В.В. Ланин, А.Б. Печенежский

Национальный исследовательский университет
«Высшая школа экономики»
(Пермский филиал)

vlanin@live.com

Пермский государственный национальный
исследовательский университет

nextzucker@gmail.com

ИНТЕЛЛЕКТУАЛЬНЫЙ СЕРВИС АНАЛИЗА ИНТЕРНЕТ-КОНТЕНТА НА ОСНОВЕ ОПИСАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

Введение

В настоящее время Интернет является основным и наиболее богатым источником бизнес-информации и знаний. Задача извлечения и агрегации информации, представленной в Интернете, актуальна во многих областях: мониторинг СМИ и социальных сетей, анализ новостных лент. Изначально эта информация ориентирована на человека, следовательно, для её автоматической обработки необходимо использовать интеллектуальные методы. HTML-документы, являющиеся основным носителем информации в Интернет, слабоструктурированы, то есть имеют некоторую внутреннюю структуру, но допускают вхождение в структурный элемент неструктурированного текста. При анализе (структурном и содержательном) слабоструктурированных источников находят широкое применение методы Text Mining, а для Web-пространства можно говорить о Web Mining [7, 8].

Традиционно решение задачи извлечения данных является уникальным для каждого конкретного случая [3, 5]. В настоящей статье

описывается подход к автоматическому сбору структурированной информации из неструктурированных Интернет-документов, позволяющий унифицировать решение этой задачи при реализации проекта создания сервиса анализа Интернет-контента.

Постановка задачи

В рамках проекта необходимо реализовать *интеллектуальный сервис анализа Интернет-контента на основе описания предметной области*.

Сервис на входе должен получать от пользователя список адресов страниц, которые необходимо проанализировать, описание предметной области в виде онтологии, описание анализируемых ресурсов также в виде онтологии. Затем поисковый робот (Web-краулер), являющийся компонентом сервиса, должен выполнять обход каждого ресурса. После этого сервис должен определять информативную часть каждой страницы, извлекать информацию из выделенной информативной части и сохранять в структурированную базу данных (БД). Данный способ работы сервиса обеспечивает инвариантность относительно предметной области и, если необходимо проанализировать новые ресурсы, то никакого изменения программного кода не требуется – меняются описания предметной области и ресурсов. Тем самым сложность поиска остается неизменной.

Кроме того, сервис должен выполнять мониторинг ресурсов, то есть повторять анализ через определенные промежутки времени. При этом необходимо выделять повторно извлеченную информацию, чтобы не сохранять (не дублировать) её в БД.

Детализируем *требования к поисковому роботу*. На вход Web-краулера поступает список начальных URL-адресов; затем для каждого URL-адреса он выполняет загрузку страницы и передает загруженную страницу анализатору страниц. От анализатора страниц поисковый робот получает URL-адреса, извлеченные с загруженных страниц. Добавление новых URL-адресов завершается после достижения определенной глубины обхода.

Кроме того, Web-краулер должен:

- следить за повторяющимися страницами;
- учитывать возможные ошибки при загрузке страниц;
- как можно раньше выявлять страницы, которые не подходят для анализа;
- обрабатывать код HTML, в котором допущены ошибки;
- поддерживать динамическую загрузку страниц;

- переводить относительные URL в абсолютные;
- использовать канонический URL, чтобы избегать дублирования страниц;
- использовать эвристические правила для отбора страниц (ограничение длины адреса, наличие или отсутствие определенных слов и пр.);
- в целях повышения эффективности хранить несколько страниц в одном большом файле, используя XML-разметку;
- соблюдать «сетевой этикет» (идентифицировать себя, не отправлять слишком много запросов на один сервер, учитывать robots.txt).

Для каждой загруженной страницы анализатор страниц должен выполнять перечисленные ниже действия:

1. Построение DOM-дерева.
2. Извлечение URL-адресов.
3. Нахождение информативной части страницы.
4. Анализ информативной части.

При анализе информативной части анализатором страниц должен поддерживаться следующий *алгоритм работы*:

1. Разбиение текста на абзацы, предложения, слова.
2. Лемматизация слов и нормализация других объектов (числа, даты). При этом должна быть обеспечена возможность изменения словаря лемматизации [2].
3. Нахождение объектов поиска и определение их типов (например: ФИО, компания). На данном этапе должна быть возможность использовать справочные списки, содержащие имена собственные. Если слово находится в каком-либо списке, то оно принадлежит категории объектов, которую описывает этот список.
4. Установление связей между объектами. Для данной задачи предлагается использовать правила анализа, описанные в виде регулярных выражений, и строить деревья разбора.

Данные шаги должны выполняться на основе *онтологии предметной области* (формализованного описания предметной области). Для уменьшения пространства поиска при анализе страницы используется *онтология ресурсов* (описание ресурсов).

Таким образом, разработка процедуры анализа каждой страницы заменяется созданием *универсального поискового робота*, *анализатора страницы* и созданием *онтологий* для извлечения информации из текстов в новой предметной области [5, 6, 9]. Следовательно, единственным трудозатратным процессом является создание прикладной онтологии, требующее наличия эксперта.

Для реализации сервиса к создаваемым онтологиям предъявляются следующие требования:

1. Простота: онтология должна включать минимально возможное количество связей и понятий.
2. Возможность динамического изменения онтологии в процессе эксплуатации сервиса.

Для увеличения скорости анализа Интернет-ресурсов следует использовать описание Интернет-ресурсов, содержащее для каждого ресурса регулярное выражение для фильтра найденных на странице URL-адресов и XPath-запрос, позволяющий выделить в коде HTML место, где находится интересующий контент, и другую полезную информацию. Данное описание легко формируется при создании сервиса и при этом заметно увеличивает эффективность его работы, так как позволяет «игнорировать» ненужные URL-адреса и извлекать только полезную для пользователя информацию.

Обзор синтаксических анализаторов HTML

В процессе реализации программы потребуется выполнять синтаксический анализ HTML-страницы [10] для поиска нужной информации и ссылок на другие страницы. Для решения этой задачи необходимо выбрать библиотеку синтаксического разбора платформы .Net с открытым исходным кодом. Указанным требованиям удовлетворяют следующие библиотеки: HTML Agility Pack, SGMLReader.

HTML Agility Pack является свободной .Net библиотекой, позволяющей анализировать HTML-файлы. Доступны следующие возможности: Linq to Objects, XPath, XSLT. Методы библиотеки имеют названия, соответствующие интерфейсам DOM (Document Object Model Core), и позволяют получить доступ к содержимому HTML-страницы, представленному в виде дерева объектных узлов.

SGMLReader – C#-библиотека для преобразования кода HTML в XML. Способ работы этого анализатора аналогичен HTMLAgilityPack. HTML-документ преобразуется в DOM-структуру и позволяет работать с HTML-документом, как с XML. Доступны возможности Linq и XPath. SGMLReader, в отличие от HTMLAgilityPack, преобразует документ в объект типа XmlDocument, а его конкурент – в собственный тип HTMLDocument, являющийся оберткой XmlDocument. Благодаря чему программистам, имеющим опыт работы с XML-документами, привычней использовать SGMLReader. Следующим отличием является отсутствие встроенного загрузчика страниц.

Majestic-12 – это синтаксический анализатор .Net HTML с исходным кодом. Он создан для использования в распределенной поисковой

системе, в которой ежедневно требуется быстро обрабатывать терабайты HTML-документов, поэтому основное внимание при его разработке было уделено высокой производительности. Библиотека также отлично подходит для разбора XML, но не имеет Linq- и API-парсера.

Альтернативным способом анализа является перевод HTML-документа в xHTML-документ для возможности использования средства работы с XML-документами. Для получения xHTML-документа подойдет библиотека Tidy.Net, которая исправляет неверный код HTML и улучшает разметку.

Результаты проведённого аналитического обзора представлены в табл. 1.

Таблица 1. Сравнение синтаксических анализаторов

Требование	HTMLAgilityPack	SGMLReader	Majestic-12
LINQ	+	+	–
Производительность	Медленная обработка тегов	Медленная обработка тегов	Заточена на производительность
Встроенный загрузчик страниц	+	–	–
Документация и рабочие примеры в Интернете	Качественная документация и множество примеров	Плохая документация и сложность с нахождением примеров	Плохая документация и отсутствие примеров
Исправление ошибочной HTML-разметки	+	+	–

Обзор систем анализа Интернет-контента

Задача извлечения и агрегации информации актуальна во многих областях: например, мониторинг средств массовой информации, социальных сетей, новостных лент. Сервисы, решающие данную задачу, можно разделить по двум признакам: способ анализа и область поиска (табл. 2).

Одним из популярных сервисов является проект 80legs, который предоставляет пользователю доступ к легко настраиваемому Web-краулеру для сбора данных из Интернета. Пользователь вводит список URL-адресов, с которых необходимо начинать обход, указывает какую

информацию необходимо сохранять с помощью ключевых слов, регулярных выражений или собственного java-алгоритма. Возможности сервиса не позволяют учитывать семантику. Кроме того, 80legs даёт доступ к обновляемым пакетам с записями о различных исследованиях, которые пополняются на 10 миллионов записей каждый месяц. 80legs используется сотнями компаний для поиска рекламных каналов, для улучшения почтового спам-фильтра и пр.

В отдельную группу можно выделить системы мониторинга социальных медиа; например, такие проекты как youscan, iqbuzz, wobot. Они используют различные статистические алгоритмы для анализа Интернет-контента и представляют извлеченные данные в структурированном виде. Отметим, что данные проекты ограничивают область поиска.

