МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Информационные системы и технологии корпоративного управления
(направленность (профиль)

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему **«Исследование и применение перспективных методов мониторинга информационных ресурсов»**

Студент Т.Д. Мирзонаботов _____
(И.О. Фамилия) (личная подпись)

Научный руководитель О.В. Аникина _____
(И.О. Фамилия) (личная подпись)

Руководитель программы д.т.н., доцент С.В. Мкртычев _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2019

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Во многих сферах деятельности продолжается процесс активного освоения новых информационных технологий. Современные информационные технологии, системы и средства помимо обычных вычислений все больше используются для систематизации, хранения обработки и анализа информации, а также обеспечения оперативного доступа к ней. При этом, кроме представления информации в традиционной числовой и текстовой форме, все больше применяются такие формы представления, как графики, карты, рисунки, аудио- и видеофайлы.

Во время исследования информационных ресурсов возникают проблемы поиска необходимых сведений об объекте в большом объеме неструктурированной или слабоструктурированной информации, хранящейся в компьютерах и компьютерных сетях, в том числе в глобальной сети Интернет.

Особенности функционирования сети Интернет определяют необходимость проведения дополнительного исследования процессов мониторинга и анализа распределенной информации в компьютерных сетях, а также создания программных продуктов, обеспечивающих доступ к неструктурированной или слабоструктурированной информации, осуществляющих сбор и анализ найденной информации.

Актуальность исследования обусловлена необходимостью разработки современных средств автоматизации мониторинга и анализа распределенной информации в сети Интернет, способствующих повышению эффективности принятия управленческих решений специалистами в предметной области.

Целью исследования является исследование перспективных методов и моделей информационного поиска и разработка программного приложения, осуществляющего поиск и анализ информации в сети Интернет для повышения эффективности управленческих решений.

Объект исследования – процессы информационного поиска и обработки информации. **Предметом** исследования являются методы сбора, анализа и оценки информации в среде Интернет.

Гипотеза исследования состоит в предположении, что разработанное на основе предложенного метода программное приложение, автоматизирующее процесс сбора и анализа больших объемов информации, способствует повышению качества принятия управленческих решений.

Задачи исследования:

1. Исследовать закономерности функционирования процессов мониторинга и анализа информации в сети Интернет, а также подходы к представлению результатов.

2. Проанализировать проблемы разработки и применения методов мониторинга информации в сети Интернет.

3. Разработать модель информационного поиска и анализа экологических данных.

4. Разработать программное приложение, автоматизирующее процесс сбора и анализа информации в сети Интернет.

5. Экспериментально проверить разработанное программное приложение и определить эффективность его использования для принятия управленческих решений.

Недостаточная проработка существующих методов и подходов к организации информационного мониторинга обусловили необходимость проведения дополнительного исследования в данном направлении.

Методы исследования: методы поиска и анализа неструктурированной информации в сети Интернет.

Публикации. Результаты исследования были представлены на VIII Всероссийской научно-практической конференции «Современная техника и технологии: проблемы, состояние и перспективы» (Алтайский край, г.Рубцовск, 22-23 ноября 2018 г.), на V Международной научно-практической конференции (школы-семинара) молодых ученых «Прикладная математика и информатика: современные исследования в области естественных и технических наук» (Тольятти, 22-24 апреля 2019 года), на научно-практической

конференции «Студенческие Дни науки в ТГУ – 2019» (Тольятти, 1-30 апреля 2019 года).

Основные этапы исследования: исследование проводилось в период 2017-2019 гг.

На **первом** этапе исследования (2017 г.) - была определена актуальность исследования, выполнена проработка литературы по выбранной теме, формулировалась гипотеза, ставились цели, задачи, определялись предмет, объект исследования, уточнены методы исследования, изучались существующие методы поиска и анализа неструктурированной информации в глобальной сети.

На **втором** этапе исследования (2017 – 2018 гг.) – выполнялся анализ существующих автоматизированных систем мониторинга ИТ-сред, изучены методы поиска и анализа информации на отдельных сайтах.

На **третьем** этапе (2018 – 2019 гг.) – выполнялась разработка сценария для автоматизированного поиска и анализа неструктурированной информации в сети Интернет.

Научная новизна исследования заключается в разработке нового алгоритма поиска и разработке на основе него сценария для автоматического поиска неструктурированной информации в сети Интернет в среде R с использованием пакета расширения RCrawler, разработаны рекомендации по локализации пакета.

Практическая значимость заключается в том, что спроектирован и реализован сценарий для автоматического поиска необходимой информации, способствующий увеличению эффективности работы лица принимающего решения.

Соответствие содержания магистерской диссертации профессиональным компетенциям по видам профессиональной деятельности выпускника:

научно-исследовательская деятельность:

способностью ставить и решать прикладные задачи в условиях неопределенности и определять методы и средства их эффективного решения (ПК-3);

аналитическая деятельность:

способностью анализировать и оптимизировать прикладные и информационные процессы (ПК-9);

способностью проводить маркетинговый анализ ИКТ и вычислительного оборудования для рационального выбора инструментария автоматизации и информатизации прикладных задач (ПК-10);

проектная деятельность:

способностью применять современные методы и инструментальные средства прикладной информатики для автоматизации и информатизации решения прикладных задач различных классов и создания ИС (ПК-11);

производственно-технологическая деятельность:

способностью использовать информационные сервисы для автоматизации прикладных и информационных процессов (ПК-23).

На защиту выносится:

1. Алгоритм автоматического поиска в среде R с использованием пакета расширения RCrawler.

2. Разработанный сценарий автоматического поиска информации в сети Интернет.

3. Алгоритм решения проблемы поиска на русскоязычных сайтах.

4. Анализ эффективности автоматизированного поиска информации.

В структуру работы входят: введение, 3 главы, заключение, список используемой литературы и приложения.

Работа изложена на 77 страницах и включает 17 рисунков, 5 таблиц, 30 источников.

ГЛАВА 1 АНАЛИЗ МЕТОДОВ И МОДЕЛЕЙ МОНИТОРИНГА ИНФОРМАЦИОННЫХ РЕСУРСОВ ПРЕДПРИЯТИЯ

1.1 Мониторинг и анализ информации в сети Интернет

Для принятия управленческих решений информационные ресурсы имеют первостепенное значение, наряду с материальными, трудовыми, сырьевыми, финансовыми. Процесс принятия управленческого решения в подавляющем большинстве случаев опирается на имеющуюся в распоряжении менеджера альтернативную информацию. В данном случае важным аспектом является использование именно той информации, которая уменьшит неопределенность непрерывно развивающихся событий и поможет принять наиболее оптимальное и эффективное управленческое решение.

Управленческая информация имеет целый ряд особенностей. К ним относятся:

- обработка больших объемов информации зачастую ограничена установленными сроками;
- исходная информация должна быть подвергнута неоднократной обработке с учетом требований потребителей;
- исходные данные и результаты произведенных расчетов зачастую хранятся достаточно длительное время.

Широкий круг таких проблем, как формализация различных видов данных, способы извлечения знаний из данных, методы представления предметной области, охватывает метод интеллектуальной обработки текстовой информации. Данному направлению посвящены работы Э. Баха, Т. А. Гавриловой, Е.Е. Раковской [5, 6, 11].

Кузьмин М.Н. в своей статье подчеркивает, что в современном развивающемся обществе важнейшим стратегическим ресурсом управления является информация, которой придается огромное значение [8]. Автор отмечает, что при выборе перспективных направлений развития организации в любой отрасли хозяйствования требуется внедрение и использование

эффективных методов и инструментов оценки и диагностики состояний объекта исследования. В настоящее время к числу таких инструментов относится мониторинг [8].

Понятие «мониторинг» (от англ. monitoring в переводе – отслеживание) является общеизвестным термином как в науке, так и в других областях деятельности. Суть этого понятия определяется в постоянном наблюдении за каким-либо процессом с целью выявления его соответствия желаемому результату. Другими словами, если ведется непрерывный процесс диагностики ситуации с определенной заданной периодичностью и с использованием некоторой базовой системы индикаторов, тогда о таком процессе говорят, как о мониторинге.

В настоящее время мониторинг применяется для отслеживания, анализа и прогнозирования текущего состояния экономических, политически, социальных, экологических и других объектов, а также тенденций их развития.

Мониторинговые исследования позволяют оперативно оценивать направленность и характер изменений, происходящих в окружающей среде, принимать адекватные и эффективные управленческие решения, прогнозировать и моделировать развитие объекта исследования.

К общим правилам проведения мониторинга можно отнести следующие [14]:

- система показателей, разработанная для мониторинга, должна адекватно отражать основные характеристики объекта и создавать комплексное представление о его функционировании;
- должен использоваться универсальный блок индикаторов, позволяющий проводить сравнительный анализ и построение динамических рядов.

Аверченков В.И. и Рощин С.М. в своей монографии отмечают, что задача мониторинга информационных ресурсов является наиболее важной при формировании знаний, определяемых как ценный и наиболее важный ресурс

любой организации [1]. Авторы приводят несколько примеров типовых задач мониторинга, которые наиболее часто встречаются в организациях:

- сбор информации для принятия эффективных управленческих решений, направленных на улучшение качества продуктов и услуг, поиск оптимальных путей достижения поставленной цели;
- сбор информации о деятельности конкурентов, в том числе поиск новых конкурентов в сфере деятельности организации;
- исследование правовой сферы, в том числе мониторинг законодательства в области деятельности организации;
- маркетинговые исследования, включающие сбор информации о покупательских потребностях, изменении цен на продукты и услуги, спроса и предложения;
- проведение исследований в области технологий с целью генерации новых идей и решений, а также выявления новых перспективных направлений деятельности организации;
- осуществление образовательной деятельности. К такой деятельности относится мониторинг современных образовательных технологий, обучающих методических и программных средств.

Перечисленные задачи, как и многие другие, могут быть решены с использованием сети Интернет, содержащей огромное множество документов из самых различных областей знаний.

В современном мире Интернет становится важнейшим источником информации при решении задач мониторинга в практически любой сфере деятельности, наряду с печатными и электронными средствами массовой информации.

Для эффективного управления организацией любого уровня необходимо своевременное получение достаточных, достоверных и оперативных знаний.

Главной целью информационных ресурсов, применяемых в организациях, является предоставление внешнеэкономической и внутрифинансовой деятельности. Согласно источнику полученные данные являются внешними

согласно взаимоотношению к нему, или внутренними, то есть корпоративными. Схематично классификация информационных ресурсов организации представлена на рисунке 1.1.

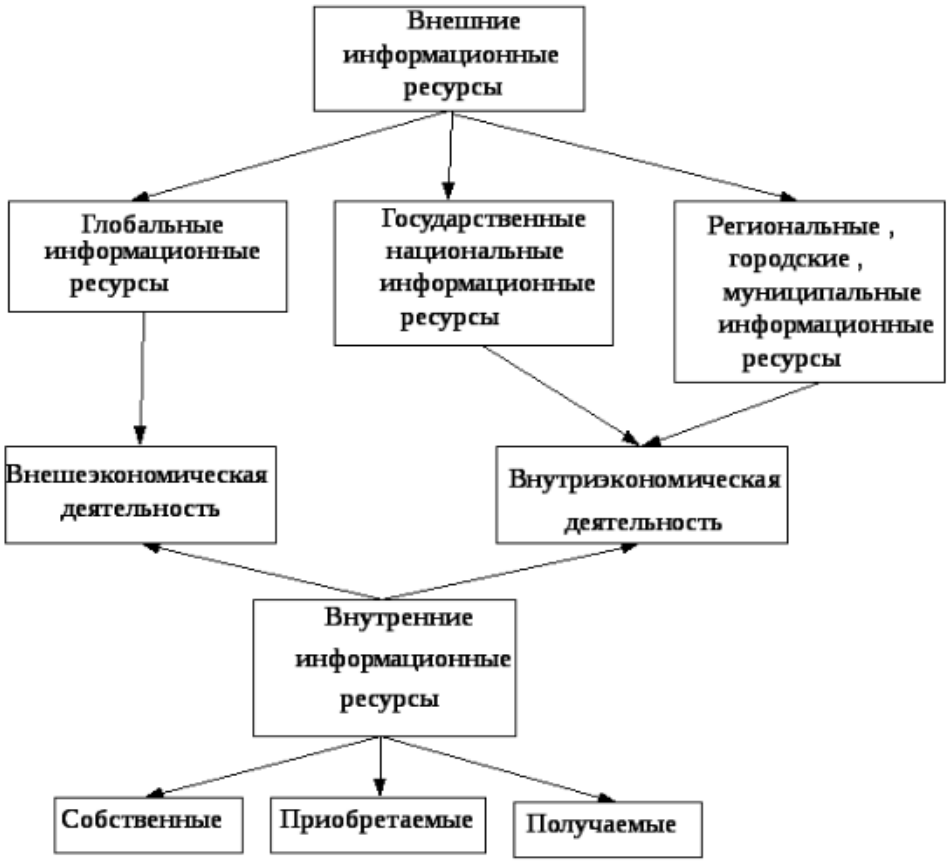


Рисунок 1.1 - Классификация информационных ресурсов предприятия по источнику возникновения

В результате мониторинга и анализа информации организации предоставляется ряд преимуществ:

- высокая оперативность и своевременность принятия управленческих решений;
- повышение конкурентоспособности организации и снижение рисков;
- использование преимуществ принятия эффективных управленческих решений, основанных на знаниях.

Технологии информационного поиска решают задачи уменьшения информационной неопределенности, в связи с чем зачастую интегрированы в

систему информационных технологий, связанную с обработкой информации и управлением [16].

Таким образом, актуальным направлением исследования является решение задач мониторинга, связанных с поиском и обработкой необходимой информации в сети Интернет.

1.2 Обзор методов и моделей информационного поиска

Применение тех или иных методов мониторинга в основном зависит от точности полученных необходимых результатов при проведении мониторинга в соответствии с тем или иным компонентом, явлением, процессом, от среды, в которой проходят исследования, доступных финансовых, материальных и иных средств.

Авторы монографии [1] выделяют ряд подходов мониторинга информации в глобальной сети:

- сбор и анализ релевантных документов, клторые полностью соответствуют предметной области, путем самостоятельного исследования всего имеющегося информационного массива в глобальной сети;
- сбор и анализ необходимых документов с использованием знаний об информационных ресурсах Интернета, с использованием средств поиска, например каталогов;
- отбор необходимых документов с помощью составления запросов существующим средством информационного поиска с использование ключевых слов предметной области и последующим анализом полученной информации.

При формировании системы знаний о предметной области применяется совокупность методов и средств исследования сложных систем, объектов и процессов, с учетом взаимосвязей и взаимодействий между элементами системы (рисунок 1.2) [1, 2].

Рисунок 1.2 – Методы системного анализа

Суть информационного поиска заключается в том, что необходимо найти документы, изображения, содержащиеся в хранилище неструктурированной или слабоструктурированной информации. В данном случае существенное значение имеет степень соответствия запроса и результата поиска – релевантность.

В настоящее время поиск необходимой и важной информации на веб-ресурсах осуществляют с использованием поисковых систем, представляющих собой совокупность программных и лингвистических средств. Данные системы реализуют основные этапы информационного поиска, такие как:

- предварительный анализ информации;
- формирование поискового запроса,

- непосредственно поиск и предоставление результатов поиска.

После завершения системного анализа информации ее необходимо предоставить пользователю в упорядоченном (ранжированном) виде. Главной проблемой представления информации является невозможность проводить быстрый анализ большого объема данных.

Text mining – это перспективная область исследований, которая направленная на решение проблем информационного кризиса путем использования технологий из data mining, машинного обучения (machine learning), обработки естественных языков (natural language processing (NLP)), поиска информации (information retrieval (IR)) и управления знаниями (nowledge management) [26].

Технология text mining относится информационному поиску, а так же является методом анализа неструктурированного текста, т.е. набора документов, не предполагающего каких-либо ограничений на его структуру [13]. Например, это могут быть web-страницы, электронная почта, нормативные документы и т. п. Такие документы могут иметь сложную структуру и включать в себя как текст, так и графическую информацию.

Процесс анализа текстовых документов представлен на рисунке 1.3 и представляет собой последовательность из пяти шагов [10].

1. На первом шаге осуществляется поиск информации. На начальном этапе необходимо определить, какие документы должны быть подвержены анализу. При достаточно большом количестве документов необходимо использовать способы автоматизированного отбора с учетом заданных критериев.

Рисунок 1.3 – Этапы технологии Text Mining

2. Предварительная обработка документов. Данный этап предполагает выполнение ряда несложных, но необходимых преобразований с документами с целью представления их в нужном виде для работы методами Text Mining. Целью таких преобразований является удаление ненужных слов и приведение текста к более строгому виду.

3. Под извлечением информации из выбранных документов предполагают выделение ключевых понятий, над которыми в дальнейшем будет производиться анализ.

4. Применение методов Text Mining является основным шагом в процессе анализа текстов. На этом шаге извлекаются шаблоны и отношения, имеющиеся в текстах.

5. На последнем шаге в процессе обнаружения знаний предполагается интерпретация полученных результатов, что, как правило, заключается или в представлении результатов на естественном языке, или в их визуализации в наглядном графическом виде.

