

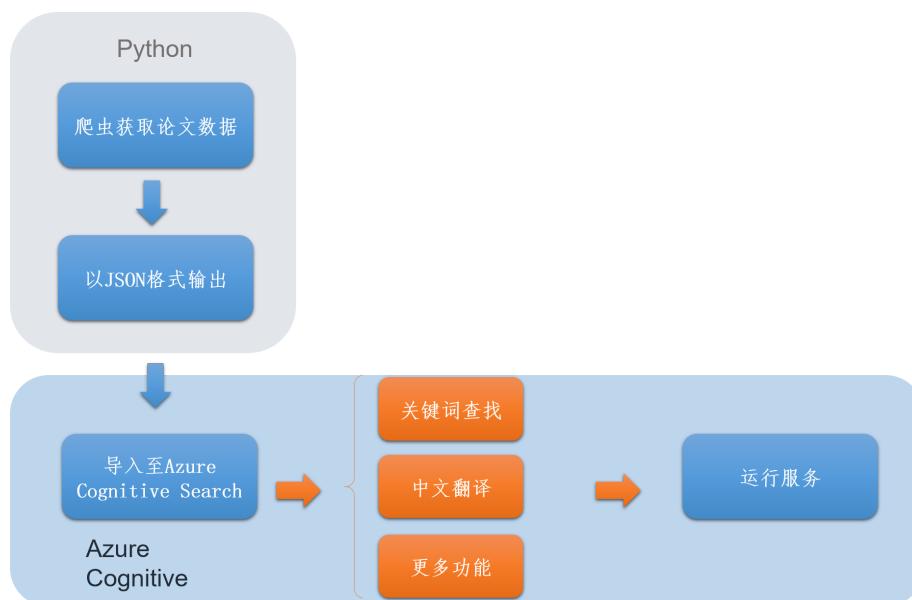
# 基于Azure Cognitive Cloud Search的COVID-19相关论文数据库搭建

## 背景

近期，新型冠状病毒感染的肺炎疫情时时刻刻地牵动着每一个中国人的心。无论身在何地，相信大家对这次的疫情都深有体会——居家隔离、小区门禁、抢不到的口罩、延迟开学等等，这次疫情已经影响到了我们生活的各个方面。这次疫情同样也在学术界掀起了轩然大波。各个领域的学者、研究机构都纷纷投入到新冠肺炎的研究中，一时间，各大期刊上关于covid-19的论文数不胜数。这些论文数据是非常重要的研究材料，需要重点收集整理，那么，如何才能快速、有效地搭建一个关于covid-19的论文数据库呢？微软Azure就提供了这样一个平台，基于Azure Cognitive Cloud Search可以很快地搭建一个数据库，并且可以直接使用其中提供的翻译、关键词提取等服务，为我们省去了手动翻译、人工提取关键词的时间。

## 构建流程

主要的数据库搭建流程如下所示。



首先，对相关网页进行分析，并且从网站上爬取论文数据，并且以JSON格式输出（当然也可以选择其他格式，本文中重点讨论json格式的存储）。在Azure中，创建Cognitive Search服务，并导入数据，在服务中加入AI选项。在服务建立成功之后，即可运行，进行文档搜索啦。整个流程非常简单明了，下面，我们一起来就此项目从头演示一遍。

## Python爬取论文数据

## 选取论文网址

本文选取的数据库，是之前在本公众号平台上就已经介绍过的Elsevier设立的研究论文免费浏览以及下载的网址：

### Elsevier COVID-19 Info

在这个网址中的如图所示部分，可以找到elsevier开源的所有COVID-19相关论文：

Clinical information   中文资源 (Chinese-language)   Research / Drug discovery

#### Articles in Elsevier journals

Below are a selection of articles curated by our clinical solutions team and directly relevant to Novel Coronavirus (2019-nCoV).

In addition to these, Elsevier has made more than 2,500 coronavirus-related articles freely available for the next six months (commencing 10/02/2019). You can find these articles [here](#) .

