1. Scrapes the UOW webpages and stores it in “uow-scrape.json”

* This iterate through the sitemap scrape and get each page’s content and title
* Data is sent to a SQLite3 DB.
* Data name “web2.db” includes 3 fields (title, url, content)

1. Extract the key words from the “content” of web2.db - Using RAKE (Rapid Automated keyword extraction)

* First, RAKE splits the text into sentences and generates the candidates.
* Second, RAKE computes the properties of each candidate, which is the sum of the scores for each of its words. The words are scored according to their frequency and the typical length of a candidate phrase in which they appear.
* Finally, the keyword candidates is ranked based on RAKE’s scores. Here, I chose they keyword with score > 3
* There are 2 codes to extract the key word by RAKE:
  + Rakeclass.py : includes the modules and function for extracting the key words.
  + Generatekeyword.py: read the data “web2.db” then generate the keyword from “content” and return the results in “keyword” and write it in the last column of data.

1. Build a recommendation search UI using React and Elasticsearch

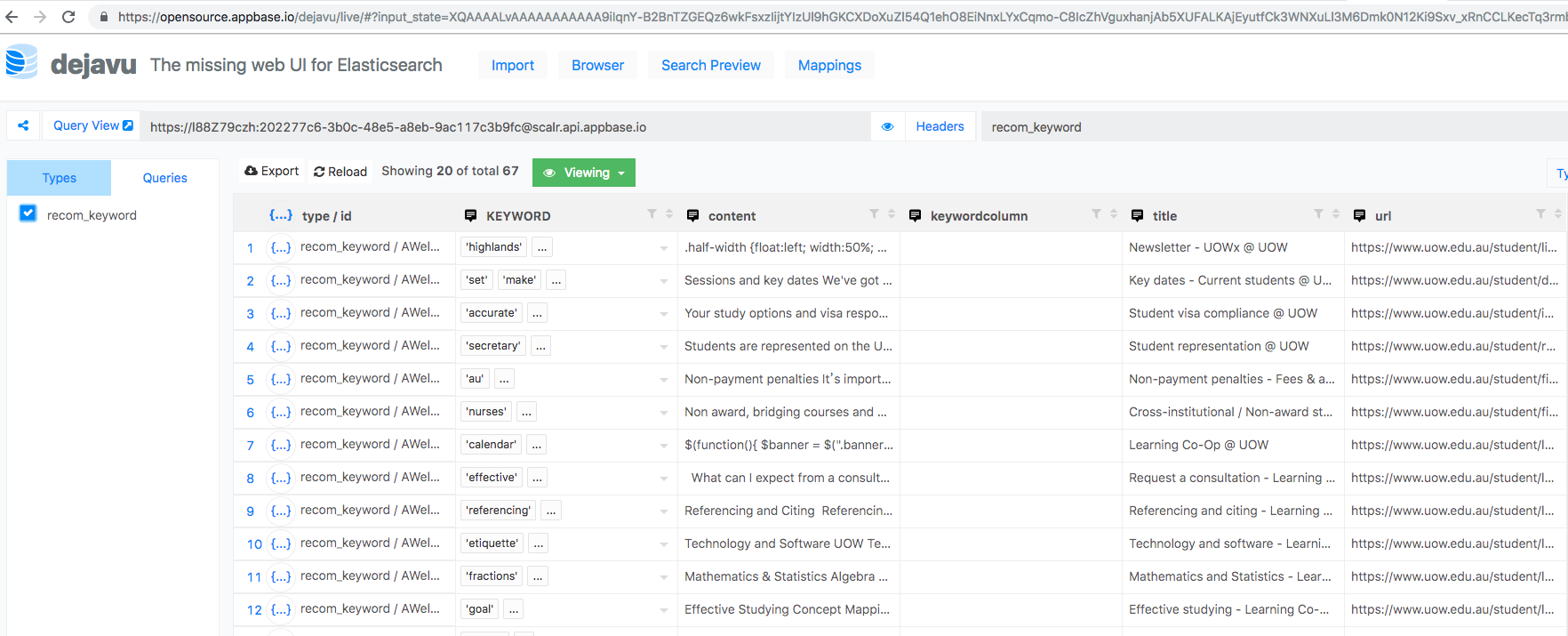
* Using appbase.io to create the index and schema for data used in Elasticsearch index.

Fig. 1 Import data into appbase.io

* Initialize a boilerplate with Create React App setup:

npm install -g create-react-app

# install CRA if you don't have it.  
create-react-app recomendationsytem

# initialize the boilerplate.  
cd recomendationsystem

* Test the default CRA app:

npm start



* Add ReactiveSearch:

npm install --save @appbaseio/reactivesearch

* All the ReactiveSearch components are wrapped inside a container component — [ReactiveBase](https://opensource.appbase.io/reactive-manual/getting-started/reactivebase.html" \t "_blank) which glues the Elasticsearch index and the ReactiveSearch components together. Edit src/App.js file.
* import React, { Component } from 'react';  
  import {  
   ReactiveBase,  
   DataSearch,  
   SingleRange,  
   ResultList  
    
  } from '@appbaseio/reactivesearch';  
  class App extends Component {  
   render() {  
   return (  
   <ReactiveBase  
   app="webRecom"  
   credentials="oHyDAVIi2:38acfba8-c9f8-443c-8b2b-52ff6e30fddb"  
   >  
   <DataSearch  
   componentId="mainSearch"  
   dataField={["KEYWORD", "KEYWORD.search"]}  
   queryFormat="and"  
   iconPosition="left"  
   />  
    
   cd ..  
    
