СИСТЕМА ТЕМАТИЧЕСКОГО АНАЛИЗА ИНТЕРНЕТ-РЕСУРСОВ

Д. И. Савченко

Введение

В связи с растущими объемами информации, содержащейся во Всемирной Паутине, все сильнее проявляется проблема поиска и классификации этой информации. Имеющиеся поисковики носят общий характер, что, порой, лишь затрудняет процесс выделения необходимого. Зачастую продукты, принадлежащие различным областям, имеют одинаковые названия (продукт питания редис и NoSQL-база данных Redis), что засоряет выдачу поисковика.

В 2000 году количество веб-страниц исчислялось миллиардами. Учитывая, что скорость сбора веб-страниц поисковой машиной Google DeepCrawl [5] составляет несколько тысяч страниц в секунду, можно сделать вывод, что сбор всего содержимого WWW занял бы несколько месяцев. Учитывая, что как количество, там и содержимое страниц непрерывно меняется, этот процесс может занять куда более значительное время.

Наличие тематической поисковой системы позволило бы обозначать конкретную тематику, в которой производится поиск (продукты питания, СУБД и т. д.), а также достаточно сильно сузить список сайтов, которые необходимо посетить, что, в свою очередь, снижает нагрузку на поисковую машину и увеличивает ее производительность и частоту посещения отдельных страниц. И хотя существуют реализации тематических поисковых роботов [1,2,4,6,7], ни одна из них не является промышленной разработкой, так как они создавались как иллюстрации того или иного аспекта тематического поиска. Имеющиеся реализации не предполагают распределения на несколько рабочих узлов, реализованы на в основном Java, что уменьшает сложность разработки, но уменьшает и производительность.

Архитектура тематического анализатора

В данной работе освещается создание прототипа тематического анализатора для поиска веб-страниц, ориентированного на промышленное использование, то есть:

1. анализатор является распределенной системой;
2. каждый элемент анализатора имеет минимальные зависимости от установленных библиотек;
3. такой подход позволяет выполнить развертку системы на большой количестве маломощных машин под управлением легковесной ОС (например, FreeBSD).

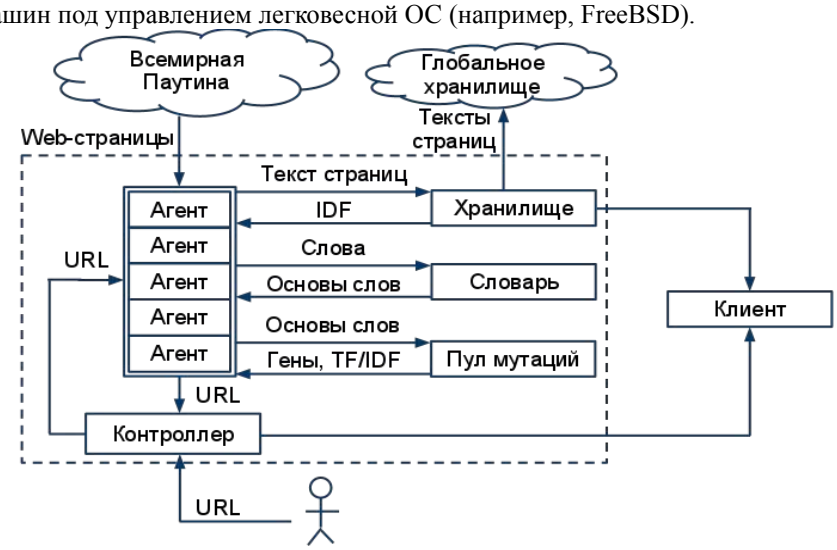

Рис.1. Архитектура системы тематического анализа

Как показано на рисунке, разработанная система состоит из 5 основных типов узлов.

1. *Контроллер* - компонент, занимающийся распределением URL из очереди по свободным агентам, а также мониторингом работы агентов для предотвращения простоев.
2. *Хранилище* - компонент, обеспечивающий хранение текстов страниц и информации, связанной с текстами страниц, является буферным хранилищем собранной информации.
3. *Словарь* - компонент для хранения и поддержания словаря вида «слово» - «корень».
4. *Пул мутаций* - компонент, обеспечивающий хранение слов (генов), а также занимающийся учетом информации, связанной с ними.
5. *Агент* - элемент, сопрягающий в себе сущности анализатора и загрузчика, то есть загружает страницу по ее URL и обеспечивает ее анализ. Выявленные URL передаются в контроллер, содержимое проанализированной страницы передается в хранилище, а список из *n* наиболее

часто встречающихся слов на странице — в пул мутации.

Глобальное хранилище содержит исходные тексты всех собранных страниц и не является элементом системы.

Работа начинается с задания URL стартовых страниц в контроллере. Агенты запрашивают URL из контроллера и анализируют страницу. Если страница соответствует тематике, то ее текст отправляется в хранилище, гиперссылки и список основных слов — в контроллер, иначе агент отправляет только список основных слов в контроллер. Контроллер производит оценку соответствия принятой или отвергнутой страницы заданной тематике и изменяет значение «энергии» агента. Если это значение превышает верхний заданный предел, то агент не мутирует на следующем шаге, если же значение становится ниже нижнего предела, то генотип такого агента принудительно переформировывается из генотипов двух наиболее успешных агентов из работающих.

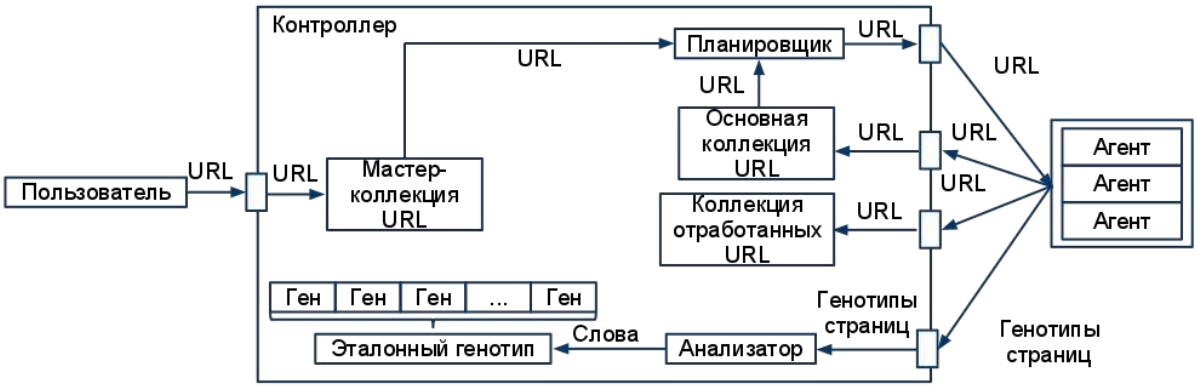

Рис.2. Схема контроллера

Основой контроллера (см. рисунок) является совокупность 3 коллекций:

1. Мастер-коллекция URL - коллекция URL страниц, с которых начинается поиск.
2. Основная коллекция — коллекция URL, поступающих от агентов.
3. Коллекция отработанных URL — URL страниц, которые были получены и проанализированы.

URL страницы является уникальных для всех трех коллекций.

Каждому URL сопоставляются 2 значения — ранг страницы [3] и количество ссылок на нее. Ранг в данном случае выражается следующим образом:

$$R_p = R_p + d \cdot (R_r) ,$$

Где R_p — текущий ранг страницы, R_r — ранг страницы, сославшейся на текущую страницу, d — коэффициент затухания, то есть мера, показывающая, насколько сильно актуальность ссылаемой страницы зависит от актуальности ссылающейся.

Страница для анализа агентом выбирается как первая в очереди, выстроенной по убывания ранга и количества ссылок.

Как упоминалось выше, каждый агент имеет генотип, характеризующий его «предпочтения» в отсеве веб-страниц. Вероятность мутации задается пользователем и не меняется в процессе работы. Всего в рамках данной системы реализованы два вида генетических операций: мутация и кроссинговер. Мутация происходит непрерывно, кроссинговер является последствием также вышеупомянутого переформирования генотипа. Тематика страницы в случае агента и контроллера оценивается как количество совпадений между основными словами страницы и генотипом узла, делённое на длину генотипа [5,7].

Тестирование

Результаты запуска системы в течении 30 минут дают результат, говорящий о том, что система действительно отбирает из анализируемых страниц только те, что действительно соответствуют тематике поиска.

Содержимое мастер-коллекции:

1. http://www.debian.org
2. http://www.debian.org/intro/about
3. http://live.debian.org
4. http://planet.debian.org

Агенты имеют генотип, показанный на рисунке рис.

debian, instal, system, see, set, like, os, free, packag, port

Рис. 3

Контроллер же имеет генотип, изображенный на рисунке. Стоит отметить, что базовое наполнение пула

мутаций совпадает с генотипом контроллера.

debian, instal, system, see, set, like, os, free, linux, packag, port, softwar, apt, pkg, debconf, gnu, languag, support, project, program, freedom, python, develop

Рис. 4 Генотип контроллера

График зависимости количества страниц от времени для проанализированных, собранных и страниц, ждущих обработки (рисунок 4) ясно дает представление о том, что примерно треть из общего числа страниц соответствует выбранной тематике. Кроме этого, на графике, показывающем объем коллекции URL контроллера, можно наблюдать участки, когда кривая становится почти параллельной оси ординат — это участки, когда несколько страниц подряд не были приняты. Большое количество таких участков может говорить о неправильной настройке системы.

На основе собранных данных (текстов страниц, их рангов, а также содержимого пула мутаций) можно построить поисковую систему, выполняющую поиск конкретных страниц по нужной тематике поиска, а также формирующую облако тэгов из страниц, соответствующих пользовательскому запросу. Прототип такой системы SkalarSearch, построенный в процессе работы над анализатором, изображен на рисунке.

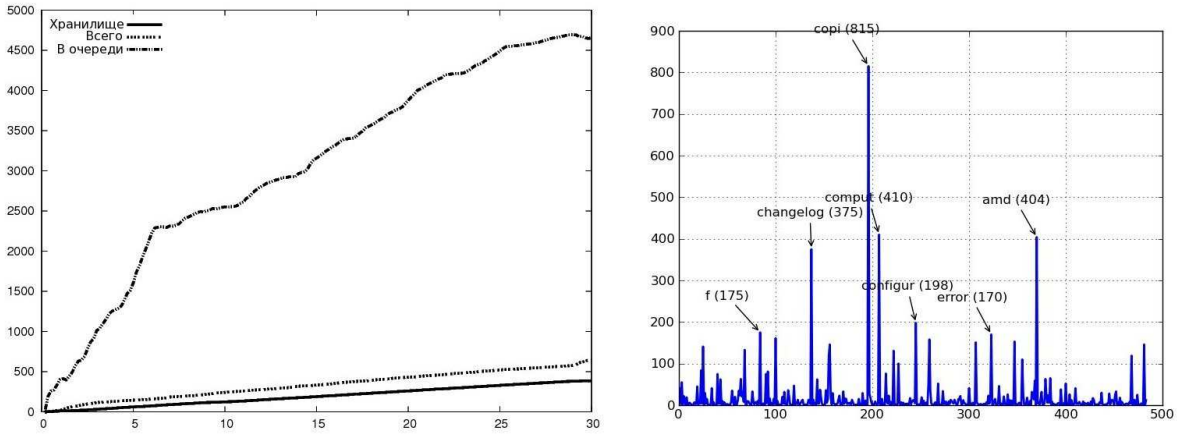

Рис. 5 Тестирование тематической поисковой системы: а) график зависимости количества страниц от времени, б) содержимое пула мутаций.

Кроме того, содержимое пула мутаций (см. рисунок 6) напрямую показывает, какие слова чаще всего употребляются в области, в которой производится поиск, и на каком количестве страниц то или иное слово появлялось, что существенно упрощает анализ подтем в выбранной тематике.

