Московский государственный университет имени М.В. Ломоносова
Механико-математический факультет
Кафедра вычислительной математики

Курсовая работа

Поиск упоминаний персон и научных тематик в новостях для выявления возможного конфликта интересов при экспертизе.

Студент: Гвоздев Михаил Александрович
Преподаватель: С.н.с Кривчиков Максим Александрович

Группа: 510

Москва
2022

Содержание

1. Введение

Введение — общее описание проблемной области информационного поиска и поиска имен в частности и формулировка задачи.

В наше время интернет является огромным хранилищем различных данных. Не все они полезны для конкретных задач. В данной работе рассматривается задача поиска экспертов в новостных публикациях, с целью нахождения связности между ними. Связность при этом может быть любого вида. Здесь мы считаем что связь это наличие совместного упоминания в публикации двух и более экспертов. Но чтобы этого достичь нужно научиться правильно искать нужные данные. Поисковые системы реализуют механизм получения срезов данных. Первым этапом в любом поиске является поверхностный сбор информации. Его осуществляют при помощи поисковых пауков [3].

Поисковый паук это программа, которая автоматически обходит определенный заранее список адресов (URL) и заносит полученные страницы в специальную коллекцию. Из полученных страниц, в свою очередь, извлекаются новые URL и добавляются в конец исходного набора. Так они работали изначально, пока весь объем данных в интернете был сравнительно мал. Пауки различаются по типам [11]. В частности, существуют *периодические* пауки общего назначения [8]. Они скачивают все указанные в списке URL, пока не наберут требуемое автору паука количество скачанных страниц и останавливаются. Эта процедура повторяется периодически, когда возникает необходимость в обновлении данных. Они не самые эффективные с точки зрения скорости исполнения, зато отсутствует возможность не обновить заранее заданные страницы. Также существуют *пошаговые* пауки. Они имеют постоянный размер коллекции и продолжают свою работу непрерывно. Их задача беспрерывно работать и заменять наименее ”полезные” страницы на более ”полезные” по некоторому правилу [8]. Такой вид пауков появился в ответ на то, что данные в интернете постоянно меняются. Одни страницы появляются, другие наоборот исчезают. Не все они одинаково ”полезные” и в силу ограниченности доступной памяти некоторые удаляются. Построение универсальной метрики полезности является отдельной сложной задачей. Если считать страницу важной по непрерывному количеству посещений ее, то можно удалить важную страницу на которой раз в месяц оплачиваются счета за важные услуги (например коммунальные). Или же считать полезными только те страницы на которые больше всего ежемесячная аудитория, что делает невозможным использование такого паука с некоторой узкой целью (поиска информации по своей узкой специ-

альности). Следующим типом являются *распределенные* пауки[5]. Они состоят из многих пауков общего назначения, которые проверяют URL только в определенной области интернета, частое использование которой характерно для ограниченной географически территории. Например для определенной страны характерно использование сайтов в основном на ее языке. При этом есть центральный сервер, контролирующий и распределяющий URL между ними. Таким образом, достигается большая отказоустойчивость всей системы, хоть и существует некоторое ограничение скорости в силу использования пауков общего назначения. Эту проблему призваны были решать *параллельные* пауки [2]. В отличие от *распределенных* они обрабатывают единый массив URL, которые разделены на несколько машин, обрабатывающих эти URL параллельно, а не последовательно. Такое улучшение позволило повысить скорость выгрузки страниц. Подтипом пауков общего назначения являются *фокусированные*. Они обходят и добавляют в список дальнейшего обхода только те URL, по которым находится документ соответствующий некоторой теме (допустим теме поискового запроса). При этом используются различные методы подсчета обратных ссылок. Добавление новых ссылок происходит до тех пор пока не наберется нужное количество страниц или не обойдется весь список URL. В Google Inc. в 1998 году для подсчета обратных ссылок использовали PageRank [6].

Для описания алгоритма PageRank зададим следующие условия. Пусть на странице A цитируются страницы $S_1, ..., S_n$ (присутствуют их URL), $d \in (0, 1)$ параметр затухания (в работе брали 0.85), $C(A)$ - общее количество цитируемых страниц на странице A. Тогда $PR(A) = (1-d) + d*(\frac{PR(S_1)}{C(S_1)} + \frac{PR(S_2)}{C(S_2)} + ... + \frac{PR(S_n)}{C(S_n)})$, где PR это PageRank. Существует множество вариантов, выбора параметров для построения фокусированных пауков. Поэтому они, в свою очередь, подразделяются на различные виды в зависимости от способа определения соответствия страницы теме и способа обработки страниц [4].

Когда мы научились правильно искать данные, возникает следующая задача - надежное хранение и быстрый доступ к ним. Отвечая на такой запрос появился язык SQL[10] для работы с реляционными базами данных. Реляционная модель представляет собой набор двумерных таблиц. Каждая таблица состоит из строк - записей и столбцов - полей. Поля обязаны быть одного из допустимых типов. Таблицы могут быть связаны друг с другом при помощи различных ключей (ссылок). Изначально реляционные базы данных создавались для хранения на одной машине сравнительного небольшого объема данных. Когда объем их достаточно вырос старая парадигма баз данных перестала работать. CAP теорема [1] утверждает, что любая система общих данных может обладать наибольшее двумя свойствами из следующих: со-

гласованность - существует только одна актуальная версия данных, доступность - данные доступны в любой момент времени, стабильность - устойчивость к физическим нарушениям связности частей данных. В связи с этим стали появляться нереляционные базы данных и системы для управления ими. Они специализированы под определенные задачи и поэтому могут делать незначительные для их цели допущения в CAP теореме. В наше время можно выделить следующие типы нереляционных моделей данных[9]. Наиболее известной является модель *ключ-значение*. Принцип ее работы похож на идею хеш-таблицы. У каждой записи есть уникальный ключ, которому соответствует единственный хранящий запись сервер. Следующий тип это *документный*. Его идея состоит в том, что документы устроены гораздо сложнее, чем просто текстовые поля. Они могут содержать ссылки на другие документы, которые в свою очередь ссылаются на дополнительные и так далее. *Столбцовая* модель отличается от предыдущих тем, что хранит данны в виде столбцов, а не записей. Основополагающей системой для этой модели является Bigtable[7]. В ней данные хранятся в виде наборов столбцов одинакового типа. При этом таких наборов может быть максимум сотни, в отличие от реляционной модели в которой может быть неограниченное количество столбцов. Дальше идет *графовая* модель. Она строится на основе модели графа из теории графов. Ее преимущество состоит в том что она позволяет эффективно обрабатывать данные с большим числом связей между объектами разной природы.

Следующий этапом поиска является полнотекстовый. В отличие от поисковых пауков он проходит весь текст содержащийся на странице. Пытаясь найти нужную последовательность символов. Это можно делать как каждый раз проходя один и тот же текст или же строить инвертированный индекс[12]. Он представляет собой структуру данных, в которой каждому слову сопоставляются места, где оно упоминается в различных текстах. Такой индекс позволяет значительно ускорить работу поисковых систем.

Перейдем к самой задаче данной работы. Нужно научиться находить ученых в полученных текстах и определять связаны ли они. Под связностью двух людей (рецензента и рецензируемого) будем понимать наличие знакомства, дружбы, родства или других отношений, которые могут оказать влияние на вынесение рецензентом вердикта на научную работу рецензируемого.

Дальше по абзацу на следующее:

1. Способы определения связности найденных слов (через графы знаний, мл, еще сказать про что вообще значит связность)

2. Методы сбора данных

В принципе описание работы CommonCrawl, почему используем его, что именно с него берем, как выбрали что брать.

3. Способы хранения данных

Краткое повторение про виды бд и подвод к тому почему сделали то что сделали. Описание работы загрузчика, описание структуры sqlite бд.

4. Поиск упоминаний персон

Краткое повторение как в принципе их в тексте ищут. Описание работы нашего поиска людей (fts5) и то как мы определяем что они связаны (python).

4.1. Определения

Тоже не знаю что тут определить. Разве что стемминг (мб его альтернативу лемматизацию которую мы не использовали) и полнотекстовый поиск.

5. Заключение

Краткий пересказ того что было сделано. Описание дальнейшей работы при помощи графов знаний, возможно с использованием онтологий.

Либо можно попытаться запихнуть онтологии в эту работу в раздел о способах поиска людей в новостных изданиях. Сделать сравнение что было до, чего добились с ними. Причину добавления вижу в том, чтобы появились формулы/определения/леммы/теоремы в курсовой. Просто неуверен что это правильно на 5 курсе быть без теорем в курсовой. С другой стороны может ничего не выйти и оказаться что в онтологиях никакого мат результата быстро не сделать. Я недостаточно глубоко в них погрузился.

Список литературы

1. CAP Twelve Years Later: How the «Rules» Have Changed // InfoQ [Электронный ресурс]. URL: `https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed/` (дата обращения: 17.05.2022).

2. Parallel Crawlers [Электронный ресурс]. URL: https://ra.ethz.ch/CDstore/www2002/refereed/108/index.html (дата обращения: 05.05.2022).
3. AbuKausar Md., S. Dhaka V., Kumar Singh S. Web Crawler: A Review // International Journal of Computer Applications. 2013. № 2 (63). С. 31–36.
4. Batsakis S., Petrakis E. G. M., Milios E. Improving the performance of focused web crawlers // Data & Knowledge Engineering. 2009. № 10 (68). С. 1001–1013.
5. Boldi P. [и др.]. UbiCrawler: a scalable fully distributed Web crawler // Software: Practice and Experience. 2004. № 8 (34). С. 711–726.
6. Brin S., Page L. The anatomy of a large-scale hypertextual Web search engine // Computer Networks and ISDN Systems. 1998. № 1 (30). С. 107–117.
7. Chang F. [и др.]. Bigtable: A Distributed Storage System for Structured Data // ACM Transactions on Computer Systems. 2008. № 2 (26). С. 1–26.
8. Cho J., Garcia-Molina H. The Evolution of the Web and Implications for an Incremental Crawler С. 18.
9. Grolinger K. [и др.]. Data management in cloud environments: NoSQL and NewSQL data stores // Journal of Cloud Computing: Advances, Systems and Applications. 2013. № 1 (2). С. 22.
10. Melton J. SQL language summary // ACM Computing Surveys. 1996. № 1 (28). С. 141–143.
11. Wireless Communication and Computing) student, CSE Department, G.H. Raisoni Institute of Engineering and Technology for Women, Nagpur, India [и др.]. Study of Web Crawler and its Different Types // IOSR Journal of Computer Engineering. 2014. № 1 (16). С. 01–05.
12. Zobel J., Moffat A. Inverted files for text search engines // ACM Computing Surveys. 2006. № 2 (38). С. 6.