Article retrieval system

Vsevolod Stepanov
Yuri Rebryk
tehnar5@gmail.com
y.a.rebryk@gmail.com

ABSTRACT

This is our report for Information Retrieval project

ACM Reference Format:
Vsevolod Stepanov and Yuri Rebryk. 2017. Article retrieval system. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, **??** pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROJECT DESCRIPTION

We are developing a system for searching scientific articles.

The data for our search engine will be collected from online libraries like arxiv.org and from publishers that have information about their articles available online (like springer.com).

Firstly, we'll crawl a large number of pages from these sites and store them for future processing.

Then, we'll filter irrelevant pages (those that doesn't contain an article description) and process the rest. For each relevant page we'll extract it's title, abstract, list of authors and probably some meta information like date of publishing. Also, we'll store a link for a PDF version of article, if available.

After that, we'll implement a search engine over these articles **to be discussed later**.

The key feature of our service would be creating some kind of graph over authors/articles and their references to each other **to be specified later**

2 SYSTEM DESIGN

We use `Python 3` for development and `Postgresql` for database.

Documents are stored in a filesystem and their metadata and (in future) results of a document processing are stored in database

3 DATA ACQUISITION

For data acquisition we implemented a web crawler. The key details of out implementation are:

(1) Politeness policy.
We follow the constraints defined in `robots.txt`. We do not visit excluded pages, do not store page if there is a `noindex` meta tag, and of course do not spam a website with a lot of queries. Even if a delay is not specified in `robots.txt` we have a default one.

(2) Distributed crawling.
The problem with Python's multithreading is that only one thread can be run at a time because of `GIL`, so we use multiprocessing instead. Each crawler runs in its own process and several crawlers can be run simultaneously.
To prevent a page being downloaded by several processes, each crawler has its own set of `allowed hosts`. This is suitable for us as
   (a) There are not so many data sources with scientific articles so we can just list them
   (b) We aren't interested in links leading to a different domain as most likely that domain will not contain any relevant documents

We store each HTML page we crawled in a filesystem and put some page's metadata (like its URL, hash of page content, date of last page modify) in database. Duplicated pages (with equal hashes) are ignored.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Conference'17, July 2017, Washington, DC, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn