Search engine project

Working title: *retrouve*

This is a toy search engine. It was built with the objectives to:

1. Learn about Python and PostgreSQL
2. Learn about crawling and indexing the web, and learn about how search engines are built
3. Have fun

Components

There are currently two entry points to this program:

- The web front end
  - Only serves as an inteface for searching and submitting new URL's for content autodiscovery.
  - Can be started by running `bin/web`
- The search spider(s)
  - Continuously download URL's from a queue.
  - Can be started by running `bin/spider`
  - Each spider is its own python process, threads are not being used (yet).

Database

This project is built solely on PostgreSQL for storing search data, and maintaining a job queue. There are three types of primary data being stored: **URL's**, **Documents** and **Excerpts**.

URL's

URL's represent all the unique webpages that the search engine knows about. A URL can have any number of jobs referring to it. This is how the spiders know when to index it.

A URL in the database is a tuple, containing the following fields:

```
scheme   0  URL scheme specifier
netloc   1  Network location part
```

```
path     2  Hierarchical path
params   3  Parameters for last path element
query    4  Query component
fragment 5  Fragment identifier
```

Documents

A document represents a URL (location) on the web as it looked at some point in time. The HTTP headers, status code and response body are stored in the database. Multiple documents may be stored for the same URL at different points in time. However, older versions may be purged to free up space.

Excerpts

Every **Document** get parsed and is split into chunks of text. Each of these chunks is called an excerpt. Excerpts have a fulltext GIN-index on them. This is the data that gets matched against the user's queries.

Queue

A special `jobs` table in the database keeps track of which URL's should be crawled by the spiders, en when they should be crawled. The job system is not built exclusively for this purpose,

How the spider works

The spider (worker):

1. Acquires a lock on the jobs table.
2. `SELECT`'s the first job that matches its given constraints.
    - The job must not be locked, must be scheduled now or in the past, and must not have failed more than 3 times.
3. `UPDATE`'s the job, setting `reserved = true` on it.
4. Goes to work.

- Fetches the URL's content.
- Indexes the page.
- Discovers new URL's, scheduling them to the queue based on constraints.
- Stores the fetched page in the `documents` table
    - and stores text fragments in the `excerpts` table.

5. Reschedules the job to be crawled again in the future.

Set up for development

Clone the project:

```
$ git clone git@git.syntaxleiden.nl:mathijs/search_engine.git
$ cd search_engine
```

Set up a database:

```
$ createdb search_engine
$ psql search_engine < sql/*.sql
$ cp .env.example .env
$ $EDITOR .env # Configure the database connection
```

Create a Python virtual environment:

```
$ [sudo] pip install virtualenv
$ virtualenv -p python3 venv
$ source venv/bin/activate
$ pip install -r requirements.txt
```

Run one or more processes:

```
$ bin/web # Start the web front end
$ bin/spider # Start a single spider process
```

That should be it! The front end can be found at http://localhost:5000, and this is where you can submit the first URL's to. After submitting, the crawler(s) will start discovering new links right away.