Системы, которые используют методы анализа, учитывающие семантику, являются, как правило, исследовательскими, например Ontos [1], PULS. При этом в данных системах область поиска часто ограничена.

Таблица 2. Сравнение синтаксических анализаторов

Название	Способ анализа	Область поиска
80legs.com	Ключевые слова. Регулярные выражения	Без ограничения
YouScan.ru **Iqbuzz.ru** **Wobot.ru**	Статистические методы	Социальные сети. СМИ
ontos.com **PULS**	Семантические методы	Новостные ленты

Архитектура системы и подходы к реализации

Интеллектуальный сервис анализа Интернет-контента должен извлекать информацию из источников, расположенных в сети Интернет. При этом поиск и анализ HTML-документов и Web-ресурсов должен происходить автоматически. Для решения данной задачи разработана архитектура системы, представленная на рис. 1.

Сервис состоит из двух основных компонентов: *поискового робота* и *анализатора страниц*. Кроме того, сервис включает *механизм журнализации* и *менеджер настроек*. Компонент журнализации используется для записи информации о работе системы. Логирование используется как метод отладки программного обеспечения, а также способ мониторинга работы системы. Менеджер настроек позволяет

производить динамическую настройку системы. Пользователь взаимодействует с базой данных адресов, описанием предметной области и описанием Интернет-ресурса.

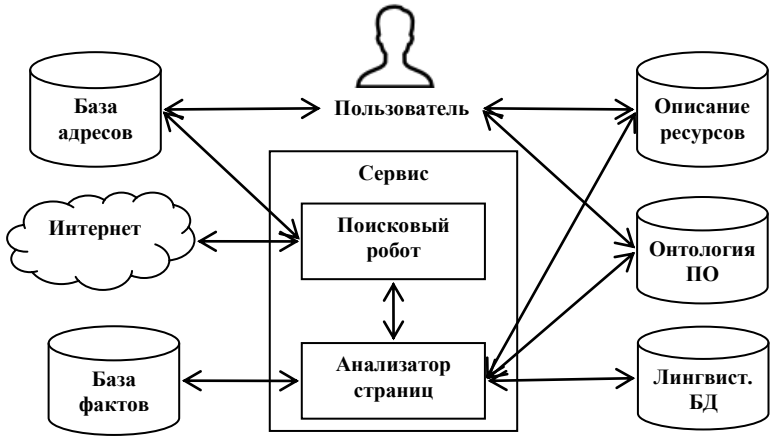

Рис. 1. Архитектура системы

База данных (БД) адресов хранится в формате XML, что позволяет хранить для каждого адреса дополнительную информацию, например глубину поиска или необходимость динамической подгрузки страницы. Описание предметной области и описание Интернет-ресурсов хранятся в формате OWL, а база данных фактов – в реляционной базе данных.

Благодаря использованию описания предметной области достигается инвариантность сервиса относительно предметной области. А описание Интернет-ресурсов позволяет увеличить эффективность анализа за счёт учёта знаний о ресурсах при обходе и извлечении информации.

Рассмотрим архитектуру поискового робота (рис. 2), который состоит из менеджера адресов и загрузчика. Алгоритм работы робота включает несколько этапов. Пользователь добавляет адреса ресурсов для анализа в базу данных адресов, менеджер адресов перебирает эти адреса. Менеджер адресов получает адрес из БД адресов и передает его загрузчику, который загружает страницу по данному адресу из сети Интернет, проверив возможные ошибки, и передаёт её анализатору страниц. Анализатор страниц кроме извлечения информации находит ссылки в HTML-документе и передает их менеджеру адресов, который

сохраняет новые адреса в БД адресов.

База данных адресов хранит информацию о посещенных страницах и страницах, которые ожидают анализа. Благодаря чему исключается повторный анализ страниц, расположенных по одинаковым адресам в текущую сессию загрузки.

Рис. 2. Архитектура поискового робота

Рассмотрим архитектуру анализатора страниц (рис. 3). Основными компонентами здесь являются анализатор HTML-документа, анализатор текста, менеджер описаний.