Text mining включает предобработку коллекции документов (категоризация (или каталогизация) текста, извлечение информации и извлечение терминов), сохранение промежуточных представлений, технологии или техники анализа этих промежуточных представлений (такие как анализ распределений, кластеризация, анализ трендов, правила ассоциации) и визуализация результатов [26].

Слабоструктурированные и полуструктурированные документы

Несмотря на несколько вводящий в заблуждение термин «неструктурированные данные» текстовый документ с различных точек зрения можно рассматривать как структурированный объект. С лингвистической точки зрения даже простой или обычный документ имеет большую (значительную) семантическую и синтаксическую структуру, хотя эта структура неявно спрятана в текстовом контенте.

Дополнительно типографские элементы, такие как, знаки пунктуации, заглавные буквы, числа, специальные символы, дополненные артефактами

раскладки (пробелы, возврат каретки, подчеркивание, звездочки, таблицы, столбцы и т.д.) могут служить в качестве мягкого языка разметки, помогающий идентифицировать важные компоненты документа, такие как абзацы, заголовки, даты публикации, фамилии авторов, табличные данные, сноски.

На другом конце «неструктурированных» данных находятся HTML- и XML-документы, которые содержат формализованные теги разметки. Документы, которые используют сравнительно мало способов типографской раскладки или тегов разметки, наподобие научных статей, бизнес-отчетов, новостей, иногда называются свободноформатированными или слабоструктурированными документами. С другой стороны, документы со значительным числом элементов форматирования, такие как e-mail, html-страницы, pdf-файлы, тексты, обработанные текстовыми процессорами, считаются полуструктурированными документами.

Даже самые простые документы содержат большое количество слов, фраз, предложений, типографских элементов, артефактов раскладки, не говоря уж о разных смыслах разных элементов (одно и то же слово может означать разный смысл) и представляют собой серьезную проблему для большинства систем текст-манинга для идентификации упрощенного подмножества свойств документа, которые могут представлять отдельный документ как целое. Мы называем такое множество свойств моделью представления документа.

В частности, одним из шагов предобработки является отбрасывание из документа стоп-слов, наподобие союзов, предлогов, связок, глаголов типа «есть», которые не несут смысла. Эти слова в модель представления не входят.

Огромное число характеристик из необходимых для представления документов коллекции влияет почти на все аспекты разрабатываемой системы дата-манинга, на его концепцию, дизайн, производительность.

Структурированное представление документов на естественных языках требует значительно большее число представительных характеристик, намного больше, чем в реляционных или иерархических БД. Например, сравнительно маленькая коллекция из 15 тысяч документов - лента новостей Reuters –

требует идентификации более 25 тысяч слов, в базах данных это намного-намного меньше.

Проблема большой размерности характеристик коллекции документов в текст-манинге является критической для всех текст-манинг систем. Другой особенностью документов на естественных языках является разреженность характеристик, т.е. только маленький процент возможных характеристик всей коллекции встречается в отдельном документе. Поэтому, когда документ представляется как бинарный вектор характеристик, почти все значения этого вектора равны нулю. Это же относится к размерности картежей, т.е. отдельные характеристики часто встречаются только в небольшом числе документов.

Обычно используемые характеристики документа: буквы, слова, термины, концепции. Модель представления с одной стороны должна более полно отражать характеристики документов коллекции, с другой стороны, должна быть вычислительно эффективной и практической для поиска шаблонов. Т.е. система текст-манинг должна генерировать как можно меньшее по размеру и как можно более семантически богатое представительные наборы слов.

Несмотря на то, что существует большое число представительных характеристик документа, основными являются следующие 4 характеристики:

1. Символы. Это индивидуальные компоненты на уровне букв, чисел, специальных символов пробелов, которые являются строительными блоками высокоуровневых семантических характеристик, таких как слова, термины, концепции. Представление, базирующееся на символах, может содержать информацию о положении этих символов.

2. Слова. Выбираются из исходного документа и могут рассматриваться как базовый уровень семантики. Характеристики уровня слова иногда рассматриваются как предметная область документа. Фразы. Многословные выражения не водят в этот уровень. Большинство представлений уровня «слово» используют некоторую оптимизацию и поэтому состоят из

подмножества отфильтрованных слов, таких как стоп-слова, символы, числа, не имеющие смысла.

3. Термины. Это отдельные слова или многословные фразы, полученные с помощью методологии извлечения терминов.

4. Концепция. Это свойство, получаемое из документа путем использования методологии ручной, статистической, основанной на правилах или гибридной каталогизации. Характеристики этого уровня могут извлекаться вручную, но сейчас это делается путем использования сложных процедур предобработки, которые идентифицируют отдельные слова, многословные выражения, целые предложения и большие синтаксические единицы, которые рассматриваются как концептуальные идентификаторы.

К средствам анализа текста также можно отнести способы визуального представления информации. Данные способы предполагают извлечение из массива ключевых понятий предметной области, которые затем представляются в наглядном графическом виде, например, в виде диаграмм и графиков. Данный подход помогает пользователю быстро идентифицировать главные темы и понятия исследуемой предметной области или процесса, а также определить степень их важности.

Маннинг в своем исследовании выделяет ряд проблем работы с информацией из Web-ресурсов [10].

Одной из таких проблем автор выделяет поиск действительно значимой информации. Существующие поисковые системы среди множества предлагаемых ссылок дают небольшой процент действительно нужной информации. Значительная часть информации в это время зачастую остается неиндексированной, и не попадает в результаты поиска.

Зачастую пользователей поисковых систем интересует решение конкретных проблем, то есть нужны не просто ссылки на Web-контент. В результате отбора собранная поисковой системой информация имеет лишь косвенное отношение к проблеме и содержит большое количество

нерелевантных сведений. Таким образом встает проблема извлечения полезных знаний из найденного множества информации.

Для преодоления перечисленных проблем используется технология web mining. Данная технология использует методы data mining для исследования и извлечения информации из веб-документов и веб-сервисов.

Автор в своих исследованиях выделяет следующие этапы применения технологии web mining [10]:

1. Поиск ресурсов, то есть локализация неизвестных документов и сервисов в веб-среде.

2. Автоматическое извлечение необходимой информации из найденных веб-ресурсов.

3. Обобщение. Подразумевает обнаружение общих шаблонов в отдельных и пересекающихся множествах веб-сайтов.

4. Анализ. На данном этапе происходит интерпретация найденных шаблонов.

В области web mining выделяют ряд направлений анализа информации:

- извлечение web-контента (Web Content Mining);
- извлечение web-структур (Web Structure Mining);
- исследование использования web-ресурсов (Web Usage Mining).

Извлечение web-контента осуществляется при помощи существующих методов извлечения нужной и полезной информации из web-ресурсов, такх как содержание, данные, документы и др. В основном организации предоставляют доступ к своим информационным ресурсам через сеть интернет, в связи с чем возрастает актуальность данного направления. Это относится как к информации в виде HTML-страниц, так и к данным из баз данных организаций, корпоративным порталам и др. Но в то же время остается некоторая часть данных, к которым доступ невозможен из соображений конфиденциальности.

Отличительной особенностью web-ресурсов является разнородность представленной информации. Это могут быть текстовые файлы, изображения, звук, видео, метаданные, а также гиперссылки.

По этим причинам технология web mining достаточно тесно связана с другими направлениями data mining. Например, методы text mining используются для анализа текстовой информации, multimedia mining используется для анализа изображений, видео- и аудиоинформации.

Извлечение web –контента применяется для повышения релевантности получаемой информации и основывается на сочетании информационного поиска, машинного обучения, data mining (рисунок 1.4): 1) Анализ содержания документов: находятся схожие по смыслу слова и их количество. Затем выполняется кластеризация, классификация, ранжирование. Все вместе это группировка документов по смысловой близости.

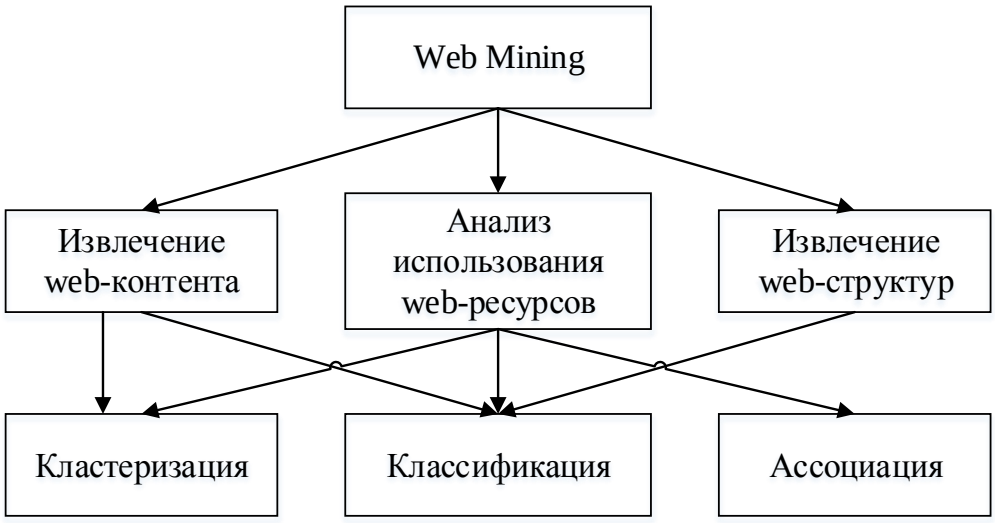


Рисунок 1.4 – Категории Web Mining и задачи Data Mining

2) Кластеризация – это группирование данных по определенному алгоритму. Выполняется без учителя, без подсказок. Заранее неизвестна информация о классах. Методы кластеризации текста разбивают коллекцию на группы связанных документов к конкретным темам или категориям. Тем не менее, эти категории не известны априори, потому что конкретные примеры желаемых категорий (например, политика) документов не предоставляются вначале поиска. Такие проблемы в обучении также называются без учителя, потому что нет руководства к решению проблемы обучения. В контролируемых приложениях можно привести примеры новостных статей, относящихся к нескольким естественным категориям, таким как спорт, политика и так далее.

При неконтролируемой настройке документы разбиваются на похожие группы, что иногда достигается с помощью функции подобия, специфичной для предметной области, такой как мера косинуса. В некоторых случаях может быть сформулирована модель оптимизации таким образом, чтобы были максимально обнаружены некоторые прямые или косвенные модели сходства внутри кластера. 3) Классификация – обучение с учителем. Известны классы, знаем, что на входе и что на выходе.

Извлечение веб-структур – это связи между веб-страницами, категоризация веб-ресурсов, поиск схожих сайтов. На этапе извлечения web-структур происходит построение модели, основанной на топологии гиперссылок, которая отображает взаимосвязи между web-страницами. Построенная таким образом модель может использовать категоризацию web-страниц и быть эффективной для генерации информации о подобности между web-сайтами. Данная категория технологий web mining эффективно используется для распознавания авторских сайтов и обзорных сайтов по определенным темам.

Поиск по ключевым словам, которые можно определить, как последовательность из одного или нескольких слов, обеспечивают компактное представление содержимого документа. Основное преимущество данного метода поиска заключается в ом, что ключевые слова представляют в сжатой форме основное содержание документа. Ключевые слова широко используются и играют важную роль в определении запросов в информационно-поисковых системах, поскольку их легко определить, пересмотреть, запомнить и поделиться. По сравнению с математическими подписями ключевые слова не зависят от любого каркаса и могут применяться к нескольким коллекциям документов.

Такие документно-ориентированные методы поиска извлекают из документа одни и те же ключевые слова независимо от текущее состояние коллекции. Таким образом, методы, ориентированные на документы, предоставляют контекстно-независимые возможности, позволяя использовать

дополнительные аналитические методы, которые характеризуют изменения в текстовом потоке с течением времени, например, сборники опубликованных технических рефератов, которые растут со временем или потоки новостных статей.

С ростом новых информационных технологий в современном информационном обществе перечисленные технологии будут активно развиваться, затрагивая новые области информационной деятельности.

Таким образом, можно предложить ряд возможных решений по повышению эффективности информационного поиска:

1. Использование краулеров. Вместо использования ручного поиска использовать автоматизированный кроулинг и разведочный анализ собранной информации.

2. Использование фильтров (фильтрации): по ключевым словам, по XPath-локаторам, CSS-селекторам, регулярным выражениям.

3. Накопление на серверах информации о пользователях.

Проблема накопления на серверах информации о пользователях в настоящей работе рассматриваться не будет, так как выходит за рамки задач работы.

Выводы по первой главе

В результате проведенного анализа работ отечественных и зарубежных ученых, а также систем информационного поиска Интернет и систем управления знаниями были сделаны следующие выводы:

1. Ввиду наличия в Интернете огромных объемов информации по всем областям знания одной из наиболее актуальных проблем является эффективный поиск и анализ релевантной информации в повседневной работе предприятий и организаций.

2. Используемые методы для обработки информации, имеющейся в сети Интернет, в том числе для поиска информации, и созданные с их применением автоматизированные программные системы реализуют лишь

часть необходимых на самом деле в организациях функций по управлению информацией.

ГЛАВА 2 МЕТОДЫ ПОИСКА И АНАЛИЗА НЕСТРУКТУРИРОВАННОЙ ТЕКСТОВОЙ ИНФОРМАЦИИ

2.1 Анализ существующих автоматизированных систем мониторинга ИТ-сред

В настоящее время освоение современных информационных технологий, таких как технологии Интернет/Интранет, OLAP, Data mining и др. активно развивается в промышленности, медицине, образовании и других сферах деятельности, где роль информации очень важна для принятия управленческих решений. Наряду с символьной информацией (числовой и текстовой) все чаще используются другие формы представления информации, такие как, например, графики и чертежи, карты, рисунки и снимки, звуковая и видеоинформация.

Организации и предприятия в своей повседневной деятельности постоянно создают и используют информационные системы, с каждым годом объем электронных данных и ресурсов увеличивается. Вновь поступающие данные приобретаются и накапливаются, образуя большие объемы неструктурированной информации.

На современном этапе развития информационных технологий ручной способ поиска и обработки информации становится неприемлемым. Автоматизация информационного поиска, обработки и анализа данных позволяет менеджерам компаний своевременно реагировать на изменяющиеся условия функционирования организации, а в следствие чего, повышения ее конкурентоспособности.

Различные поисковые системы имеют свои отличия в стратегии и полноте охвата информационных ресурсов, что часто приводит к тому, что разные средства поиска дают разноречивые ответы на один и тот же запрос. Используя потенциал других средств информационного поиска, можно использовать эти возможности при построении метапоисковой системы. Пример такой системы представлен на рисунке 2.1.

Применение метапоисковых систем зачастую позволяет улучшить некоторые показатели, такие как «полнота» и «объективность» полученных результатов, а также сократить время, затрачиваемое на поиск необходимой информации.

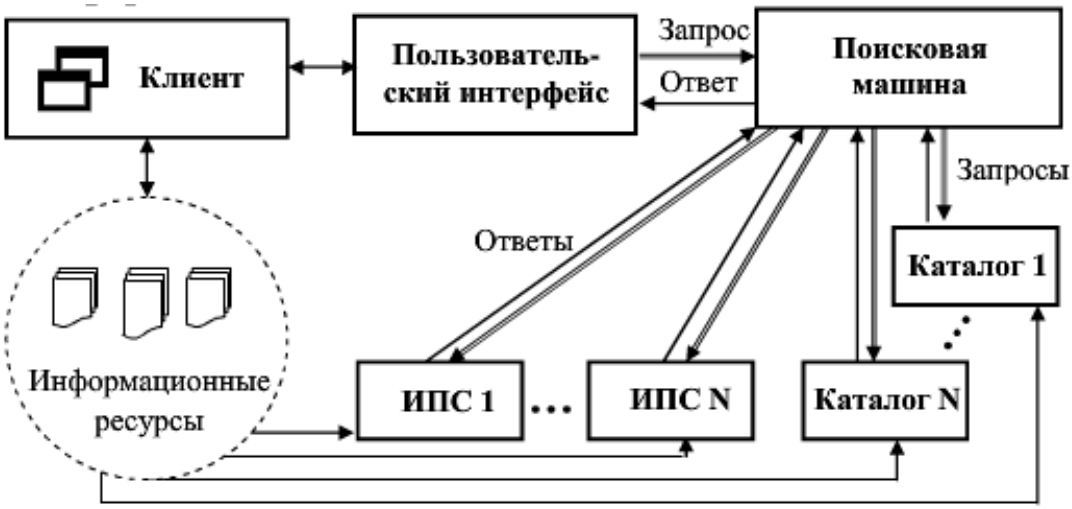


Рисунок 2.1 – Структура системы метапоиска данных

Клиент через пользовательский интерфейс обращается к поисковой машине с запросом на поиск информации. Поисковая машина ищет информацию в разных информационно-поисковых системах и каталогах. Затем ранжирует ответы по степени релевантности и возвращает результат клиенту через пользовательский интерфейс.

На начальных этапах поиска информации такие системы являются наиболее эффективными. Это связано с тем, что они позволяют быстро проверить, существует ли в сети Интернет нужная информация и локализовать средства поиска, в которых она присутствует.

Осуществляя выбор автоматизированных средств мониторинга необходимо учитывать ряд факторов, таких как, например, соответствие функциональных возможностей потребностям организации и ее бизнес-процессам. При выборе функциональности необходимо учитывать потребности разных пользователей. Ими могут быть разработчики, эксплуатационный персонал и другие сотрудники организации. Например, менеджер компании

может быть заинтересован в информации, которая в свою очередь может оказаться полезной и для технических специалистов.

