматизированной системы:

- Серверная часть. Работа с данными полностью возложена на сервер. Модель распределенного приложения выбрана для повышения гибкости системы - добавление новых заведений происходит без необходимости обновления мобильного приложения.
- Мобильное приложение, в котором у удобном для посетителей заведений питания виде представлено меню с возможностью произвести заказ, что-либо доказать в последствии, подозвать официанта, попросить счет.
- База данных содержит информацию о заведениях питания, карты меню зарегистрированных заведений, информацию о клиентах.
- Система аутентификации пользователя. Для подтверждения заказа пользователю нужно пройти аутентификацию при помощи подключения к Wi-Fi сети заведения. Данный модуль используется для предотвращения ложных заказов.
- Веб-клиент предназначен администраторов заведений питания. Позволяет моментально реагировать на произведенные заказы, загруженность кухни, контроль за исполнением заказов и их оплаты.

Рисунок 1. Архитектура системы

Ядро представляет собой контейнер Java сервлетов. Сервлет – Java класс, реализация которого расширяет функциональные возможности функциональные возможности сервера; он взаимодействует с клиентами посредством принципа запрос-ответ. Контейнер представлен в виде сервера Apache Tomcat.

Ядро приложения представляет собой иерархию следующих слоев:

- Стартовый слой, определяющий рабочий процесс начала исполнения программы
- Сетевой слой, обеспечивающий механизм транспортного взаимодействия.
- Слой API – система команд взаимодействия между клиентом и сервером.
- Слой валидации данных.
- Слой сущности данных передаваемых по сети.
- Модель данных обеспечивает взаимодействие сущностей.
- Слой рабочих процессов, включающий классы и алгоритмы специфичные для данного приложения.

Мобильное приложение обеспечивает связь с серверной частью, расположенной у администратора заведения питания, с которой происходит получение актуальных данных вблизи заведений. Такой подход упростит настройку приложения для каждого заведения в отдельности. Данные полученные с серверной части должны быть в формате JSON. Протокол обмена данными реализуется с помощью RESTfull Web-сервиса Jersey.

HTML код веб-приложения совместим со стандартом HTML 5.0. Отображаемый пользователю интерфейс веб-приложения должен соответствовать Impact Internet Application User Interface Standard, Version 2.0. Используется фреймворк Bootstrap 3.x.

Мобильное приложение разрабатывалось для телефонов iPhone и планшетов iPad, и должно было с легкостью использовать все функции данных устройств и при этом бережно расходовать ресурсы (аккумулятор, память). Поэтому в качестве основного языка разработки был выбран Objective-C, а IDE для разработки – Xcode. Приложение содержит базу данных CoreData для хранения полученной информации о заведении питания, а также всей необходимой информации о пользователе.

Клиент-серверное приложение является гибкой системой с возможностью добавления новых заведений питания, без обновления мобильного приложения. Система защищена от ложных заказов. Загрузка приложения не более 5 секунд, в первую загружаются необходимые данные, с последующей дозагрузкой остальных. Данные аутентификации предварительно шифруются для передачи протоколом SSL. Приложение использует метки подтверждения для выполнения следующего запроса пользователя к серверу. Аутентификации происходит следующим образом. Мобильное приложение отправляет запрос на авторизацию, на серверной стороне генерируется token пользователя, который отправляется в ответ на запрос от клиента. В последствии полученный token на стороне клиента добавляется как HTTP заголовок при каждом запросе к серверной части.

Список использованных источников:

1. Zambito, Christine. Process Impact Internet Application User Interface Standard, Version 2.0, www.processimpact.com/corporate/standards/PI_internet_ui_std.doc
2. Patterns of Enterprise Application Architecture / Martin Fowler. - Addison-Wesley Professional, USA, 2002. – 560 с.

АЛГОРИТМЫ АНАЛИЗА ДАННЫХ, ПОЛУЧЕННЫХ ИЗ ОТКРЫТЫХ ИСТОЧНИКОВ СЕТИ ИНТЕРНЕТ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Крошнер А. О.

Самаль Д. А. – доцент, канд. техн. наук

В современном мире в связи с бурным развитием информационных технологий все возрастающее накопление объема информации приводит к необходимости их классификации. При анализе объектов или явлений появляется необходимость учитывать все большее количество параметров. В результате разрабатываются и применяются методы, специализирующиеся на классификации многомерных данных. Во многих случаях возникает необходимость классифицировать данные или найти в них закономерности. Этого можно добиться, используя алгоритмы кластеризации.

Огромной популярностью пользуются социальные сети, которые генерируют подавляющую часть всего интернет-траффика. Примеры таких сетей: Facebook, Twitter, Instagram, SoundCloud, Вконтакте и многие другие. Несмотря на количество пользователей социальных сетей, их создатели заботятся о привлечении все новых и новых пользователей, поскольку это напрямую отражается на денежной конверсии таких ресурсов, значительную часть доходов которых составляет контекстная реклама. В то же самое время, реклама должна быть релевантной интересам пользователя. Поскольку пользователей очень много, сервисам необходимо составлять некоторые модели, описывающие поведение, привычки и интересы пользователей. Составить единую модель, которая будет покрывать всех пользователей, не представляется возможным ввиду их (пользователей) огромного количества и различий. Несмотря на это, можно попытаться составить несколько моделей для различных групп пользователей, которые вполне будут удовлетворять нуждам сервиса. Алгоритмы кластеризация помогают разделить пользователей на некоторые кластеры (группы), на основе которых могут быть составлены модели. Соответственно, сервис будет определять, в какой из групп находится текущий пользователей, и согласно модели данного кластера, предоставлять пользователю информацию.

Абсолютное большинство социальных сетей предоставляют API (программное средство для работы с базой данных сервиса), с помощью которого можно получить данные из сетей. Большинство сервисов предлагает публичные методы для получения информации о пользователях, такой как: базовая информация о пользователе (пол, возраст, образование), информация о связях с другими пользователями системы, новости, которые понравились пользователю и другого рода информация. API сервисов активно используют сторонние компании, которые, применив собственные реализации и комбинации алгоритмов анализа данных пользователей, могут извлечь некоторое полезное «знание», которое может быть использовано по их усмотрению.

Кластеризация представляет из себя задачу по разбиению множества объектов на кластеры. Отличительными признаками кластера являются однородность и изолированность. Однородность означает, что внутри кластера будут находиться схожие между собой объекты. Изолированность же показывает, что объекты одного кластера имеют как можно больше отличий от объектов в другом кластере.

Основной проблемой кластеризации данных является отсутствие универсального решения и, как следствие, необходимость выбора алгоритмов и их параметров экспериментальным путем. Выделение универсального решения задачи кластеризации невозможно по следующим причинам:

- разнородность данных и необходимость их предварительного преобразования;
- выбор параметров объекта, которые будут использоваться в процессе классификации;
- определение меры расстояния (сходства/подобия) между объектами;
- выбор метода кластеризации; заранее неизвестное итоговое количество кластеров.

Алгоритмы кластеризации могут быть классифицированы по результату работы алгоритма. Выделяют