[RFC] Enable search on file content

Summary: Allow users to retrieve files in dropbox containing relative data

Created: Jun 04, 2021
Current Version: 1.0.0
Target Version: 1.1.0

Status: WIP | **In-Review** | Approved | Obsolete

Owner: abhilash.raj.me@gmail.com
Contributors: abhilash.raj.me@gmail.com
Approvers: pr@borneo.io

To provide a product for users for context based search for files stored in dropbox containing relevant information. To have exposed end-points for the product for easy integration with other products that would require the services.

Background

Organising cloud stored documents can easily become chaotic, making search for the relevant required files a painstaking process. Manual process of searching and retrieving files is prone to error and it's probability rises exponentially with large files.

Hence, the product is needed to facilitate users to easily perform context based search and retrieve the relevant files based on it's content.

Proposal

The Dropbox files text contents are extracted, along with their shareable link from the dropbox and indexed in Elasticsearch periodically. Users can perform searches on the text content stored on Elasticsearch and matched file name and shareable links are shared with the user.

Backend API

The backend API is developed using python with flask framework. Two Endpoints **/search** and **/sync** are exposed. The search endpoint takes requests with the search token and performs the search on the Elasticsearch index and returns files containing the search token.

The sync endpoint takes requests for manual triggers of synchronization of dropbox contents and elasticsearch index. Other than the manual requests, the synchronization can be configured to run periodically.

Since the periodic synchronisation process and manual trigger for synchronization can occur simultaneously and can cause fatal errors, a static json file is being used as the synchronization resolver, which is being updated at beginning and at the end of the synchronization. The status of the json is being checked before starting the synchronization, if the status is set to true then other synchronization requests because of ongoing synchronization (manual or scheduled) will get rejected.

Implementation

The solution is implemented using Python 3 and Flask framework. Below libraries were used which are not under the python standard installation.

● docx	:	To extract text from docx files
● json	:	To parse the synchronization resolver
● pandas	:	To load and store different data
● dropbox	:	To connect to the dropbox api
● pdftotext	:	To extract text from pdf files
● apscheduler	:	To schedule periodic synchronization of dropbox files with DB
● elasticsearch	:	To connect to elasticsearch DB

Synchronization occurs In following steps:

1. Synchronization resolver state is updated as True
2. Connecting to the Dropbox API using the dropbox library and retrieving the files, it's content ,shareable link and date modified.
3. Text is extracted from the file's content.
4. File name, text, shareable link and date modified are stored in pandas dataframe.
5. The pandas dataframe is then converted into list of dictionary(row wise) and indexes in elasticsearch using the elasticsearch library
6. Synchronization resolver state is updated as False

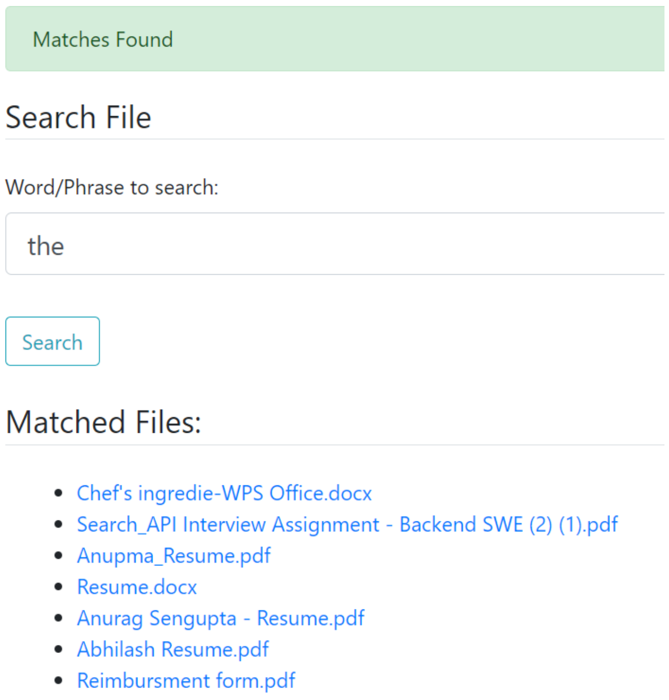
Search occurs in the following steps:

1. Search request is analysed for query q. If not present, it is treated as a bad request.
2. Synchronization resolver state is checked in case of any ongoing synchronization, if True, response is sent back stating the same.
3. Double quotes are removed from the search token, which are present in case of search phrases.
4. Elasticsearch Index is queried using elasticsearch library and the relevant data is extracted from the response.
5. If matches are found the data is stored in a pandas dataframe and shared to the endpoint.
6. The pandas dataframe is converted into a list of dictionaries which is then sent in the form of JSON in the response.