В настоящее время существует множество инструментов мониторинга, однако их пользовательский интерфейс не всегда соответствует современным требованиям пользователей.

Ниже перечислены несколько наиболее широко применяемых инструментов мониторинга ИТ-сред из существующих на рынке:

Nagios является одним из наиболее известных инструментов с открытым кодом для мониторинга ИТ-инфраструктур, в том числе рабочих станций конечного пользователя, ИТ-сервисов и активных сетевых компонентов. так же существует как бесплатная версия с открытым кодом Nagios Core, так и коммерческая Nagios XI, имеющая дополнительные возможности. Имеет современный и простой в навигации веб-интерфейс, предлагающий интерактивную информационную панель с обзором хостов, сервисов и сетевых устройств. Имеется такая возможность, как построение графиков тенденций, а также наличие наглядных инструментов планирования мощности.

Zabbix – система с открытым кодом. Данная система характеризуется высоким быстродействием при сборе данных, масштабируется до корпоративного уровня, а также позволяет вести мониторинг серверов, сетевых устройств и приложений со сбором детальной статистики, касающейся производительности. Рассматриваемая система отличается простотой инсталляции, но сложности может вызывать конфигурирование в случае, если потребуется настраивать особые режимы проверки. Данная система имеет продуманный веб-интерфейс и развитые средства создания отчетов и построения графиков, в стандартный пакет которой помимо функций мониторинга еще входят функции реализации возможностей выявления тенденций.

HP Operations Manager представляет собой центральный компонент комплекса для системного мониторинга от HP. Это клиент-серверное решение, которое в свою очередь требует наличия программных агентов на каждом узле.

Начальная настройка может оказаться непростой в случае если нужно установить несколько пакетов.

HP Operations Manager имеет понятный графический интерфейс пользователя и предназначен для мониторинга состояния приложений, систем и сети. В состав данного решения входят средства планирования, в том числе инструменты прогнозного анализа и моделирования. Сведения о событиях сопровождаются рекомендациями по исправлению ситуации, имеются готовые инструменты и автоматизированные операции устранения неисправностей.

R - язык программирования для статистической обработки данных и работы с графикой, а также свободная программная среда вычислений с открытым исходным кодом [19, 23].

RCrawler – это пакет среды анализа данных R для выгрузки веб-сайтов и извлечения структурированных данных, что можно использовать для широкого использования полезных применений, таких как web mining, text mining, web content mining и web structure mining. Например, можно выгрузить все опубликованные посты блога или извлечь все данные о продуктах на торговых web-сайтах, или собрать все комментарии и резюме web-сайта для индивидуальных исследований отзывов посетителей. Кроме того, RCrawler помогает изучить структуру веб-сайта путем представления в виде сети внешних и внутренних гиперссылок [19]. Кроме того, Rcrawler помогает изучить структуру веб-сайта путем представления в виде сети внешних и внутренних гиперссылок (узлы и дуги).

RCrawler позволяет сканировать, анализировать, хранить страницы, извлекать содержимое и создавать данные, которые могут быть непосредственно использованы для приложений интеллектуального анализа web-материалов. Он также является гибким и может быть адаптирован для других приложений. Основные особенности RCrawler это многопоточный обход, извлечение контента и обнаружение дублированного контента. Кроме того, он включает в себя такие функции, как фильтрация URL-адресов и контента, контроль уровня глубины и анализатор robot.txt, имеет высоко

оптимизированную систему и может загружать большое количество страниц в секунду.

Изучение научной и учебной литературы показало, что отсутствуют серьезные работы по использованию этого пакета в практических приложениях. В основном описывается формат использования отдельных функций пакета.

Отсюда, возникает необходимость разработки алгоритма и сценария веб-кроулинга сайтов с использованием этого пакета. Применение пакетов для поиска на основе R имеет ряд преимуществ:

- R – это мощный функциональный язык программирования;
- применение R позволяет быстро и достаточно просто производить статистическую обработку данных;
- R – программное обеспечение с открытым кодом, что позволяет использовать его свободно и бесплатно;
- R – является кроссплатформенным программным обеспечением;
- существует возможность расширения сценария для решения других задач обработки и анализа информации на сайте, например, статистический анализ, построение графиков, диаграмм, гистограмм и множество других возможностей;
- R используется более чем 2 миллионами специалистов по всему миру;
- R активно развивается и на данный момент разработано уже более 9 тысяч библиотек, облегчающих поиск и анализ, включая большие данные, например, можно дополнительно с поиском использовать возможности пакета tm для анализа больших данных;
- R работает с различными системами графики.

Таким образом, проведенный анализ позволяет сделать вывод о том, что одной из важнейших задач использования информационных ресурсов сети Интернет является усовершенствование современных систем автоматизации мониторинга и системного анализа, а также используемых в них подходов.

Актуальным направлением является разработка методов мониторинга документов во всем информационном массиве Интернета.

2.2 Анализ особенностей поиска неструктурированной информации в Интернете

Text-mining – это область исследований, которая пытается решить проблему информационного кризиса путем использования технологий из data mining, машинного обучения (machine learning), обработки естественных языков (natural language processing (NLP)), поиска информации (information retrieval (IR)) и управления знаниями (nowledge management).

Text-mining включает предобработку коллекции документов (категоризация или каталогизация) текста, извлечение информации и извлечение терминов), сохранение промежуточных представлений, технологии или техники анализа этих промежуточных представлений (такие как анализ распределений, кластеризация, анализ трендов, правила ассоциации) и визуализация результатов.

Text-mining широко можно определить, как интенсивный процесс извлечения знаний из коллекции документов, используя набор инструментов анализа. Аналогично data-mining, text-mining пытается извлекать полезную информацию из источников данных путем идентификации и исследования интересных шаблонов. Однако, в случае text-mining источники данных – это коллекции документов и интересные шаблоны ищутся не среди формализованных записей в БД, а в неструктурированных текстовых данных документов.

Естественно, text-mining использует многое из исследований по data-mining. Поэтому, не удивительно, что системы text-mining и data-mining во многом схожи на высоком уровне архитектуры. Например, оба типа систем основываются на процедурах обработки, алгоритмах поиска шаблонов и элементах уровня представления, таких как инструменты визуализации.

Так как data-mining предполагает, что данные уже сохранены в структурированном формате, большая часть предобработки фокусируется на двух критических задачах:

1. Очистка и нормализация данных.

2. Создание большого числа связанных таблиц.

Наоборот, для систем text-mining операции предобработки опираются на идентификацию и извлечение представительных характеристик (или свойств) из документов на естественных языках. Эти операции предобработки отвечают за преобразование неструктурированных данных, сохраняемых в коллекциях документов в более структурированный промежуточный формат.

Ключевым элементом text-mining является коллекция документов. В простейшем случае это набор или группа текстовых документов. Число документов в таких коллекциях может варьироваться от тысяч до десятков миллионов. Коллекции документов могут быть либо статическими или динамическими, когда документы в коллекции со временем обновляются. Очень большие и быстроизменяющиеся коллекции документов требуют оптимизации производительности всех компонентов системы text-mining.

Примером типичной реальной коллекции документов являются PubMed – это национальная библиотека (репозитарий) аннотаций по биомедицинским исследованиям. PubMed содержит более 29 миллионов ссылок на биомедицинскую литературу [27]. Публикации собираются с 1966 г. по настоящее время, т.е. коллекция динамическая и растущая, по оценкам каждый месяц добавляется примерно 40000 новых биомедицинских аннотаций. Каждый подраздел репозитория PubMed – это значительная коллекция элементов для специфических применений text-mining. Например, сравнительно недавние исследования содержат аннотации документов, содержащих слова протеин и ген, запрос на такие документы возвращает порядка трех миллионов документов, из которых более 66% были опубликованы за последние 10 лет.

Огромный размер коллекций документов, представленных в репозиториях вроде PubMed, делает ручные попытки найти корреляцию между

документами, отобразить сложные отношения между ними или идентифицировать тенденцию чрезвычайно трудоемкой в лучшем случае, или невозможной в худшем случае.

Автоматические методы идентификации и исследования отношений между документами (данными в документах) сильно увеличивают скорость исследовательской деятельности. Часто техники автоматического исследования в text-mining являются не только полезным дополнением, но и основным требованием для практического выполнения исследований по поиску шаблонов в большом числе документов на естественных языках.

Системы text-mining обычно не выполняют алгоритмы поиска знаний в неподготовленных или необработанных коллекциях документов. Значительный акцент в text-mining делается на операциях предобработки. Операции предобработки включают множество разных техник и технологий из поиска информации, извлечения информации, исследования компьютерной лингвистики, которые преобразуют сырой исходный неструктурированный контент в тщательно структурированный промежуточный формат данных. Операции поиска знаний работают с этим структурированным представлением первоначальной коллекции документов.

Другой базовый элемент text-mining – это документ. Документ – это единица дискретных текстовых данных в коллекции (например, файл), которая обычно, но не обязательно, относится к какой-то реальной области (например, отчет по бизнесу, электронная почта, научная статья, монография, просто статья, пресс-релиз или новость).

Таким образом, документ существует в пределах коллекции. Важно понимать, что один и тот же документ может относиться или существовать в различных коллекциях, или в подмножествах, или подмножествах из одной коллекции.

Основными элементарными объектами анализа в информационных массивах могут быть обычные слова и простые словосочетания, а основная информация о них – это частотность и их местоположение в документе. Роль

связей могут играть: совместная встречаемость в одном документе, в том числе с учетом расстояния, относительная совместная встречаемость в коллекции документов.

Даже самые простые документы содержат большое количество слов, фраз, предложений, типографских элементов, артефактов раскладки, не говоря уж о разных смыслах разных элементов (одно и то же слово может означать разный смысл) и представляют собой серьезную проблему для большинства систем text-mining для идентификации упрощенного подмножества свойств документа, которые могут представлять отдельный документ как целое. Такое множество свойств можно назвать моделью представления документа.

В частности, одним из шагов предобработки является отбрасывание из документа стоп-слов, наподобие союзов, предлогов, связок, глаголов типа «есть», которые не несут смысла. Эти слова в модель представления не входят.

Огромное число характеристик из необходимых для представления документов коллекции влияет почти на все аспекты разрабатываемой системы data-mining, на его концепцию, дизайн, производительность.

Структурированное представление документов на естественных языках требует значительно большего числа представительных характеристик, намного больше, чем в реляционных или иерархических БД. Например, сравнительно маленькая коллекция из 15 тысяч документов - лента новостей Reuters – требует идентификации более 25 тысяч слов, в базах данных это намного-намного меньше.

Проблема большой размерности характеристик коллекции документов является критической для всех text-mining систем. Другой особенностью документов на естественных языках является разреженность характеристик, т.е. только маленький процент возможных характеристик всей коллекции встречается в отдельном документе. Поэтому, когда документ представляется как бинарный вектор характеристик, почти все значения этого вектора равны нулю. Это же относится к размерности кортежей, т.е. отдельные характеристики часто встречаются только в небольшом числе документов.

Поиск текстовых данных в больших коллекциях электронных документов включает в себя несколько сложных этапов. Это происходит потому, что тексты, с точки зрения компьютера, представляют собой довольно неструктурированные наборы слов. Анализ массива неструктурированной информации обычно начинается с набора сильно разнородных исходных текстов. Так первым шагом является импорт этих текстов в любимую вычислительную среду, в нашем случае R. Одновременно важно организовать и структурировать тексты, чтобы иметь возможность доступа их в едином стиле. Как только тексты организованы в хранилище, на втором шаге производится предварительная обработка текста. Делается это для того, чтобы получить удобное представление для последующего анализа. Этот шаг может включать переформатирование текста (например, удаление пробелов) или удаление стоп-слов. Далее необходимо преобразовать предварительно обработанные тексты в структурированные форматы для фактического вычисления. Зачастую задачи интеллектуального анализа текста, как правило, подразумевают создание так называемой матрицы терминов, это самый распространенный формат для представления текстов для исследования. Далее аналитик может работать и вычислять тексты со стандартными методами из статистики и интеллектуального анализа данных, таких как методы кластеризации или классификации.

Эта стандартная модель процесса выделяет важные шаги, которые требуют поддержки со стороны инфраструктуры интеллектуального анализа: платформа интеллектуального анализа текста должна предлагать функциональные возможности для управления документами, должна абстрагироваться от процесса манипулирования документами и облегчить использование разнородных текстовых форматов.

Поскольку текстовые данные в сети Интернет представлены в разных форматах и в различных местах, например, сжатый файл в Интернете или локально сохраненный текстовый файл с дополнительными аннотациями. Помимо фактических текстовых данных многие современные форматы файлов

предоставляют функции для аннотирования текста документы (например, XML со специальными тегами), то есть доступны метаданные, которые дополнительно описывают и обогащают текстовое содержимое и могут предложить ценную информацию о документе структура или дополнительные понятия. Поэтому автоматизированная поисковая система должна обеспечивать использование метаданных в удобной форме, как на уровне документа (например, краткие резюме или описания выбранных документов), так и на уровне коллекции документов (например, теги классификации всей коллекции).

Другая важная концепция - фильтрация, которая в основном включает применение предикатных функций в коллекциях документов, чтобы извлечь интересующие наборы слов. Объединение наборов документов позволяет не сложным способом осуществлять поиск, в отличие от объединения метаданных, которое нуждаются в более сложной обработке, поскольку хранение метаданных из разных источников в последовательных шагах обязательно приводит к иерархической древовидной структуре. Автоматизированные сценарии в text-mining используют для обработки как минимум несколько сотен текстовых документов, вплоть до несколько сотен тысяч документов.

Таким образом можно сформулировать *проблемы поиска* информации в Интернете [15, 18]:

1. Пользователь не может сразу найти все необходимые ресурсы, поисковые системы на запрос пользователя возвращают небольшой процент релевантных ссылок.

2. Вторая сложность извлечения новых знаний из обнаруженных источников, поскольку информация не структурирована (текстовые документы) и требуют осмысления сведений, заложенных разными авторами.

3. Сложность адаптации серверов под конкретных пользователей. Например, подсказки по товарам с учетом требований покупателя.

Изучив проблемы поиска информации в web нами предложены возможные решения:

1. Использование краулеров. То есть автоматизированный кроулинг и разведочный анализ собранной информации.

2. Использование фильтров (фильтрации): по ключевым словам, по XPath-локаторам, CSS-селекторам, регулярным выражениям.

3. Накопление на серверах информации о пользователях.

Существует ряд методов, с помощью которых происходит извлечение полезной информации из web-ресурсов, таких как содержание, данные, документы и другие. В настоящее время многие компании предоставляют доступ к своим информационным ресурсам не только в виде HTML-страниц, но так и к данным из баз данных организаций, корпоративным порталам и др. Несомненно, остается некоторая часть данных, к которым доступ невозможен из соображений конфиденциальности.

2.3 Методы поиска и анализа информации на отдельных сайтах

В данном разделе приведен анализ методов поиска необходимой информации на отдельных we-сайтах.

1 Способ. Мониторинг работы сайта с использованием лог-файлов или журналов

Структура лог-файлов веб-сервера, или журналы веб-серверов:

- Поле "удаленный хост". Содержит веб-адрес или доменное имя хоста, создавшего запрос.
- Поле "дата/время". Время поступления запроса.
- Поле "HTTP запроса". Содержит четыре части: метод запроса, унифицированный индикатор ресурса (URI) – это имя страницы или путь к ней, заголовок, протокол передачи данных.
- Поле кода состояния. Не все запросы успешны. Это трехзначное число (xxx). Если начинается с 2, то запрос успешный.
- Поле переданного количества данных от сервера к клиенту.

2 Способ. Поиск информации с использованием xPath локаторов и CSS селекторов

Ниже перечислены типы локаторов элементов:

- id;

- атрибут name;

- text. Text удобен для поиска ссылок и кнопок на страницах, содержащих видимый текст, но не рисунок с текстом. Ищет названия(метки) кнопок, входные значения, текст ссылок, метки, элементы с атрибутом title;

- линк-текст;

- класс CSS. Находит элементы, которые имеют уникальный стиль на странице;

- XPath – это язык для перемещения по структуре DOM (объектная модель документа). XPath локаторы очень мощные и гибкие. Любой элемент на странице можно локализовать с помощью одного или нескольких (более) XPath. Хорошо написанный XPath может быть очень устойчивым;

- селектор CSS – аналогичны XPath. В отличие от XPath они не основываются на структуре DOM (т.е. не перемещаются по структуре документа), однако они могут легко делать вещи, которые иногда трудно сделать с помощью XPath.

XPath locator

Абсолютный Xpath. Один слеш вначале означает начало поиска от корня (Абсолютный XPath).

Относительный Xpath. Два слеша вначале означает начало поиска из любого места структуры DOM (короче, чем Абсолютный XPath).