   <ResultList  
   componentId="results"  
   dataField="content"  
   react={{  
   "and": ["mainSearch", "KEYWORD"]}}  
   pagination={true}  
   size={10}  
   onData={(res)=>(  
   {  
    
   "title": res.title || " ",  
   "description":  
   "</span><br/><br/><div class='result-author' TITLE='" + res.title + "'>by " + res.url + "https://google.com/search?q=" + res.url+"</div>",  
   "url": res.url  
   }  
   )}  
   className="result-data"  
   innerClass={{  
   "resultStats": "result-stats"  
   }}  
   />  
    
    
   </ReactiveBase>  
   );  
   }  
  }  
  export default App;

ResultList

DataSearch box

Credential is got from data index (appbase.io)

* Beside, ReactiveSearch offers 20+ components for creation.

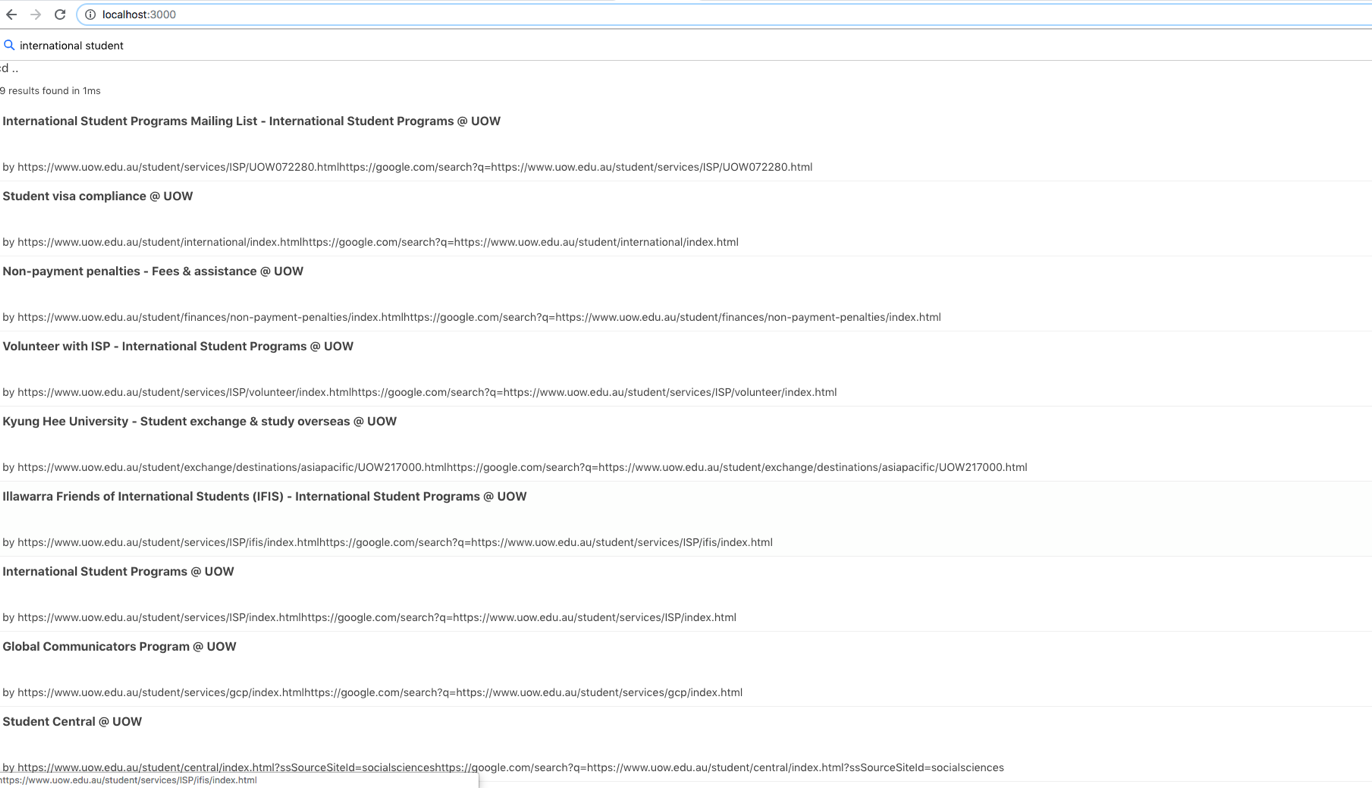


Fig. 2 Recommendationsystem search UI using React and Elasticsearch