UI

A basic UI has been developed using Python 3, Flask framework, Jinja template and bootstrap 4. The UI has one input text box and one search button. On form submission, the form is validated for the input field to be non-empty. On successful form validation, a request is sent to the Backend API search endpoint and on receiving the response, it is interpreted by the UI. In all the cases, the messages come from the backend API.

- **In case of files found, the same page shows file names linked with their shareable link, along with a success alert message.**


Matches Found

Search File

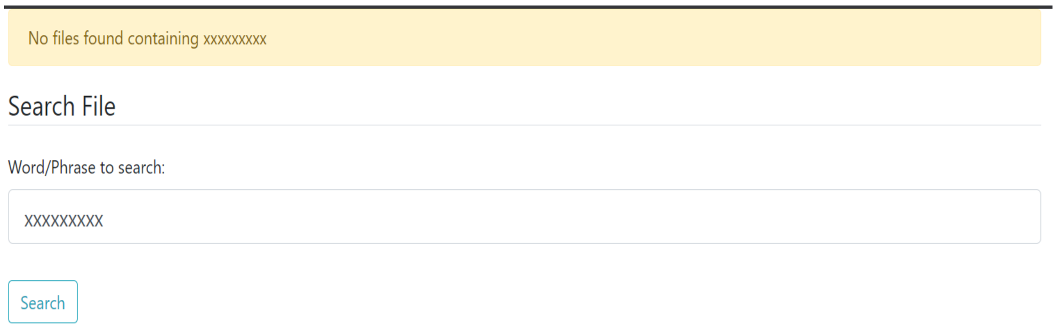
Word/Phrase to search:

the

Matched Files:

- Chef's ingredie-WPS Office.docx
- Search_API Interview Assignment - Backend SWE (2) (1).pdf
- Anupma_Resume.pdf
- Resume.docx
- Anurag Sengupta - Resume.pdf
- Abhilash Resume.pdf
- Reimbursment form.pdf

- **In case of files not found or DB being synchronized, it gives a warning alert message.**

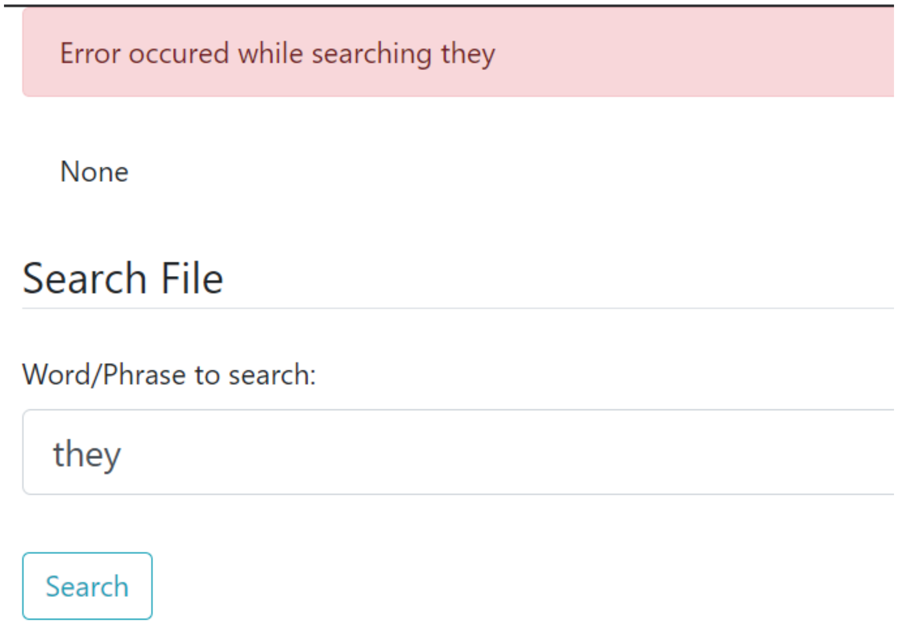

No files found containing xxxxxxxxx

Search File

Word/Phrase to search:

xxxxxxxxx

- **In case of any other error, it gives the danger alert message.**



Error occured while searching they

None

Search File

Word/Phrase to search:

they