Чтобы найти ссылку на странице, XPath перемещается по иерархии от корневого к нужному элементу документа.

```
<html><body>
<p>The fox jumped over the lazy brown <a
href="dogs.html">dog</a>.</p>
</body></html>
```

Иерархия (как отыскать ссылку a в вышепредставленном примере):

```
/html/body/p/a
```

XPath может найти элемент по его идентификатору id так:

```
//*[@id="element_id"]
```

Например, если нужно найти элемент, ближайший к другому элементу с id, например, ссылку:

```
<html><body>
<p id="fox">The fox jumped over the lazy brown <a
href="dogs.html">dog</a>.</p>
</body></html>
```

можно задать XPath следующим образом:

```
//*[@id="fox"]/a
```

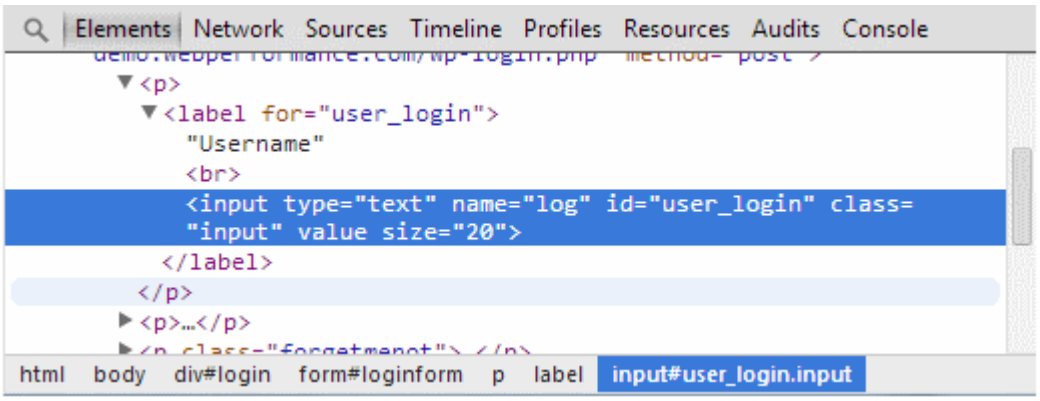


Рисунок 2.2 - Поиск страницы идентификации

Существует два способа объявить кнопку в HTML:

1. С помощью тега button

```
//button(contains(., 'press me')]
```

2. Кнопка submit-форма, в скобках объявляется тип ”submit” или ”button”

Тогда XPath будет иметь следующий вид:

```
//input[@value='press me']
```

Иногда часть текста оформляется как ссылка или кнопка. Чтобы найти его, используется:

```
//*[text()='the visible text']
```

Чтобы найти n-тый элемент, нужно заключить ваш XPath в скобки и задать n в квадратных скобках:

```
(xpath)[n]
```

Пример, найти третью ссылку на странице:

```
(//a)[3]
```

Пример 2, найти четвертое текстовое входное поле на странице:

```
(//input[@type="text"])[4]
```

Можно также перемещаться от индексированного элемента. Пример, найти ссылку во втором div класса abc:

```
(//div[@class='abc'])[2]/a
```

Для установки расширения xPath Finder необходимо выполнить ряд действий:

1. Зайти на сайт по ссылке https://chrome.google.com/webstore/detail/xpath-finder/ihnknokegkbpmofmafnkoadfjkhlogph/related и нажать кнопку Установить. Нажать на любой элемент, чтобы получить xPath. Плагин для получения элементов xPath.

Для использования необходимо выполнить ряд следующих действий:

1. Нажмите на значок плагина  (в правом верхнем углу), курсор изменится на перекрестие  .

2. Наведите указатель мыши на нужный элемент (элементы выделяются при наведении курсора).

3. Нажмите на элемент, и его xPath отобразится на панели внизу страницы (слева).

Доступные при использовании опции:

- включить / выключить инспектора;
- включить / выключить автоматическое копирование в буфер обмена;
- выбирать между короткими идентификаторами или обычным путем4
- изменить положение поля xpath.

3 Способ. Поиск информации с помощью регулярных выражений

Регулярные выражения – язык, используемый для разбора и манипулирования текстом. Они часто используются для выполнения сложных операций поиск/замена, а также для подтверждения, что текст хорошо оформлен [24].

В настоящее время регулярные выражения включаются в большинство языков программирования, во многие языки скриптов, редакторы, приложения, инструменты командной строки.

Регулярное выражение – это строка, содержащая обычные символы и специальные метасимволы или мета-последовательности. Обычные символы согласуются так, как они есть. Метасимволы – это символы или последовательности символов, которые представляют идеи, такие как количество, положение или тип символов.

Список в «Regex Metacharacters, Modes, and Constructs» содержит большинство употребительных метасимволов и мета-последовательностей в мире регулярных выражений.

Можно предсказать результаты большинства согласований, применяя в уме два правила:

1. Побеждает самое раннее (самое левое) согласование. Как только движок найдет согласование, он завершает работу.

2. Стандартные кванторы жадные. Кванторы определяют, сколько раз что-то можно повторить. Стандартные кванторы пытаются согласовать как можно больше раз, поэтому жадные. Они выполняются меньшее число раз только если это нужно для успеха поиска. Процесс отбрасывания символов и использования менее жадных согласований называется бэктрекингом (backtracking).

Существует два класса движков: детерминированные конечные автоматы (DFA) и недетерминированные конечные автоматы (NFA).

DFA более быстрые, но им недостает таких характеристик NFA, как захват (capturing), поиск (lookaround) и нежадные кванторы (nongreedy quantifiers).

Среди NFA различают два типа: традиционные и POSIX.

\n – новая строка

\num – две или три восьмиричные цифры

\xnum, \x{num}, \unum, \Unum – представляет шестнадцатиричные числа. Четыре цифры и больше могут представлять диапазон символов юникода. Пример: \x0D\x0A соответствует ASCII-последовательности CR/LF.

\cchar – соответствует управляющим символам ASCII-кода (первые 32 символа в таблице ASCII-кодов, например, \cH – Backspace).

Классы символов используются для задания множества символов. Символ на входе движка считается согласованным, если он имеется в заданном множестве.

[...] – перечисление множества символов для согласования;

[^...] – перечисление символов для несогласования.

“-“ (dash) – означает диапазон символов, например, [a-z] - согласование всех строчных английских букв, [^0-9] – не должны содержаться цифры. Чтобы включить “-“ в список символов, либо поставьте его первым элементом множества, либо укажите его со слешем (\-).

dot (.) (точка) – обычно согласовывает любой символ, кроме символа новой строки.

\w, \d, \s, \W, \D, \S – класс сокращений. Сокращения для символов слов, цифр, пробелов.

\w – все алфавитно-цифровые символы ASCII-кода (буквы и цифры) плюс символ подчеркивания. Алфавитно-цифровые символы могут включать дополнительно локальные и Unicode-символы.

\s, \S. \s согласовывает символы из класса, \S согласовывает символы не из класса.

\d – согласовывает одну цифру и обычно эквивалентно [0-9].

[:alnum:] – класс символов POSIX. Определяет несколько классов символов, например, [:lower:] если записано [[:lower:]], то это эквивалентно [a-z].

^, \A – согласует начало строки текста.

$, \Z, \z - согласует конец строки текста. В многострочном режиме знак $ согласует сначала каждой новой строки. \Z – согласует конец строки или точки перед каждым переходом на новую строку. \z – согласует только конец строки.

\G – в интерактивном согласовании согласует позицию где закончилось предыдущее согласование, часто устанавливается в начало строки, если предыдущее закончилось неудачей.

\b, \B, \<, \> - границы слов. Определяют позицию, в которой после текстового символа следует нетекстовый. \b часто задает положение границы слова, а \B – отрицание этого. В некоторых реализациях “\<” означает начало слова, а “\>” – конец слова.

(?=...) – ищет субсогласования вперед, а (?!...) – не ищет.

(?<=...), (?<!...) – то же самое, но назад. Например, foo(?=bar) – согласовывает foo в foobar, но не в food.

Функции, которые работают с множествами:

?intersect() – вызов справки по этим функциям.

union(x, y)

intersect(x, y)

setdiff(x, y)

setequal(x, y)

Если нужно импортировать данные в табличном формате, то можно использовать ряд функций для чтения таблиц.

Эти функции считывают данные и создают фрейм данных.

read.table() – главная функция для чтения в табличном формате (разделитель можно указать),

read.csv() – csv-файл с разделителем «,»,

read.csv2() – csv-файл с разделителем «;»,

read.delim(),read.delim2() – читает файла с разделителем «табуляция»,

read.fwf() – читает файлы с фиксированной (заданной) шириной элементов.

Для импорта текста как он есть, используется функция `readLines()`.

Регулярные выражения (regex) – это специальная текстовая строка для описания некоторого количества текста, называемая шаблоном (pattern). Следовательно, регулярные выражения – это шаблон, который описывает множество строк.

\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b - регулярное выражение для поиска любого email-адреса.

^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\$ - шаблон позволяет программисту проверять, правильно ли пользователь ввел email-адрес.

Самое основное регулярное выражение – это один символ, например, a. Он согласуется с первым таким символом в тексте. Чтобы найти другие совпадения, нужно подсказать движку регулярных выражений продолжать поиск по тексту. Например, в текстовых редакторах продолжение поиска

осуществляется с помощью кнопки Next. В языках программирования для продолжения поиска используется специальная функция.

Существует 12 символов, имеющих специальное назначение в регулярных выражениях. Эти специальные символы называются метасимволами. Большинство из них приводит к ошибке, если их использовать индивидуально. Если нужно использовать любой из этих символов как букву в регулярном выражении, то перед ним нужно поставить “обратный слеш”.

Например, если нужно согласовать выражение 1+1=2, то правильная запись будет такой 1\+1=2. В противном случае знак + будет считаться метасимволом и имеет специальное значение.

Класс символов согласует один из нескольких символов. Например, класс [ae] согласует либо a либо e. Согласуется только один символ. Например, graay, graey не согласуется. Такие gray или grey согласуются. [0-9] – любая цифра от 0 до 9. Можно использовать больше одного диапазона. Например, [0-9a-fA-F] согласует любую 16-ричную цифру без учета регистра.

Можно комбинировать диапазоны и отдельные буквы. Например, шаблон [0-9a-fxA-FX] согласует любую 16-ричную цифру или букву *X* без учета регистра.

Все приведенные выше классы согласуют одну букву текста.

^ после открывающейся [отрицает данный класс символов. Например, шаблон q[^x] согласует qu в слове question, т.е. согласует все буквы, кроме x.

Близко связанными с классами символов регулярных выражений классы символов POSIX (таблица 2.1).

Таблица 2.1 - Классы символов POSIX

Класс	Описание
[[:lower:]]	Lower-case latter
[[:upper:]]	upper-case latter
[[:alpha:]]	Alphabetic characters ([[:lower:]] and [[:upper:]]
[[:digit:]]	Digit: 0,1,2,3,4,5,6,7,8,9
[[:alnum:]]	Alphanumeric characters (([[:alpha]] and [[:digit:]]
[[:blank:]]	Blank characters: space and tab
[[:cntrl:]]	Control characters
[[:punct:]]	Punctuation characters: ! " #%&'()*+,-./:;
[[:space:]]	Space characters: tab, newline, vertical tab, form feed, carriage return, and space
[[:xdigit:]]	Hexadecimal digits: 0-9 A B C D E F a b c d e f
[[:print:]]	Printable characters ([[:alpha:]]), [[:punct:]] and space)
[[:graph:]]	Graphical characters ([[:alpha:]] and [[:punct:]])

Функции для регулярных выражений перечислены в таблице 2.2:

Таблица 2.2 – Функции регулярных выражений в R

function	Purpose	Characteristic
grep()	finding regex matches	Which elements are matched (index or value)
grepl()	finding regex matches	which elements are matched (TRUE &FALSE)
regexpr()	finding regex matches	positions of the first match
gregexpr()	finding regex matches	positions of all matches
regexes()	finding regex matches	hybrid of regexpr() and gregexp()
sub()	replacing regex matches	only first match is replaced
gsub()	replacing regex matches	all matches are replaced
strsplit()	splitting regex matches	split vector according to matches

Функция grep() используется для поиска согласований в символьных векторах, разница между ними в формате вывода.

sub() и gsub() – используются для замены

strsplit() – используется для расщепления элементов символьного вектора на подстроки в соответствии с согласовании.

В целом все функции требуют два главных аргумента: паттерн (т.е. регулярные выражения) и текст, в котором ищутся согласования.

Каждая функция имеет другие дополнительные аргументы, но главными являются эти два.

Функция grep(). Возвращает индексы элементов символьного вектора, в которых имеются согласования.

Горячие классы символов (или сокращенные):

\d – согласует любую цифру.

\w – согласует любые алфавитно-цифровые символы плюс подчеркивание.

\s – согласует символ пробелов, включая табуляцию и разрывы строк (переход на новую строку).

Непечатаемые символы:

\t – символ табуляции (ASCII 0x09).

\r – возврат каретки (0x0D).

\n – переход на новую строку (0x0A).

\a – звонок (сигнал) (bell, 0x07).

\e (escape, 0x1B)

Если ваше приложение поддерживает unicod, то используйте \uFFFF или \x{FFFF}. Например, шаблон \u20AC или \x{20AC} означает знак европейской валюты (евро).

Если ваше приложение не поддерживает unicod, то используйте \xFF.

Все непечатаемые символы можно использовать непосредственно в регулярных выражениях или как часть символов, или как часть класса символов.

Метасимвол “.” (точка). Согласует один символ, кроме символов разрыва (переход на новую строку). Например, gr.y согласует любой символ, кроме перехода на новую строку.

Часто класс символов или отрицание класса символов работает более быстрее и более точно.

Якори. Не согласуют символы, они согласуют позицию. Например, ^ согласует начало строки, $ согласует конец строки.

Большинство движков регулярных выражений имеют режим multiline (многостроковый), в котором ^ согласует начало очередной строки, а $ означает символ перед переходом на новую строку (конец строки). Пример, ^b согласует первое слово, которое начинается на маленькую b.

\b – это позиция между символом, который согласуется шаблоном \w и символом, который не согласуется \w. \B согласует любую позицию, которую не согласует \b.

Alternation (Чередование). Эквивалентно логическому or. Например, cat|dog согласует cat в предложении About cats and dogs. Если в регулярном выражении встречается слово, то ищется dog. Можно использовать произвольное число альтернатив: cat|dog|mouse|fish. Удобно использовать для поиска по ключевым словам.

Альтернативы имеют низший приоритет в регулярных операторах и операциях. Пример, cat|dog food согласуют cat или dog food. Чтобы создать выражение, которое согласует cat food или dog food, нужно сгруппировать альтернативы: (cat|dog) food.

Метасимвол ‘?’. Делает предшествующий символ необязательным, например, colou?r согласует colour или color.

Метасимвол ‘*’. Говорит движку делать согласование предшествующего символа **0** или большее число раз.

Символ ‘+’ говорит движку согласовывать предшествующий символ **1** и большее число раз.

<[A-Za-z0-9]+> согласовывает любой html-тег без атрибутов.

Используйте {} для указания конкретного числа повторений. Например, \b[1-9][0-9]{3}\b согласует число между 1000 и 9999. \b[1-9][0-9]{2,4}\b согласует число между 100 и 99999.

Операторы повторения (или квантифаеры) жадные. Они расширяют согласования как можно дальше. Пример, <.+> согласовывает <EM>first</EM> в предложении This is a <EM>first</EM> test. Чтобы сделать ленивые повторения, после знака повторения (* или +) нужно поставить знак ?. Например, <.+?> согласует <EM> в предыдущем предложении.

<[^<>]+> быстрое согласование html-тега с атрибутами либо без атрибутов.

Чтобы сгруппировать много символов, поместите их в скобки. Затем к группе можно применить квантифаер (сколько раз повторить), например, Set(Value)? согласует Set или SetValue. Скобки создают группу захвата.

Предыдущий пример содержит одну такую группу. После согласования слова Set группа 1 не содержит ничего, и содержит Value, если согласовано SetValue, как обеспечивается доступ к содержимому групп зависит от ПО или языка программирования. Группа ноль всегда полное согласование regex. Используйте специальный синтаксис Set(?:Value)? для группировки символов без создания захватывающих групп. Это более эффективно, если вы не планируете использовать содержимое групп.

Можно использовать ссылку “назад” \1, чтобы согласовать такой же текст, который был согласован предыдущей группой захвата (захватывающей группой). Например, ([abc])=\1 согласует a=a, b=b, и c=c, и ни что другое.