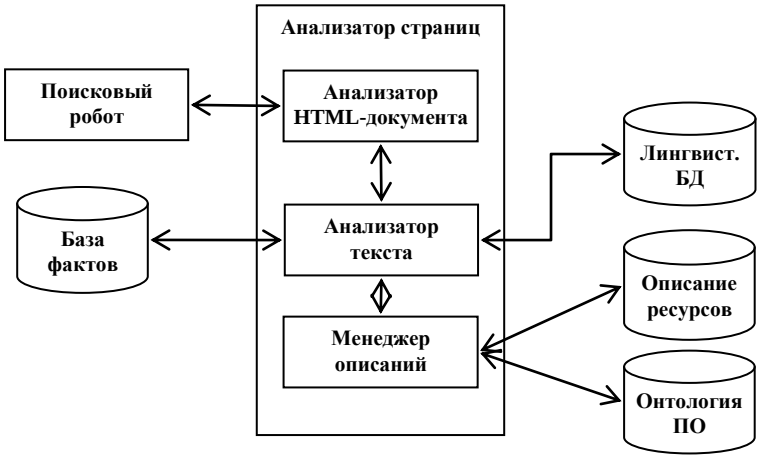

Рис. 3. Архитектура анализатора страниц

Поисковый робот передает загруженную страницу анализатору HTML-документа, который строит DOM-дерево HTML-документа, извлекает информативную часть страницы и передает её анализатору текстовой информации.

Анализатор текста, используя базу лемматизатора и описание предметной области, через менеджер описаний извлекает факты предметной области и сохраняет их в БД фактов. При этом для получения знаний о текущем Интернет-ресурсе анализатор страниц использует его описание. В качестве редактора описаний предполагается использовать Protégé.

База данных фактов предназначена для хранения извлеченной информации. При сохранении нового факта необходима проверка на наличие аналогичного объекта в базе фактов. Кроме того, в базе данных хранится информация о пройденных страницах. Описание предметной области создаётся заранее и содержит термины предметной области, связи между ними.

Заключение

Прораммы представленной системы разрабатываются в Microsoft Visual Studio, а в качестве СУБД используется Microsoft SQL Server. Для реализации компонента разбора HTML выбрана библиотека Agility Pack. Для лемматизации используется библиотека морфологического анализа на основе словаря А.А. Зализняка (MCR.DLL). А для работы с описанием предметной области, которое представляется в виде онтологии и хранится в формате OWL, используется библиотека OWLdotnetapi.

Преимущества разрабатываемой системы обеспечиваются применением описаний предметной области и ресурсов при анализе Интернет-контента. Это позволяет извлекать более качественную информацию и даёт возможность анализировать данные без ограничения области поиска. Кроме того, для расширения множества анализируемых источников не требуется вносить изменения в код программ, следовательно, сложность разработки не изменяется.

Библиографический список

1. *Хорошевский В.Ф.* OntosMiner: семейство систем извлечения информации из мультиязычных коллекций документов // Труды IX Национальной конференции по искусственному интеллекту с международным участием КИИ-2004. В 3 т. М.: Физматлит, 2004. Т. 2. С. 573-581.

2. *Лукашевич Н.В, Добров Б.В.* Отношения в онтологиях для решения задач информационного поиска в больших разнородных текстовых коллекциях // Труды IX Национальной конференции по искусственному интеллекту с международным участием КИИ-2004. В 3 т. М.: Физматлит, 2004. М.: Физматлит, 2004. Т. 2. С. 544-551.
3. *Chugunov A., Lanin V.* Intelligent search based on ontological resources and graph models // Proceedings of the 7th Spring/Summer Young Researchers' Colloquium on Software Engineering, SYRCoSE 2013 / Ed. by: A. Petrenko; A. Terekhov; A. Kamkin. Kazan, 2013. P. 133-135.
4. *Lanin V., Nesterov R., Osotova T.* Intelligent Service for Aggregation of Real Estate Market Offers // Proceedings of the 7th Spring/Summer Young Researchers' Colloquium on Software Engineering, SYRCoSE 2013 / Ed. By: A. Petrenko; A. Terekhov; A. Kamkin. Kazan, 2013. P. 136-138.
5. *Buitelaar P., Cimiano P., Frank A., Hartung M., Racioppa S.* Ontology-based information extraction and integration from heterogeneous data sources. Int. J. Hum.-Comput. Stud. 66, 11. 2008. P.759-788.
6. *Grigalis T.* Towards web-scale structured web data extraction // Proceedings of the sixth ACM international conference on Web search and data mining (WSDM '13). ACM, New York, NY, USA, 2013. P. 753-758.
7. *Stumme G., Hotho A., Berendt B.* Semantic Web Mining // Web Semant. № 4, V.2. 2006. P. 124-143.
8. *Liu B.* Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data. Chicago:Springer, 2007. 532 p.
9. *Weal M.J., Kim S., Lewis P.H., Millard D.E., Sinclair P.A.S., De Roure D.C., Nigel R.* Ontologies as facilitators for repurposing web documents / Shadbolt. Southampton, 2007. P. 537-562.
10. *Xu Z., Yan D.* Designing and Implementing of the Webpage Information Extracting Model Based on Tags // Proceedings of the 2011 International Conference on Intelligence Science and Information Engineering (ISIE '11). IEEE Computer Society, Washington, DC, USA, 2011. P. 273-275.