### COVID-19开源论文

这里也推荐COVID-19科研动态监测，同样可以查询到很多的论文数据。当然，大家可以去爬取微博上关于COVID-19的讨论动态等等。

上面的网页打开之后如下所示，为了演示的方便起见，我们选择2019与2020年的论文建立数据库。最后我们选择的链接是[最终url](#)。

Find articles with these terms

Title, abstract, keywords: Coronavirus OR "Corona virus" OR "2019-nCoV" OR "SADS-CoV" OR "SARS-CoV"...

Advanced search

2,534 results

Set search alert

Refine by:

Years

☐ 2020 (90)

☐ 2019 (115)

☐ 2018 (128)

Show more

Article type

☒ Review articles (285)

☐ Research articles (2,249)

Publication title

☐ Download selected articles   Export

☐ Review article   Full text access

Coronavirus Disease 2019: Coronaviruses and Blood Safety

Transfusion Medicine Reviews, In press, corrected proof, Available online 21 February 2020

Le Chang, Ying Yan, Lunan Wang

Download PDF   Abstract   Export

☐ Review article   Full text access

Coronavirus Disease 2019 (COVID-19) and Pregnancy: What obstetricians need to know

American Journal of Obstetrics and Gynecology, In press, journal pre-proof, Available online 24 February 2020

Sonja A. Rasmussen, John C. Smullan, John A. Lednický, Tony S. Wen, Denise J. Jamieson

Download PDF   Abstract   Export

☐ Review article   Full text access

Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) and coronavirus disease-2019 (COVID-19): T1

International Journal of Antimicrobial Agents, In press, corrected proof, Available online 17 February 2020, Article 105924

Chih-Cheng Lai, Tzu-Ping Shih, Wen-Chien Ko, Hung-Jen Tang, Po-Ren Hsueh

Download PDF   Abstract   Export

## 分析网页结构

这里我们使用的爬虫工具是Python上的lxml库，可以根据网页的html结构进行爬虫，对于搭建小型爬虫非常的方便。

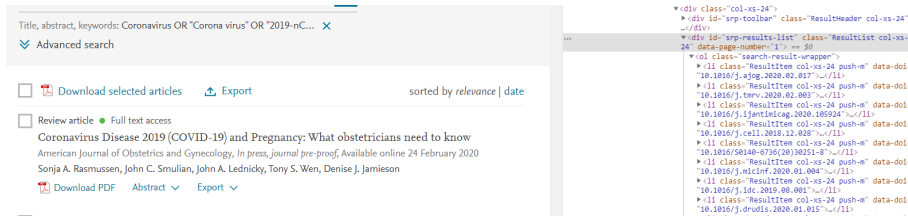
爬虫的主要想法是，首先在主页面上爬取到每个论文的链接，以及论文的标题（论文的标题是为了文本分析使用，对数据库的建立没有太大的影响，在这里可以忽略），然后再进入每个论文的页面，爬取每篇论文的详细信息。

### 主界面爬取论文链接

首先设置主页面链接以及请求头部信息。

```
global_url = 'https://www.sciencedirect.com/search/advanced?
tak=Coronavirus%20OR%20Corona%20virus%22%20OR%202019-nCoV%22%20OR%20SADS-CoV%22%20OR%20SARS-CoV%22
%20OR%2022MERS-CoV%22%20OR%20E2%80%9CSevere%20Acute%20Respiratory%20Syndrome%E2%80%9D%20OR%20E2%80%9CMiddle
%20East%20Respiratory%20Syndrome%E2%80%9D&articleTypes=REV%2CFLA&show=100&ent=true&years=2020%2C2019&
lastSelectedFacet=years'
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
76.0.3809.87 Safari/537.36'
}
```

查看网页源代码，找到包含了每篇论文信息的部分。



可以看到，论文的搜索结果都包含在`<ol class="search-result-wrapper">`的标签中，每一篇论文的详细信息都包含在`<li>`标签中。