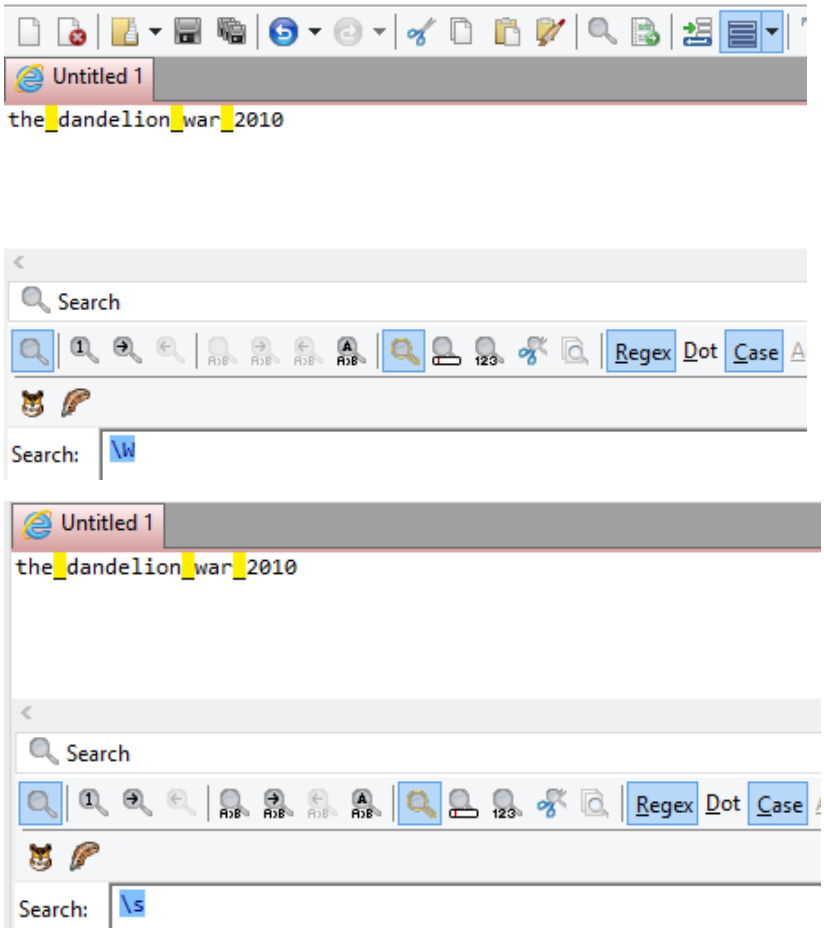
Если выражение содержит много захватывающих групп, они нумеруются по открывающимся скобкам слева направо.

Справку по регулярным выражениям в R можно получить с помощью функции help(regex).

Функция sub() заменяет первое согласование, а gsub() - все совпадения.

```
> sub("\\d", "_", "the dandelion war 2010")
[1] "the dandelion war _010"
> gsub("\\d", "_", "the dandelion war 2010")
```

```
[1] "the dandelion war ____"


> sub("\\D", "_", "the dandelion war 2010")
[1] "_he dandelion war 2010"
> gsub("\\D", "_", "the dandelion war 2010")
[1] "__________________2010"
> gsub("\\D", "_", "the dandelion war; 2010")
[1] "____________________2010"


> sub("\\s", "_", "the dandelion war 2010")
[1] "the_dandelion war 2010"
> gsub("\\s", "_", "the dandelion war 2010")
[1] "the_dandelion_war_2010"


> sub("\\S", "_", "the dandelion war 2010")
[1] "_he dandelion war 2010"
> gsub("\\S", "_", "the dandelion war 2010")
[1] "___ _________ ___ ____"


> sub("\\b", "_", "the dandelion war 2010")
[1] "_the dandelion war 2010"
> gsub("\\b", "_", "the dandelion war 2010")
[1] "_t_h_e_ _d_a_n_d_e_l_i_o_n_ _w_a_r_ _2_0_1_0_"


> sub("\\B", "_", "the dandelion war 2010")
[1] "t_he dandelion war 2010"
> gsub("\\B", "_", "the dandelion war 2010")
[1] "t_he d_an_de_li_on w_ar 2_01_0"


> sub("\\w", "_", "the dandelion war 2010")
[1] "_he dandelion war 2010"
> gsub("\\w", "_", "the dandelion war 2010")
[1] "___ _________ ___ ____"


> sub("\\W", "_", "the dandelion war 2010")
[1] "the_dandelion war 2010"
> gsub("\\W", "_", "the dandelion war 2010")
[1] "the_dandelion_war_2010"
```



Для вызова справки по функции в окне команд набрать:

```
library(Rcrawler)
?grep
```

Функции `grep()`, `grepl()`, `regexpr()`, `gregexpr()` and `regexec()` ищет согласования для агрумента `pattern` с каждым элементом после `pattern` символьного вектора: они различаются форматом и суммой подробностей в результате.

Функции `sub()` and `gsub()` выполняют замену первого и всех совпадений (или согласований) соответственно.

Аргументы функции:

`pattern` – символьная строка, содержащая регулярное выражение. Автоматически преобразуется в строку, если это возможно (приведение типов). Если `pattern` двумерный и более вектор, то используется первый элемент вектора с предупреждением. Допускаются отсутствующие значения (NA), кроме функции `regexpr()` и `gregexpr()`;

`x, text` – символьный вектор, в котором ищется согласование, или объект, который может быть преобразован в символьный вектор. Поддерживаются длинные векторы;

`ignore.case` – логический, если FALSE, то согласования (совпадения) чувствительны к регистру букв, иначе – нет;

`perl` – логический, должны ли использоваться регулярные выражения, совместимые с `Perl`, по умолчанию FALSE (т.е. используется R);

`value` – логический, если FALSE, то возвращается вектор, содержащий целочисленные индексы согласований, найденные функцией `grep()`. Если TRUE, то возвращаются найденные совпадения:

```
> s <- c("Пример", "Прием", "Каприз", "Опера")
> pattern <- "ри"
> grep(pattern, s)
[1] 1 2 3
> s <- c("Пример", "Каприз", "Опера", "Прием")
> pattern <- "ри"
> grep(pattern, s)
[1] 1 2 4
> grep(pattern, s, value = T)
[1] "Пример" "Каприз" "Прием"
```

`fixed` – логический, если TRUE, то `pattern` – это строка, как она есть (фиксированный шаблон, над которым не делается никаких преобразований);

`useBytes` – логический, если TRUE, то согласование делается побайтово, а не по символам;

`invert` – логический, если TRUE, то возвращаются индексы несогласованных элементов;

`replacement` – замена для согласований в функциях `sub()` и `gsub()`. Подробности см. в Справке.

Для функций `regexpr()`, `gregexpr()` и `regexec()` возникает ошибка, если `pattern = NA`, в других функциях она разрешается и согласование равно `NA`.

NA – это значение или элементы вектора (фрейма), которые по какой-то причине не известны (или отсутствуют) (например, сбой при вводе, не сработал

датчик, ошибочные значение, которые неизвестно на что заменить). В R есть функции, которые работают с этими значениями NA.

Функция `grep(value = FALSE)` – вектор возвращает индексы согласованных элементов x (т.е. которые удовлетворяют условию), (или несогласованных, если `invert = TRUE`).

Функция `grep(value = TRUE)` – возвращает символьный вектор, содержащий согласованные элементы x.

Функция `grepl()` – возвращает логический вектор (согласование или нет для каждого элемента x).

Функции `sub()` и `gsub()` - возвращают символьный вектор такой же длины и с такими же атрибутами, как и x с заменой.

Функция `regexpr()` – возвращает целочисленный вектор такой же длины, как и text, содержащий начальную позицию первого совпадения в элементах text (или возвращается -1, если согласований нет), с атрибутом "`match.length`" – целочисленным вектором, содержащим длину согласований в элементах (или -1, если согласования нет):

```
> s <- c("Пример", "Каприз", "Опера", "Прием")
> pattern <- "ри"
> regexpr(pattern, s)
[1]  2  4 -1  2
attr(,"match.length")
[1]  2  2 -1  2
```

Функция `gregexpr()` – возвращает список такой же длины, как длина `text`, каждый элемент которого имеет такую же форму, как и значение, возвращаемое функцией `regexpr()`:

```
> s <- c("Пример", "Каприз", "Опера", "Прием прихожан")
> pattern <- "ри"
> gregexpr(pattern, s)
[[1]]
[1] 2
attr(,"match.length")
[1] 2

[[2]]
[1] 4
attr(,"match.length")
```

```
[1] 2

[[3]]
[1] -1
attr(,"match.length")
[1] -1

[[4]]
[1] 2 8
attr(,"match.length")
[1] 2 2
```

Функция `regexec()` – возвращает список такой же длины, как длина `text`, каждый элемент которого равен -1, если нет совпадения, или последовательности целых чисел со стартовой позиции согласования и всеми подстроками, соответствующими выражениям в скобках в `pattern`, с атрибутом `"match.length"` – вектором, содержащим длину согласования (или -1, если нет согласования):

```
## Раскладывает URL на его компоненты
> x <- "http://stat.umn.edu:80/xyz"
> m <- regexec("^(([^:]+)://)?([^:/]+)(:([0-9]+))?(/.*)", x)
> m
[[1]]
[1]  1  1  1  8 20 21 23
attr(,"match.length")
[1] 26  7  4 12  3  2  4
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE

> regmatches(x, m)
[[1]]
[1] "http://stat.umn.edu:80/xyz"  "http://"  "http"
[4] "stat.umn.edu"  ":80"  "80"  "/xyz"
```

Функция `regmatches()` – извлекает или заменяет согласованные подстроки в согласованных данных, получаемых функциями `regexpr()`, `gregexpr()` и `regexec()`.Например:

```
> s <- c("Пример", "Каприз", "Опера", "Прием прихожан")
> pattern <- "ри"
> m <- gregexpr(pattern, s)
> regmatches(s, m, invert = FALSE)
[[1]]
[1] "ри"
[[2]]
```

```
[1] "ри"
[[3]]
character(0)
[[4]]
[1] "ри" "ри"
```

Выводы по второй главе

Во второй главе рассмотрены методы мониторинга веб-страниц отдельных сайтов. Первый метод позволяет провести статистический анализ, но отсутствует доступ к информации. С помощью данного способа можно определить рейтинг страниц или сайта, время запросов, частоту посещений, объем информации, переданной каждому пользователю и т.д. Недостатком данного способа является то, что достаточно часто пользователям запрещен доступ к лог-файлам и журналам.

Второй способ представляет собой расширенный поиск по идентификаторам элементов html-страниц, стилям форматирования, например, выделенные слова на страницах. Можно осуществить поиск по заголовкам разных уровней, параграфам, ссылкам и т.д.

Третий способ является самым глубоким поиском с использованием регулярных выражений, определенных шаблонов в тексте, например, ключевых слов, синтаксических выражений и т.п.

Таким образом, можно вторым методом определить коллекцию релевантных страниц на сайте, т.е. осуществить поиск, а третьим методом выполнить автоматизированный интенсивный анализ содержимого страниц.

ГЛАВА 3 РАЗРАБОТКА СЦЕНАРИЯ ДЛЯ АВТОМАТИЗИРОВАННОГО ПОИСКА И АНАЛИЗА НЕСТРУКТУРИРОВАННОЙ ИНФОРМАЦИИ В СЕТИ ИНТЕРНЕТ

3.1 Выбор среды разработки

Основной отличительной особенностью web-ресурсов является разнородность представленной информации. Информация может быть представлена в виде текстовых файлов, изображений, звука, видео, метаданных, а также гиперссылок. По этим причинам технология Web Mining достаточно тесно связана с другими направлениями Data Mining. Например, методы Text Mining используются для анализа текстовой информации, Multimedia Mining используется для анализа изображений, видео- и аудиоинформации.

R - язык программирования для статистической обработки данных и работы с графикой, а также свободная программная среда вычислений с открытым исходным кодом [19, 23].

RCrawler – это пакет среды анализа данных R для выгрузки веб-сайтов и извлечения структурированных данных, что можно использовать для широкого использования полезных применений, таких как web mining, text mining, web content mining и web structure mining.

Rcrawler – главная функция пакета, которая, используя URL веб-сайта и шаблоны Xpath и селекторы CSS может перемещаться по всему сайту и автоматически извлекать содержимое веб-страниц для получения структурированного набора данных. Операция кроулинга выполняется несколькими процессами параллельно, поэтому рекомендуется использовать 64-битную версию R. Основными параметрами функции являются:

Website – корневой URL веб-сайта для кроулинга.

no_cores – тип данных целый, задает число кластеров (логических процессоров) для параллельного кроулинга. По умолчанию = число имеющихся ядер.

no_conn – тип данных целый, число параллельных соединений на одно ядро. По умолчанию = значению no_cores.

Encod – кодирование страниц веб-сайта, по умолчанию кроулер автоматически определяет тип кодирования страниц (текста) сайта.

Timeout – максимальное время ожидания ответа на запрос (чтобы не тратить время на ответы от медленных серверов или огромных страниц), по умолчанию = 5 секунд.

urlregexfilter – символьный вектор, это шаблон регулярного выражения, который используется для извлечения страниц с контентом, удовлетворяющим этому выражению. Позволяет отфильтровать станички, которые удовлетворяют требованиям.

KeywordsFilter – символьный вектор, позволяет скачивать только страницы, содержащие только указанные ключевые слова. Параметр должен быть вектор хотябы с одним ключевым словом, например, c("mykeyword").

KeywordsAccuracy – точность ключевых слов, целое значение от 1 до 100, используется с предыдущим параметром KeywordsFilter. Значение рассчитывается исходя из числа совпавших ключевых слов и их частотности (как часто встречаются).

ExtractXpathPat – символьный вектор, вектор шаблонов XpathPat для выгрузки данных.

ExtractCSSPat – символьный вектор, вектор шаблонов селекторов CSS для выгрузки данных.

NetworkData – если true, то кроулер отображает все внутренние связи по гиперссылкам и возвращает DATA для создания сети связей.

В таблице 2.3 представлено сравнение некоторых популярных пакетов для сбора данных.

Таблица 3.1 - Сравнение популярных пакетов для сбора данных

Название пакета	**Сканирование**	**Извлечение информации**	**Анализ**	**Описание**
scrapeR	-	+	+	Используя URL веб-сайта и шаблоны

Название пакета	Сканирование	Извлечение информации	Анализ	Описание
				XPath извлекает с веб-страниц интересующую информацию
Rvest	-	+	+	Извлекает фрагменты из HTML-документов с помощью селекторов XPath и CSS
RCrawler	+	+	+	Сканирование web-сайтов и извлечение их контента с использованием различных методов
RSelenium	-	-	+	Автоматизация браузера

Чтобы запустить Rcrawler, нужно предоставить корневой URL интересующего сайта. URL может быть доменом, поддоменом или разделом веб-сайта.

По мере выгрузки страниц сайта в глобальном окружении R обновляется (или дополняется) фрейм данных INDEX, который содержит адрес ссылки, тип контента, состояние http, число выходных и входных ссылок, тип кодирования, уровень. Дальше Rcrawler создает каталог, содержащий выгруженные веб-страницы. Если при выгрузке используются параметры ExtractXpathPat или ExtractCSSPat, то в глобальном окружении создается список DATA, содержащий выгруженные контенты и CSS-файл с выгруженными данными. Если NetworkData=true, то функция возвращает две дополнительные переменные:

Алгоритм использования пакета RCrawling состоит из следующих этапов:

1. Сканирование по сайту и выгрузка страниц сайта на жесткий диск компьютера.

2. Поиск информации на сайте по ключевым словам, XPath локаторам и CSS селекторам.

3. Выгрузка контента с помощью функции ContentScraper.

4. Сравнение методов выбор оптимального (из 1-3).

5. Поиск информации в выгруженных данных с использованием регулярных выражений.

6. Отфильтровывание ненужных страниц.

7. Написание сценария автоматического выполнения пп.1-6.

8. Аналитический анализ и визуальное представление полученных результатов поиска.

Функция `ContentScraper()` извлекает один и больше элементов с веб-страницы с помощью шаблонов XPath и CSS; извлекает данные с множества веб-страниц/URL; выгружает данные веб-страниц, используя веб-драйвер для javascript; выгружает данные защищенных паролем веб-страниц; выгружает данные со страниц и преобразует их во фрейм данных.

Для поиска по ключевым словам предназначены следующие функции:

Функции `grep()`, `grepl()`, `regexpr()`, `gregexpr()` and `regexec()` ищет согласования для агрумента `pattern` с каждым элементом после `pattern` символьного вектора: они различаются форматом и суммой подробностей в результате.

Функции `sub()` and `gsub()` выполняют замену первого и всех совпадений (или согласований) соответственно.

Аргументы функций:

`pattern` – символьная строка, содержащая регулярное выражение. Автоматически преобразуется в строку, если это возможно (приведение типов). Если `pattern` двумерный и более вектор, то используется первый элемент вектора с предупреждением. Допускаются отсутствующие значения (NA), кроме функции `regexpr()` и `gregexpr()`;

`x, text` – символьный вектор, в котором ищется согласование, или объект, который может быть преобразован в символьный вектор. Поддерживаются длинные векторы;

`ignore.case` – логический, если FALSE, то согласования (совпадения) чувствительны к регистру букв, иначе – нет;

`perl` – логический, должны ли использоваться регулярные выражения, совместимые с `Perl`, по умолчанию FALSE (т.е. используется R);

`value` – логический, если FALSE, то возвращается вектор, содержащий целочисленные индексы согласований, найденные функцией `grep()`. Если TRUE, то возвращаются найденные совпадения.

`fixed` – логический, если TRUE, то `pattern` – это строка, как она есть (фиксированный шаблон, над которым не делается никаких преобразований);

`useBytes` – логический, если TRUE, то согласование делается побайтово, а не по символам;

`invert` – логический, если TRUE, то возвращаются индексы несогласованных элементов;

`replacement` – замена для согласований в функциях `sub()` и `gsub()`.

Стратегии скрепинга представлены ниже.

Если сайт большой (больше 50000 страниц) и извлекается специфическая информация:

1. Попытаться ограничить диапазон кроулинга путем обнаружения, не локализован ли желаемый контент в определенном разделе сайта.

2. При поиске нужной информации и вместо указания домашней страницы использовать для кроулинга (поиска) другой стартовый адрес.

3. Поискать нужный контент на сайте, используя поисковую систему Google. Строка запроса: `Keywords+ site:https://targetwebsite.org`. А затем использовать функцию ContentScraper.