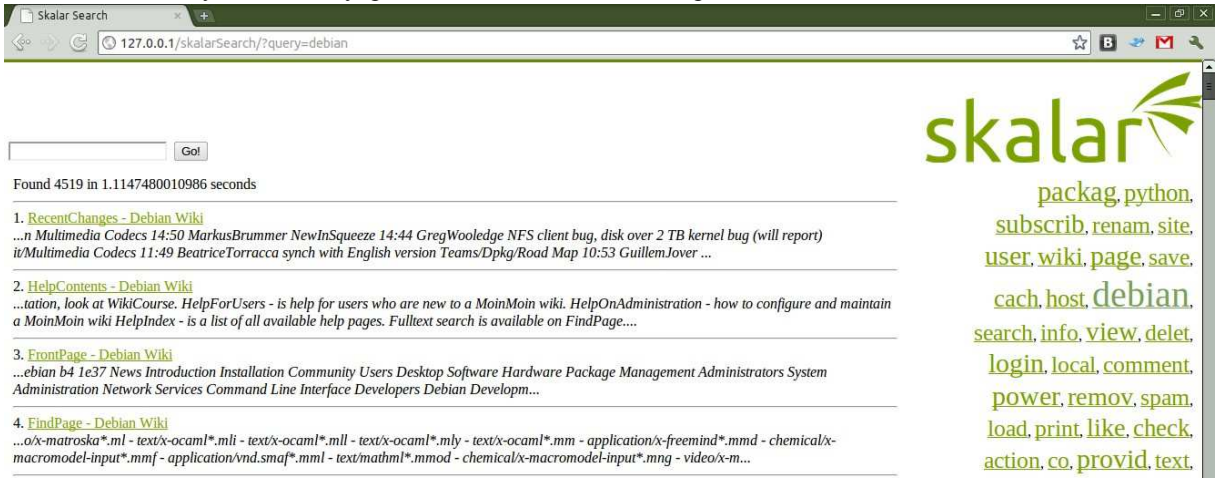

Рис. 6 Внешний вид поисковой системы

Заключение

В статье приведено архитектурное решения для построения распределенного поискового робота, обеспечивающая распределение вычислительной нагрузки, возникающей в процессе анализа и ранжирования страниц. Предложены методы ранжирования и анализа страниц с помощью генетического алгоритма, что позволяет постепенно улучшать качество анализа и релевантность получаемых результатов. Также был построен прототип поисковой системы SkalarSearch, позволяющий выполнять поиск в рамках определенной тематики.

ЛИТЕРАТУРА:

1. Batsakis S., Patrakis E. G. M., Milios E. Improving the perfomance of focused web crawlers // Data &

Knowledge Engineering 68. 2009. 1001-1013.
2. Cecchini R. L., Lorenzetti C. M., Maguitman A. G., Brignole N. B. Using genetic algorithms to evolve a population of topical queries // Information Processing and Management 44. 2008. P. 1863-1878.
3. Cho J., Garcia-Molina H., Page L. Efficient Crawling Through URL Ordering // Computer Networks and ISDN Systems. 1998. Vol. 30. Issues 1-7. P. 161-172.
4. Guilherme T. de Assis, Alberto H. F. Laender, Marcos Andre Gonçalves, Altigran S. da Silva. The Impact of Term Selection in Genre-Aware Focused Crawling // Proceedings of the 2008 ACM symposium on Applied computing. Fortaleza, Ceara, Brazil. 2008. P. 1158-1163.
5. Jayant Madhavan et al. Google's Deep-Web Crawl // VLDB 2008. P. 1241-1252.
6. Menczer F. ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery // In D. Fisher, ed., Machine Learning: Proceedings of the 14th International Conference (ICML97).
7. Rungsawang A., Angkawattanawit N. Learnable topic-specific web crawler // Journal of Network and Computer Applications 28. 2005. P. 97-114.
8. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: Монографія / Під заг. ред. С.О. Субботіна. Запоріжжя: ЗНТУ. 2009. 375 с.