4. Перед выгрузкой всего сайта, используя функцию Rcrawler, попытайтесь выгрузить только одну страницу, используя функцию ContentScraper, чтобы удостоверится, что ваши шаблоны XPath и CSS корректны.

Таким образом, можно выделить *ряд преимуществ* пакета Rcrawler.

Rcrawler позволяет:

- загружать все страницы веб-сайта;
- загружать все собранные файлы в окружение R (в память);
- извлекать все структурированные данные изо всех веб-страниц (заголовки, посты, описания, продукты и т.д.);
- выгружать целевой контент, используя параметры поиска, имея нужные ключевые слова Rcrawler может перемещаться по всем ссылкам веб-сайта и извлекать из него только нужные веб-страницы;
- на больших сайтах предоставляются некоторые полезные параметры для ограничения процесса кроулинга, такие как 1) фильтрация URL с помощью ключевых слов или шаблонов согласования (регулярные выражения), 2) заданием, сколько уровней должно быть просканировано, начиная с заданной точки (по умолчанию 10 уровней), 3) игнорированием некоторых параметров URL в процессе кроулинга;
- при манинге веб-структуры дополнительно предоставляется возможность представления веб-сайта в виде сети путем отображения всех его внешних и внутренних ссылок;
- выгружать данные из списка URL и исключать из выгружаемых данных внутренние элементы (узлы).

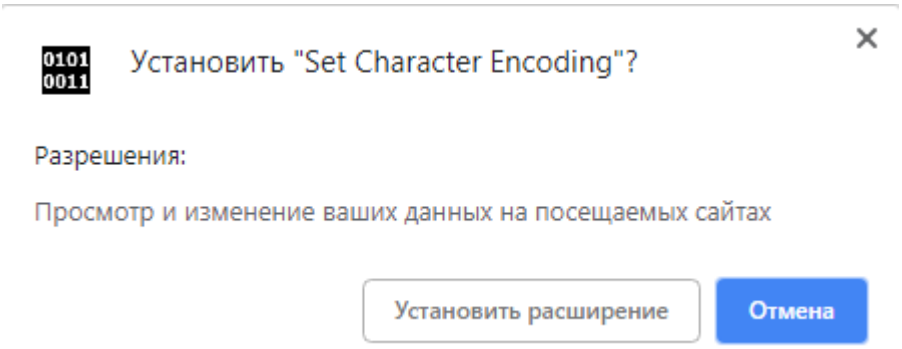
3.2 Настройка среды разработки

Google Chrome не имеет опции изменения кодировки страниц (предполагается, что это делается автоматически). Однако, когда RCrawler скачивает страницы и сохраняет их, то при открытии этих страниц в Google Chrome они могут отображаться в неправильной кодировке.

На сайте Google Play Market имеется расширение Set Character Encoding (https://chrome.google.com/webstore/detail/set-character-encoding/bpojelgakakmcfmjfilgdlmhefphglae), которое позволяет изменять кодировку страницы из контекстного меню (щелчок правой кнопкой мыши на странице).

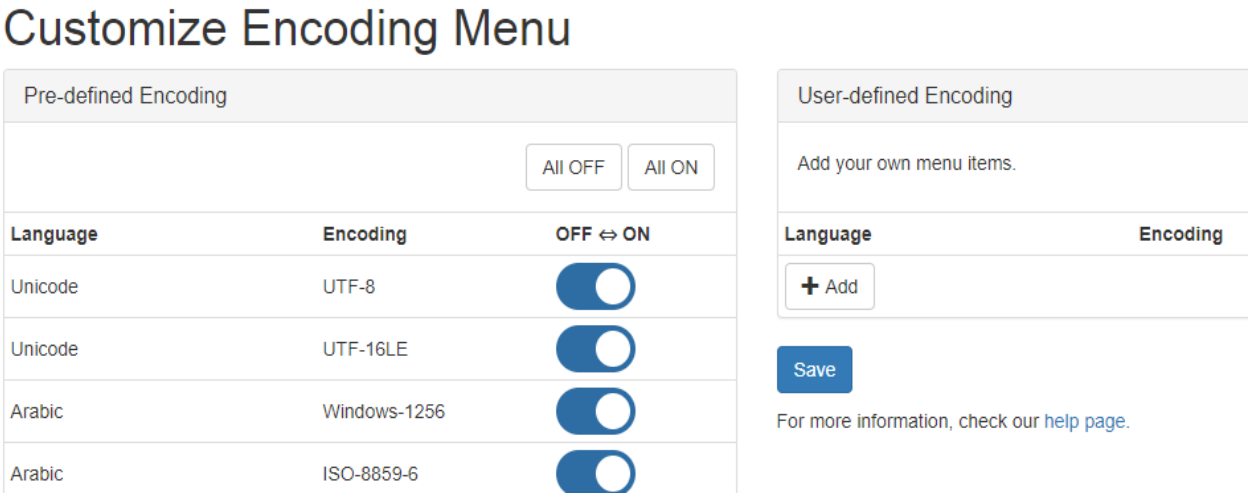
Установка и настройка расширения состоит из следующих этапов:

1. На странице с указанным адресом щелкнуть кнопку Установить. Появится диалоговое окно, в котором щелкнуть кнопку «Установить расширение».

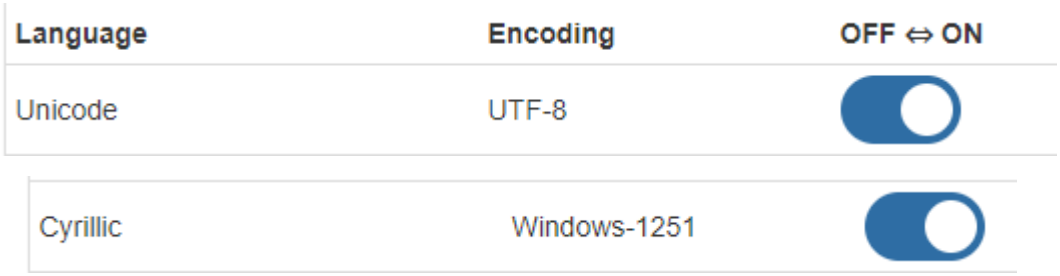


2. После установки появится диалоговое окно Расширение установлено, которое нужно закрыть щелчком на крестике.

3. В меню Google Chrome появится значок 01. Щелчком на этом значке запускаем команду Параметры. Откроется страница Customize Encoding Menu.



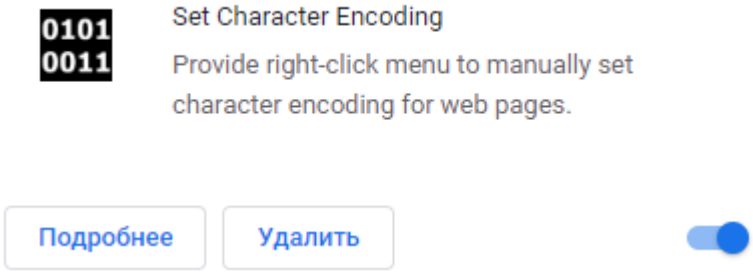
В этом окне щелкнуть кнопку AllOFF и выбрать:



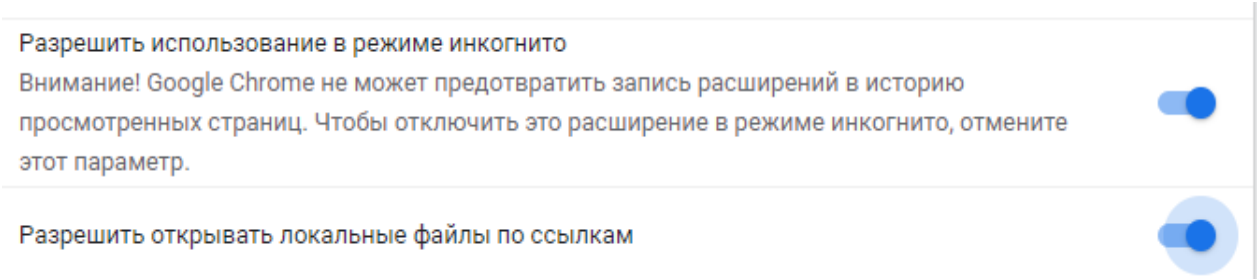
И нажать кнопку Save.

4. Закрыть страницу Customize Encoding Menu. В меню Google Chrome щелкнуть кнопку ⋮ и выбрать опцию Дополнительные инструменты / Расширения.

5. В окне расширения щелкнуть кнопку Подробнее.



И установить необходимые флажки:



6. Закрыть окно расширения.

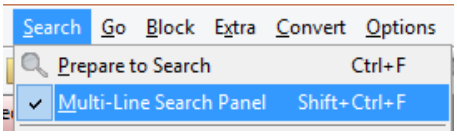
Установка расширения завершена.

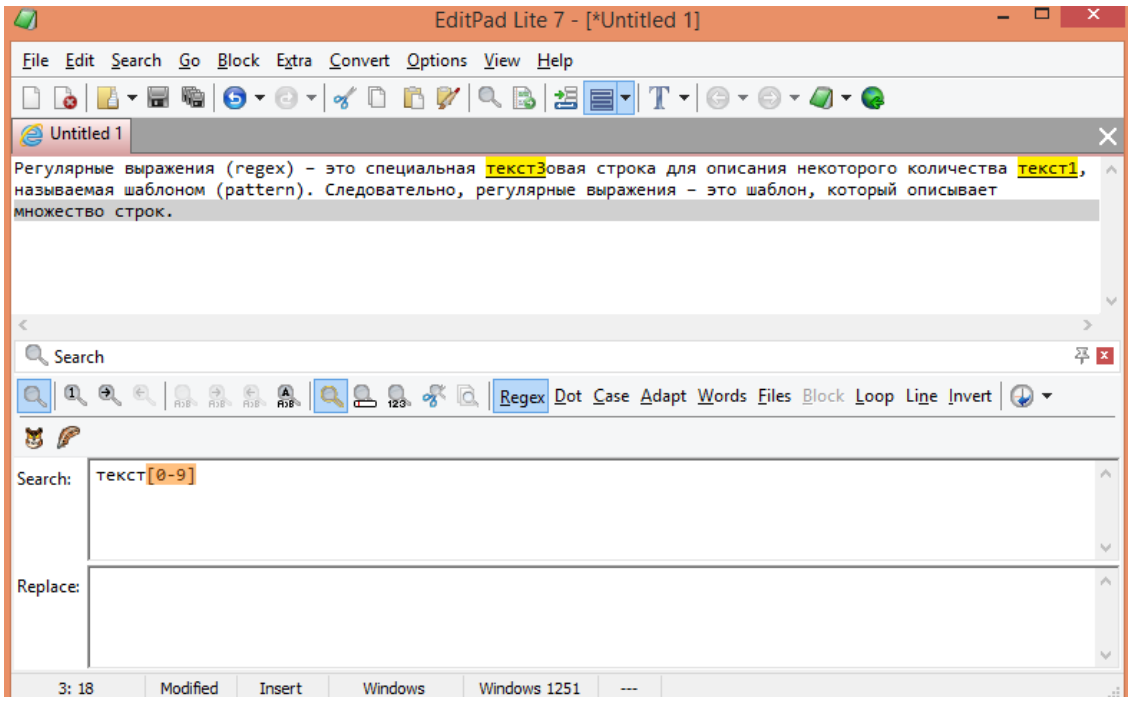
Для отладки регулярных выражений удобно использовать EditPad Lite.

1. Зайти на сайт EditPad Lite и скачать программу.

2. Для работы в программе нудно выполнить ряд следующих действий:

- добавить панель:





- вставить нужный текст;
- в окне Search ввести шаблон поиска регулярного выражения и нажать кнопку Regex, а затем 🔍;
- текст, который будет найден с помощью указанного шаблона, будет выделен желтым цветом;
- при нажатой кнопке Case поиск будет чувствителен к регистру букв.

3.3 Разработка сценария для автоматизированного поиска информации на веб-сайтах

Автоматизация поиска необходимой информации на web-сайтах осуществлялась на языке программирования R с использованием пакета Rcrawler на примере экологических данных. Диаграмма работы сценария для автоматизированного поиска представлена на рисунке 3.1.

На диаграмме видно, что пользователь только подключает необходимый пакет в программе, все действия по поиску информации выполняются автоматически с помощью возможностей пакета Rcrawler.

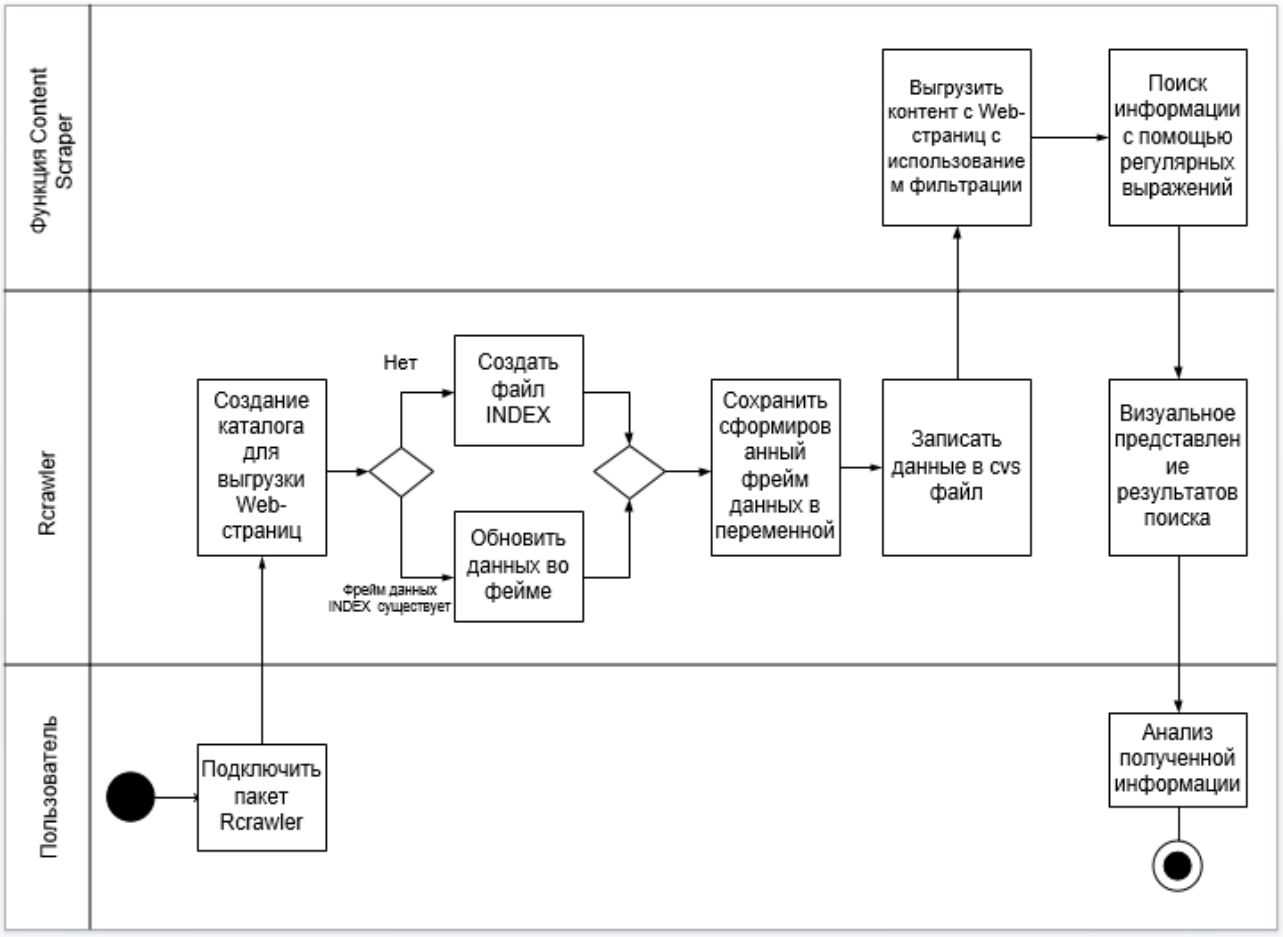


Рисунок 3.1 – Диаграмма работы сценария для автоматизированного поиска

Созданный сценарий работает следующим образом:

В строке `library(Rcrawler)` подключается пакет Rcrawler.

Функция `Rcrawler(Website = "http://greenologia.ru/eko-problemy/goroda/tolyatti.html", no_cores = 4, no_conn = 4)` сканирует по сайту greenologia.ru. Найденные страницы сохраняются в заданной папке на компьютере (рис. 3.2). Если нужная папка не была ранее создана, то она создается по умолчанию.

Имя	Дата изменения	Тип	Размер
1 .html	08.01.2019 18:24	Chrome HTML Document	95 КБ
2 .html	08.01.2019 18:24	Chrome HTML Document	72 КБ
3 .html	08.01.2019 18:24	Chrome HTML Document	67 КБ
4 .html	08.01.2019 18:24	Chrome HTML Document	67 КБ
5 .html	08.01.2019 18:27	Chrome HTML Document	342 КБ

Рисунок 3.2 – Содержимое папки с выгруженными web-страницами (фрагмент)

Кроме того, в окружении создается фрейм данных INDEX, содержащий информацию о скачанных файлах (Рис. 3.3).

	Id	Url	Stats	Level	OUT	IN	Http Resp	Content Type	Encoding
1	1	http://greenologia.ru/eko-problemy/goroda/tolyatti....	finished	0	54	1	200	text/html	UTF-8
2	2	http://greenologia.ru/o-proekte	finished	1	50	1	200	text/html	UTF-8
3	3	http://greenologia.ru/kontakty	finished	1	50	1	200	text/html	UTF-8
4	4	http://greenologia.ru/reklama	finished	1	50	1	200	text/html	UTF-8
5	5	http://greenologia.ru/sitemap	finished	1	903	1	200	text/html	UTF-8

...

952	952	http://greenologia.ru/eko-zhizn/sistemy/vodosnabz...	finished	4	72	1	200	text/html	UTF-8
953	953	http://greenologia.ru/eko-zhizn/xozyajstvo/fermerst...	finished	4	79	1	200	text/html	UTF-8
954	954	http://greenologia.ru/eko-zhizn/xozyajstvo/fermerst...	finished	4	69	1	200	text/html	UTF-8
955	955	http://greenologia.ru/eko-zhizn/xozyajstvo/fermerst...	finished	4	67	1	200	text/html	UTF-8
956	956	http://greenologia.ru/eko-zhizn/xozyajstvo/fermerst...	finished	4	77	1	200	text/html	UTF-8

Рисунок 3.3 - Фрагмент фрейма данных INDEX (первые и последние пять строк)

Сформированный фрейм данных сохраняется в переменной `index.greenologia` и записывается файл `index.greenologia.csv` с помощью следующих команд:

```
index.greenologia <- INDEX
write.table(index.greenologia, file="index_greenologia.csv",
row.names = FALSE, sep = ";")
```

На рисунке 3.4 показаны первые 10 строк файла `index.greenologia.csv`.

	A	B	C	D	E	F	G	H	I
1	Id	Url	Stats	Level	OUT	IN	Http Resp	Content Type	Encoding
2	1	http://greenologia.ru/eko-problemy/goroda/tolyatti.html	finished	0	54	1	200	text/html	UTF-8
3	2	http://greenologia.ru/o-proekte	finished	1	50	1	200	text/html	UTF-8
4	3	http://greenologia.ru/kontakty	finished	1	50	1	200	text/html	UTF-8
5	4	http://greenologia.ru/reklama	finished	1	50	1	200	text/html	UTF-8
6	5	http://greenologia.ru/sitemap	finished	1	903	1	200	text/html	UTF-8
7	6	http://greenologia.ru/eko-problemy	finished	1	69	1	200	text/html	UTF-8
8	7	http://greenologia.ru/eko-problemy/biosfera	finished	1	55	1	200	text/html	UTF-8
9	8	http://greenologia.ru/eko-problemy/biosfera/bolota	finished	1	57	1	200	text/html	UTF-8
10	9	http://greenologia.ru/eko-problemy/goroda	finished	1	75	1	200	text/html	UTF-8
11	10	http://greenologia.ru/eko-problemy/gidrosfera	finished	1	65	1	200	text/html	UTF-8

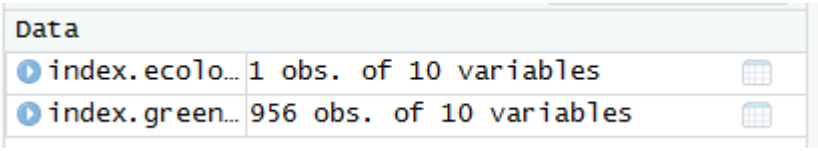
Рисунок 3.4 - Фрагмент содержимого файла `index.greenologia.csv`

Описание полей полей фрейма данных INDEX: Id – идентификатор сохраненной страницы, Url – ее адрес, Stats – статус страницы, Level – уровень иерархии страниц на сайте (0-домашняя, страницы 1-го уровня, 2-го и т.д.),

OUT – число выходов со страницы, IN – число входов на страницу, Http Resp – код ошибки (например, 200 – страница успешно загружена), Content Type – тип контента (text/html – это html-страница), Encoding – кодировка страницы.

В следующих командах найденная информация записывается в дополнительные переменные.

```
index.greenologia <- INDEX
write.table(index.greenologia, file="index_greenologia.csv",
            row.names = FALSE, sep = ";")

index.ecologytogliatti <- INDEX
rm(DATA,INDEX,excontent2)
```



Функция `ContentScraper()` сканирует страницы web-сайта, но дополнительно осуществляет поиск названия страниц и извлекает информацию внутри тегов div, а также сохраняет найденную информацию во фреймах данных. В данных командах используются Xpath-локаторы.

```
DATA<-ContentScraper(Url
="http://glofile.com/index.php/2017/06/08/athletisme-m-a-rome/",
         XpathPatterns=c("//head/title","//*/article"),
         PatternsName=c("title", "article"))
RobotParser("http://www.glofile.com","AgentX")
```

Команда `RobotParser("http://www.glofile.com","AgentX")` загружает с сайта и разбирает содержимое файла `Robot.txt`, в котором записаны правила доступа клиента к серверу (разрешен, запрещен, заблокирован).

Следующая команда выводит список всего кроулинга в каталоге Rcrawler. Функция загружает в окружение содержимое файлов одной из папкой, возвращенных `ListProjects()`.

```
> ListProjects()
[1] "ecologytogliatti.narod.ru-051358" "ecologytogliatti.narod.ru-131253"
[3] "greenologia.ru-081724"            "greenologia.ru-111647"
[5] "greenologia.ru-131250"
```

По мере выгрузки страниц сайта в глобальном окружении R обновляется фрейм данных INDEX, который содержит адрес ссылки, тип контента,

состояние http, число выходных и входных ссылок, тип кодирования, уровень. Затем RCrawler создает каталог, содержащий выгруженные веб-страницы.

Следующий фрагмент программного кода осуществляет поиск текстовой информации по ключевым словам (рис. 3.5). В результате работы сценария в созданном каталоге сохраняются только те web-страницы, которые содержат указанные ключевые слова.

```
ListProjects()
MyData.greenologia <- LoadHTMLFiles(ListProjects()[2], type = "vector", max=50)
str(MyData.greenologia)
MyData.greenologia[1]
pattern <- "сток|материалы"
npgs <- dim(index.greenologia)[1]
for (i in 1:npgs){
  if (length(grep(pattern, MyData.greenologia[i]))>0){
    cat("\ni =", i, "\n")
    m <- gregexpr(pattern,MyData.greenologia[i])
    print(m[[1]])
    print(regmatches(MyData.greenologia[i],m)[[1]])
  }
}
```

Рисунок 3.5 - Фрагмент программного кода для поиска по ключевым словам

Во время поиска текстовой информации на русскоязычном сайте столкнулись с проблемой, что русские ключевые слова не включаются в результаты поиска. То есть Rcrawler не способен найти на русскоязычном сайте страницы по ключевым словам и выгрузить только их из-за проблем кодировки страниц. Причем попытка решения проблемы путем изменения кодировок не дал положительных результатов, поэтому было принято решение анализировать страницы web-сайта с помощью xPath-локаторов и CSS-селекторов.

Подход, предложенный для решения данной проблемы, заключается в том, что в папку на компьютере пользователя выгружаются все страницы сайта, затем они загружаются в глобальную память R. Далее среди загруженных в память R web-страниц осуществляется поиск как по ключевым словам, так и с помощью регулярных выражений.

В целях повышения точности результатов контент-майнинга, только значимые данные должны быть извлечены. Некоторые нерелевантные данные,

такие как навигационные панели, баннеры и реклама, должны быть исключены из результатов поиска. Извлечение данных – это проблема извлечения целевой информации с web-страниц для создания структурированных данных, которые готовы для постобработки. Сканирование в Интернете и извлечение данных может быть реализовано либо как две отдельные последовательные задачи (сканер выбирает все веб-страниц в локальный репозиторий, затем процесс извлечения применяется ко всей коллекции), или как одновременные задачи (в то время как сканер выбирает страницы, к которым применяется процесс извлечения страница индивидуально).

Актуальной задачей является необходимость создания улучшенного веб-сценария, который может сканировать и очищать содержимое веб-страницы (статьи, заголовки и метаданные) в автоматическом режиме для создания структурированного набора данных.

На рисунке 3.6 показан алгоритм поиска на страницах русскоязычного web-сайта заголовков первого уровня.

Пользователю необходимо задать адрес URL для поиска, а дальнейший поиск происходит в автоматическом режиме с использованием предложенного сценария. Все страницы указанного сайта загружаются в каталог на компьютере пользователя, дальнейший поиск необходимой информации производится по регулярным выражениям среди скаченных web-страниц.

Сценарий поиска заголовков первого уровня представлен на рисунке 3.7.

```
47 url <- "http://greenologia.ru/eko-problemy/goroda/tolyatti.html"
48 Rcrawler(Website = url, no_cores = 4, no_conn = 4, ExtractXpathPat = c("//h1", "//h2", "//h3"))
49 lw <- LoadHTMLFiles(ListProjects()[6], type = "vector")
50 length(lw) #число загруженных страниц
51 for (i in 1:length(lw)){
52   pattern1 <- "<h1"
53   gr1 <- gregexpr(pattern1,lw[i])
54   pattern2 <- "</h1>"
55   gr2 <- gregexpr(pattern2,lw[i])
56   h1 <- substr(lw[i],gr1[[1]][1], gr2[[1]][1]+4)
57   h1 <- substr(h1,1, nchar(h1)-5)
58   h1 <- sub(pattern = '\\"', "", h1)
59   h1 <- sub(pattern = '\\"', "", h1)
60   h1 <- substr(h1, gregexpr(">", h1)[[1]][1]+1, nchar(h1))
61   cat("i =", i, ":\n")
62   cat("h1 =", h1, "\n")
63 }
```

Рисунок 3.7 – Сценарий поиска заголовков первого уровня

Аналогичным образом, следуя этапам предложенного алгоритма, можно осуществлять поиск заголовков других уровней, полей форм, ссылок и других элементов объектной модели документа и CSS-стилей.

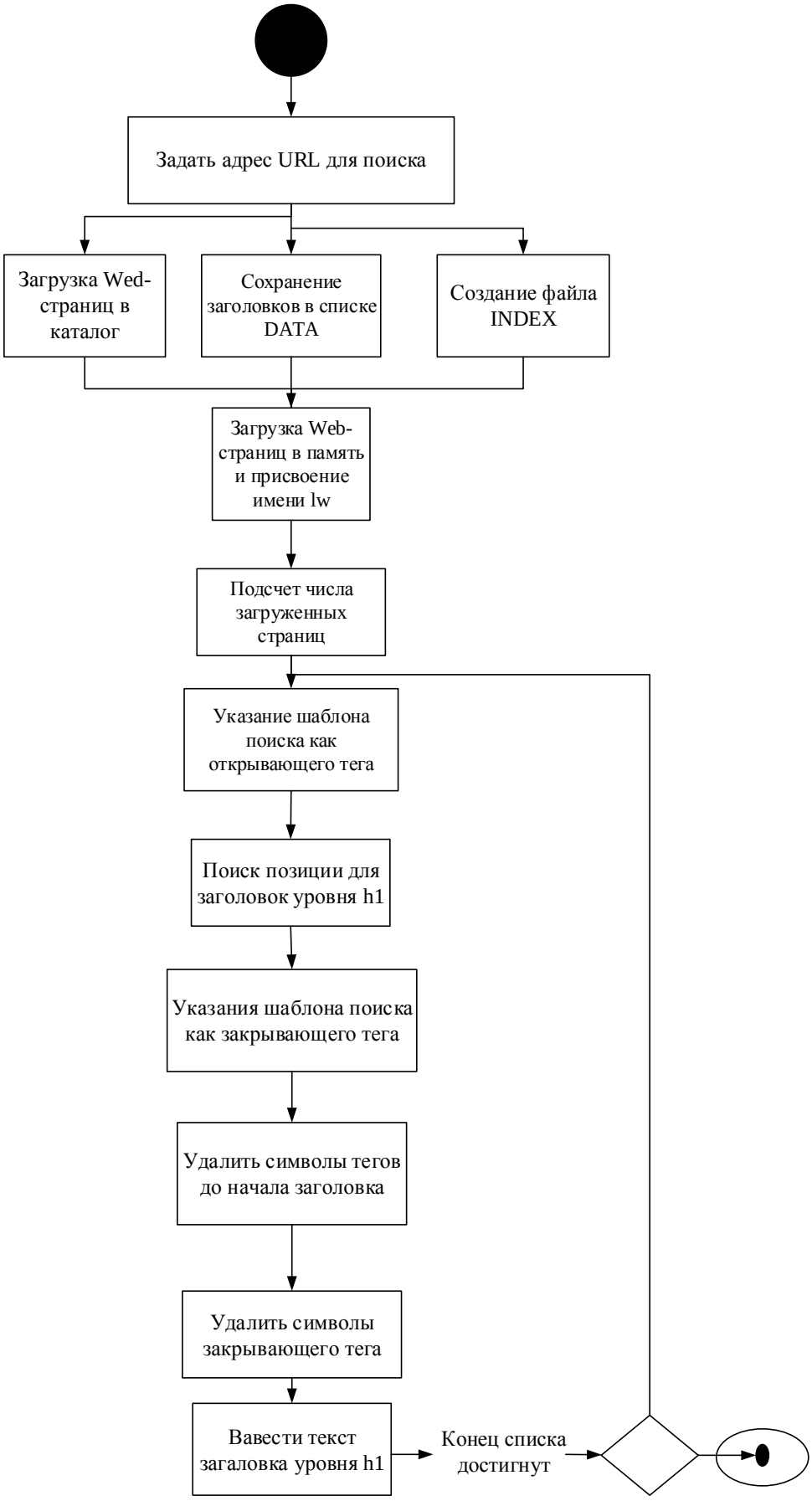


Рисунок 3.6 – Блок-схема алгоритма поиска заголовков первого уровня

В результате работы сценария получаем список заголовков первого уровня с указанием соответствующей web-страницы (рис. 3.8).

```
i = 50 :
h1 = Чугунный лом – вид металолома
i = 51 :
h1 = Пользу или вред приносит энергосберегающая лампа и как поступить, если она разбилась
i = 52 :
h1 = Применение ртути и опасности которые вас подстерегают
i = 53 :
h1 = Симптомы отравления ртутью, оказание первой помощи
i = 54 :
h1 = Безотходное производство в деревоперерабатывающей промышленности
i = 55 :
h1 = Обзор трех бытовых котлов (печей) работающих на опилках
```

Рисунок 3.8 – Результат работы сценария

Заголовки разных уровней позволяют дать представление о содержимом как всей страницы в целом, так и ее подразделах. Также названия страниц и подразделов позволяют сформировать представление, может ли данная веб-страница содержать полезную информацию или нет.

На рисунке 3.9 представлен фрагмент содержимого переменной DATA в результате работы представленного сценария.

Name	Type	Value
DATA	list [166]	List of length 166
[[1]]	list [3]	List of length 3
[[1]]	character [1]	'Напряжённая экологическая ситуация в Тольятти'
[[2]]	character [1]	'Экологическая ситуация в городе'
[[3]]	character [1]	'Нет комментариев'
[[2]]	list [3]	List of length 3

Рисунок 3.9 – Фрагмент содержимого переменной DATA

Перед проведением анализа структуры сайта необходимо ответить на ряд вопросов:

1. Какая страница является общей точкой входа для пользователей, т.е. наличие домашней страницы?

2. В каком порядке просматриваются страницы?

3. Какие другие веб-сайты направляют пользователей на исследуемый сайт?

4. С каких сайтов поступает наибольшее и наименьшее число пользователей?

5. На какие внешние сайты направляет пользователей исследуемый сайт?

Для построения сетевого графа сайта в R необходимо выполнить ряд шагов:

Добавить библиотеку `library(igraph)`.

Если в функции Rcrawler() задать параметры NetworkData = TRUE, то краулер будет создавать две дополнительные глобальные переменные, управляющие вершинами и дугами графа:

1. NetwIndex – вектор, отображающий все гиперссылки (узлы) на уникальный целый идентификатор `ID` (позиция элемента в векторе).

2. NetwEdges – фрейм данных, представляющий дуги графа со столбцами: From, To, Weight (уровень вложения гиперссылки) и Type (1 – внутренние ссылки, 2 – внешние ссылки).

```
library(igraph)
Rcrawler(Website = "http://greenologia.ru/eko-
problemy/goroda/tolyatti.html",
         no_cores = 4, no_conn = 4, NetworkData = TRUE,
NetwExtLinks =TRUE,
         statslinks = TRUE, MaxDepth = 1)
network<-graph.data.frame(NetwEdges[1:50,], directed=T)
plot(network)

network<-graph.data.frame(NetwEdges[200:240,], directed=T) #в
квадратных скобках указываются номера дуг
plot(network)
```

Одним из универсально полезных инструментов визуализации в интеллектуальном анализе текста является простая сетевая структура сайта. Этот вид инструмента визуализации также может быть легко применен, чтобы позволить пользователю проанализировать ссылки и перейти к следующим документам (web-страницам), содержащим необходимую информацию.

На рисунке 3.10 показана структура веб-страниц, согласно представленному сценарию страницы с 200 по 240. Представленный сценарий позволяет изучить структуру веб-сайта, создав сетевое представление внутренних и внешних гиперссылок (узлов и ребер) сайта.

Сетевой граф позволяет производить анализ ссылок как на другие страницы, так и внутри страницы.

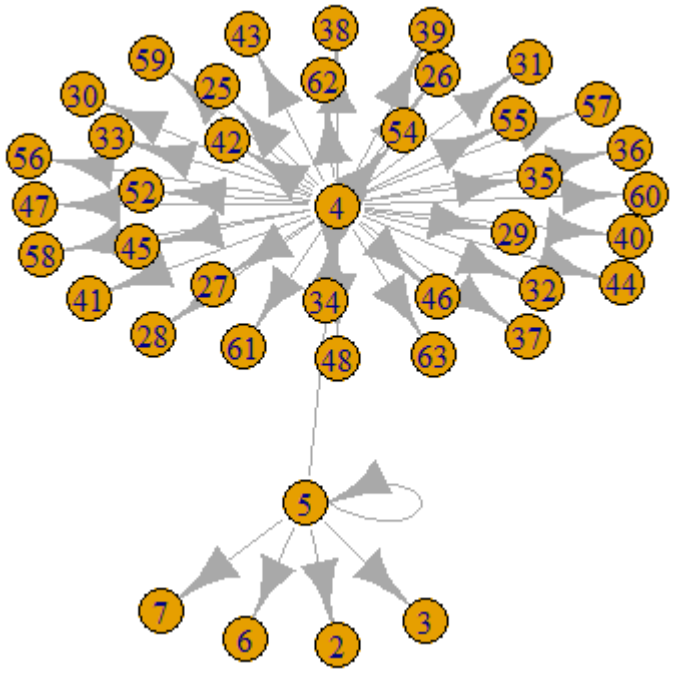


Рисунок 3.10 – Сетевое представление структуры сайта

Схематичное представление структуры сайта позволяет увидеть количество вложенных уровней ссылок, а также количество переходов с той или иной страницы на другие страницы веб-сайта.

Исходя из представленной структуры можно увидеть, что на странице 4 имеется множество одноуровневых ссылок на другие страницы сайта, и только со страницы 5 имеются ссылки следующего уровня вложения.

Следующий фрагмент кода формирует сетевой граф связей страниц с 238 по 245.

```
network<-graph.data.frame(NetwEdges[238:245,], directed=T) #в
квадратных скобках указываются номера дуг
plot(network)
```

Представленный фрагмент сценария может быть отправной точкой в составлении и визуализации более сложных сетевых представлений структуры сайта.

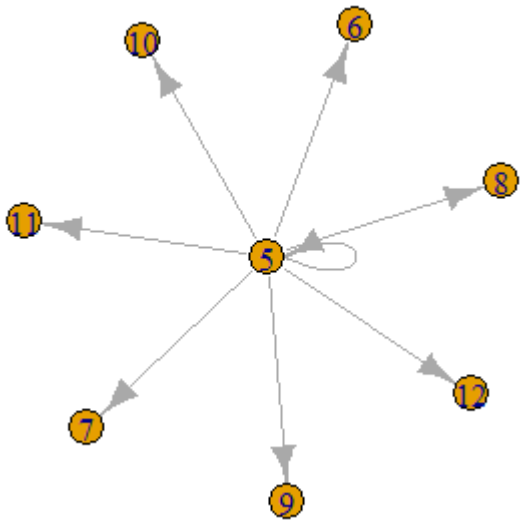


Рисунок 3.11 – Сетевой граф связей страниц с 238 по 245

В заключение можно сказать, что пакет RCrawler является лучшим решением для извлечения неструктурированных данных с web-сайтов. Он прост в использовании, позволяет сканировать и загружать страницы с высокой скоростью и эффективностью, представляет собой полное решение для сбора данных с возможностью настроек параметров поиска.

3.4 Анализ эффективности автоматизированного поиска и анализа неструктурированной информации

Информационно-поисковая система (ИПС) имеет ряд таких возможностей, как поиск полных текстов документов, поиск вторичных документов (например, рефератов) или поиск названий и адресов документов, т.е. библиографических описаний. Процесс, в результате которого, пользователю выдаются полные тексты документов можно назвать системой поиска документов. В то же время поисковая система, которая на запрос предоставляет только описание документов, может быть системой поиска ссылок. Зачастую поиск необходимой информации на веб-ресурсах включает несколько стадий. На первом этапе результат поиска может иметь форму ссылок, среди которых пользователь производит выбор необходимых в настоящий период для решения текущей задачи. На следующем этапе

пользователь может осуществить запрос уже полных текстов отобранных источников.

Пользователь поисковой системы, прежде всего, заинтересован в получении документов, которые способствуют удовлетворению его потребности в необходимой информации в конкретный период времени. В некоторых случаях для решения задачи будет достаточно единственно документа, в других – нескольких основных документов, а в определенных ситуациях пользователю необходимо получить как можно больше источников по исследуемой предметной области.

Основное требование, предъявляемое к автоматизированной поисковой системе – система должна отыскивать документы, релевантные различным предметным запросам. Способность системы отыскивать релевантные документы называется полнотой (recall).

Под релевантностью поиска понимают соответствие документа ожиданиям пользователя, то есть степень удовлетворения пользователя показанными в ответ на его запрос поисковыми результатами.

Наиболее важным и значимым требованием пользователя к поисковой системе является полнота, так как единственная цель его обращения к системе состоит в получении одного или более документов, полезных с точки зрения его информационной потребности.

Показатель полноты ИПС можно выразить количественно с помощью *коэффициента полноты*, определяемого формулой 100 C/R, где R – общее число документов в системе, о которых известно, что они релевантны определенному запросу, а C – число этих релевантных документов, найденных в процессе поиска по данному запросу в указателе к фонду. Пусть мы определили, что по некоторому запросу, введенному в систему, имеется 10 релевантных документов, которые были индексированы в нашем поисковом массиве. Если при проведении поиска мы в состоянии отыскать восемь из них, мы говорим, что коэффициент полноты для данного конкретного поиска равен (8/10) * 100, или 80%.

100 C/R=(8/10) * 100=80%

Учитывая такое обстоятельство, что 100% – ной полноты поиска всегда можно добиться путем просмотра всего фонда документов, получается, что:

1. численное значение полноты еще не является мерой эффективности ИПС;

2. назначение указателя состоит в том, чтобы отсеять как можно больше нежелательных документов, потеряв при этом как можно меньше полезных документов.

При рассматриваемом подходе указатель по существу является фильтром, цель которого состоит в сокращении количества найденных документов (или их заменителей), которые приходится просматривать во время поиска, при сохранении приемлемого значения полноты выдачи. Так как коэффициент полноты определяется как мера способности фильтра пропускать желательные документы, необходимо располагать еще неким дополняющим его коэффициентом, который бы являлся мерой способности фильтра задерживать ненужные документы. Сирил Клевердон предположил в качестве подходящей меры коэффициент точности 100 R/L, где R – по-прежнему число отысканных при поиске релевантных документов, а L – общее число документов, выданных при этом поиске. Например, если в массиве имеется 10 релевантных данному запросу документов, из которых при поиске в указателе пользователь может найти только 8 (полнота 80%). Если при поиске этих восьми желательных документов выдается 100 документов (8 желательных, 92 ненужных), тогда говорят, что коэффициент точности для данного поиска составляет 8/100, или 8%.

100 R/L=(8/100)*100=8%

В настоящее время существует огромное число сайтов определенной тематической направленности, например, экология, космос, образовательные ресурсы, логистика. Сайты могут иметь большое количество страниц и в этом объеме страниц нужно найти информацию по какой-то тематике, рубрике.

Варианты поиска. Вручную просматривать все страницы; по карте сайта, т.е. по заголовкам страниц (релевантные страницы); скачивать отдельные страницы и искать информацию в них по ключевым словам с помощью регулярных выражений и т.д.

Все процедуры «ручного» поиска требуют большого ручного труда и больших затрат времени на поиск нужной информации. Для преодоления этих затруднений необходим автоматизированный поиск и анализ информации.

Эффективность информационного поиска (по производительности) – это время автоматизированного поиска, деленное на время ручного поиска.

Таблица 3.2 - Эффективность информационного поиска

Вид поиска / Режим поиска	Время просмотра 1000 страниц (мин)	По карте сайта (т.е. по названию страниц) (мин)	Поиск XPath-локатор и CSS селектор (мин)	По ключевым словам и с помощью регулярных выражений (мин)
Ручной	5000	180	120	120
Автоматический	0,5	0,5	0,5	0,5

Основные функции пакета по кроулингу и выгрузки контента с веб-страниц сайта были использованы на примере сайта *greenologia.ru* по экологии Тольятти. Сайт содержит около тысячи html-страниц, поэтому ручной поиск и анализ его трудоемкий и затруднительный. Время автоматизированного поиска определяется лишь временем скачивания страниц сайта.

Недостаток ручного поиска – выгружается только часть страниц сайта, найденная по карте сайта (т.е. получается неполный поиск). С помощью сценария осуществляется полный поиск, т.к. выгружаются все страницы и ведется поиск по всем страницам. Отсюда можно сделать вывод, что автоматизированный поиск эффективен в несколько раз в сравнении с ручным.

Для примера проведем сравнительный анализ нескольких способов информационного поиска с сети Интернет: по ключевым словам , по карте сайта, по xPath-локаторам и CSS-селекторам и по ключевым словам.

Ключевые слова: “сток | материалы”

L=946 – всего страниц (документов);

R=16 – число релевантных страниц (документов);

C=3 – релевантные страницы, найденные по карте сайта;

C_1=7 - релевантные страницы, найденные по xPath и CSS;

C=R - релевантные страницы, найденные по ключевым словам.

Поиск по карте сайта:

Коэффициент полноты C/R*100%=(3/16) * 100%=18.8%;

Коэффициент точности 100 R/L=(16/946)*100%=1.7%.

Поиск по xPath и CSS:

Коэффициент полноты C_1/R*100%=(7/16) * 100%=43.8%;

Коэффициент точности 100 R/L=(16/946)*100%=1.7%.

Поиск по ключевым словам:

Коэффициент полноты R/R*100%=(16/16) * 100%=100%;

Коэффициент точности R/L*100%=(16/946)*100%=1.7%.

Поиск информации производился с использованием предложенного сценария и показал, что использование автоматизированного способа повышает коэффициент полноты, то есть позволяет отыскать больше релевантных документов. Возможность зайти в структуру сайта и в элементы форматирования документа повышает количество релевантных документов по результатам поиска, в сравнении с поиском по заголовкам страниц, т.е. по карте сайта.

Выводы по третьей главе

В данной главе представлен алгоритм работы сценария, а также реализован на языке R с применением пакета расширения RCrowler сценарий для автматического поиска неструктурированной информации в сети Интернет. Показано преимущество автоматизированного поиска в сравнении с ручным.

ЗАКЛЮЧЕНИЕ

В рамках диссертационной работы получены следующие основные результаты:

1. На основе исследования научной литературы определены требования к технологии автоматизированного поиска информации на веб-ресурсах.

2. Исследованы возможности программной среды R совместно с пакетом расширения RCrawler для разработки сценария автоматического поиска информации в сети Интернет.

3. Разработан алгоритм работы сценария автоматического поиска в среде R.

4. Разработан сценарий автоматического поиска информации в сети Интернет.

5. Выработаны практические рекомендации и этапы работы с разработанным сценарием поиска.

6. Проведен сравнительный анализ ручного и автоматизированного поиска, в результате которого показана эффективность автоматизированного поиска информации.

Можно сказать, что разработанный экспресс-вариант сценария автоматического text-mining с целью извлечения полезной информации об экологической ситуации в г. Тольятти значительно повышает эффективность поиска нужной информации в глобальной сети.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Аверченков В.И. Мониторинг и системный анализ информации в сети Интернет: монография / В.И. Аверченков, С.М. Рощин. – Брянск: БГТУ, 2012. – 160 с.

2. Аверченков В.И., Казаков Ю М., Рощин С.М., Шкаберин В.А. Мониторинг и анализ информации в Интернет // Известие ВГТУ Серия «Актуальные проблемы ВТ и информатики в технических системах»: № 3(35). - Волгоград.- 2007, с. 29-35.

3. Аверченков В.И., Казаков Ю.М. Аспекты создания интеллектуальной поисковой системы для формирования предметно-ориентированных информационных ресурсов // Известия ОрелГТУ Серия "Фундаментальные и прикладные проблемы техники и технологии; информационные системы и технологии". №1-2/269(544) Орел. – 2008, с.4-10.

4. Ашихмина Т. Л. Экологический мониторинг / Т. Л. Ашихмина. – Изд. 4-е. – М.: Академический Проект; Альма Матер, 2008. – 416с.

5. Бах Э. Неформальные лекции по формальной семантике: Пер. с англ. / Под. ред. О. А. Митрофановой, О. В. Митрениной; Предисл. Б. Парти. - М.: Книжный дом «ЛИБРОКОМ», 2010.-224 с.

6. Гаврилова Т.А. Базы знаний интеллектуальных систем /Т. А. Гаврилова, В. Ф. Хорошевский. - СПб.: Питер, 2000 - 384 с.

7. Ермаков Д.Г., Родина У.А., Хандыго В.Г. Технология информационного мониторинга // Научное сообщество студентов XXI столетия. Технические науки: сб. ст. по мат. IV междунар. студ. науч.-практ. конф. № 4. 2012. URL: http://sibac.info/archive/technic/4.pdf

8. Кузьмин М.Н. Мониторинг как составная часть информационного обеспечения процесса управления. Мордовский государственный университет им. Н.П. Огарева. URL: http://sisupr.mrsu.ru/2009-1/pdf/17_Kyzmin.pdf

9. Малышев М.Л. Мониторинг социально-трудовой сферы: опыт социолого-статистического анализа. М.: Изд-во "Союз", 2005.

10. Маннинг К. Введение в информационный поиск / К. Маннинг, П. Рагхаван, Х. Шютце. М.: Вильямс, 2011. С. 528.

11. Раковская Е.Е. Кластеризация и ее применение для интеллектуальной обработки текстовой информации // Перспективы развития информационных технологий, 2016. № 32 – С. 45-50

12. Система формирования знаний в среде Интернет [Электронный ресурс] : монография / В.И. Аверченков [и др.]. - Электрон. текстовые данные. - Брянск: Брянский государственный технический университет, 2012. - 181 c. - Режим доступа: http://www.iprbookshop.ru/7006.html

13. Солонин Е.Б. Интеллектуальные технологии поиска и анализа данных // Учебное электронное текстовое издание, 2015.

14. Социологическая энциклопедия: В 2 т. Т.1 / Руководитель научного проекта Г. Ю. Семгин. – М.: Мысль, 2003. – 694 с.

15. Технологии обработки текстовой информации. Технологии обработки графической и мультимедийной информации / А.В. Могилев, Л.В. Листрова. – СПб.: БХВ-Петербург, 2010. – 304с.

16. Цветков В.Я. Информационная неопределенность и определенность в науках об информации // Информационные технологии. 2015. №1. с. 3-7

Электронные ресурсы

17. Технологии анализа данных [Электронный ресурс] // Base Group. URL: https://basegroup.ru/community/articles/basic-conceptions – BaseGroup Labs (дата обращения: 17.11.2018).

18. Web Mining: интеллектуальный анализ данных в сети Internet [Электронный ресурс] // Управление знаниями. URL: https://sites.google.com/site/upravlenieznaniami/tehnologii-upravlenia-znaniami/text-mining-web-mining/web-mining (дата обращения: 18.11.2018).

19. RCrawler: An R package for parallel web crawling and scraping [Электронный ресурс] // URL: https://www.sciencedirect.com/science/article/pii/S2352711017300110 (дата обращения: 18.11.2018)

20. Роберт, И, Кабаков. R в действии. Анализ и визуализация данных в программе R. [Электронный ресурс] / И. Роберт, Кабаков. — Электрон. дан. — М. : ДМК Пресс, 2014. — 588 с. — Режим доступа: http://e.lanbook.com/book/58703

Литература на иностранном языке

21. Robert I.Kabacoff. R in Action: Data analysis and graphics withR / Shelter Island, 2014

22. Data Visualization Tools for Big Data [Электронный ресурс]. – Режим доступа https://blog.profitbricks.com/39-data-visualization-tools-for-big-data/

23. Официальный сайт RStudio – Режим доступа: http://shiny.rstudio.com/

24. Tony Stubblebine. Regular Expression Pocket Reference, Second Edition/ O'Reilly Media, 2007. – 128 p.

25. Gaston Sanchez. Handling and Processing Strings in R, 2014. - 113p.

26. Ronen Feldman, James Sanger. The Text Mining HandBook, 2007. – 423p.

27. PubMed — англоязычная текстовая база данных медицинских и биологических публикаций. - Режим доступа: https://ru.wikipedia.org/wiki/PubMed

28. Data Visualization Tools for Big Data [Электронный ресурс]. – Режим доступа https://blog.profitbricks.com/39-data-visualization-tools-for-big-data/

29. Allaire, JJ, Joe Cheng, Yihui Xie, Jonathan McPherson, Winston Chang, Jeff Allen, Hadley Wickham, Aron Atkins, Rob Hyndman, and Ruben Arslan. Rmarkdown: Dynamic Documents for R., 2017.

30. Rozenberg I. N. Information Construction and Information Units in the Management of Transport Systems // European Journal of Technology and Design. 2016. Vol. 12. Iss. 2. P. 54-62.