一篇论文的详细信息如下：

```
<ol class="search-result-wrapper">
  <li class="ResultItem col-xs-24 push-m" data-doi="
    10.1016/j.ajog.2020.02.017">
    <div class="result-item-container u-visited-
      link">
      <div class="DownloadCheckbox select-result-
        container">...</div>
      <div class="result-item-content">
        <div class="OpenAccessArchive hor">...</div>
        <h2>
          <span>
            <a href="/science/article/pii/
              S0002937820301976" class="result-list-
                title-link u-font-serif text-s" data-rank=
                  "1" data-docsubtype="rev" data-aa-region=
                    "srp-results-list" data-aa-name="srp-
                      article-guest-was-entitled-with-entitled-
                        rank-1" data-hack="#">
                        "Coronavirus Disease 2019 (COVID-19)
                          and Pregnancy: What obstetricians need
                            to know"
                      </a>
                    </span>
                  </h2>
                <div class="SubType hor">...</div>
                <ol class="Authors hor">...</ol>
                <div class="PreviewLinks">...</div>
            </li>
          </ol>
        </div>
      </li>
    </ol>
```

追根溯源，我们找到了包含论文标题以及论文具体地址的标签`<a>`，我们可以通过`text()`以及`@href`来爬取获得其具体的链接以及标题。

这里需要强调的是，由于我在进行爬虫的过程中，`div`以及`ol`的`class`不知为何无法对上，所以代码里只能从头开始一个标签一个标签地写。实际上并不需要如此繁杂，可以直接使用`//div[@class="..."]`这样的书写格式来简化标签的搜索过程，在之后的代码中我们可以看到，这种搜索方法更加简便。

话不多说，本节的代码如下，定义在 `getelsevier` 函数中。

```
def getelsevier(url, headers, page):
    for i in range(page):
        paper_url = global_url + '&offset=' + str(page * 100)
        try:
            response = requests.get(url, headers = headers)
            print('successfully get')
        except:
            pass
        global_html = etree.HTML(response.content)
        page_paper_list = global_html.xpath('/html/body/div/div/div/div/div/div/div/div/div/div/div[2]/div[2]/ol/li/div/div/h2/span/a/text()')
        paper_list.append(page_paper_list) #print titles
        page_paper_href = global_html.xpath('/html/body/div/div/div/div/div/div/div/div/div/div/div[2]/div[2]/ol/li/div/div/h2/span/a/@href')
        paper_href.append(page_paper_href)
        print('Getelsevier end.')
    return paper_list, paper_href
```

这样我们就获得了所有的论文链接。接下来我们来分析单个的论文链接。

## 单个论文网页分析

单个论文爬取的任务较重，需要爬取一篇论文的作者、摘要、题目、关键词以及DOI。虽然之前有爬取过论文的标题，但是由于之后爬取能够得到全部信息的论文并不一定与之前爬取得到的论文的标题一致，所以在这里还需要再爬取一次。

每个论文的尾部链接都储存在之前的 `paper_href` 中。对每一个链接进行补全，并且访问，爬取其中的论文信息。

网页的主要布局如下，很多文章中还包含了文章的简要介绍视频，以及一些图表，有条件的同学可以自行爬取。

Expert Review	<div data-bbox="448 1328 992 1408"> <h1>Coronavirus Disease 2019 (COVID-19) and Pregnancy: What obstetricians need to know</h1> </div> <div data-bbox="992 1328 1129 1408"> <p>➔ <b>Title</b></p> </div>
	<div data-bbox="448 1408 992 1462"> <p>Sonja A. Rasmussen MD, MS <sup>1, 2, 8, ✉</sup>, John C. Smulian MD, MPH <sup>3</sup>, John A. Lednický PhD <sup>4</sup>, Tony S. Wen MD <sup>3</sup>, Denise J. Jamieson MD, MPH <sup>5</sup></p> </div> <div data-bbox="992 1408 1129 1462"> <p>➔ <b>Author</b></p> </div>
	<div data-bbox="448 1462 992 1516"> <p> <a href="#">Show more</a>  <a href="https://doi.org/10.1016/j.ajog.2020.02.017">https://doi.org/10.1016/j.ajog.2020.02.017</a> </p> </div> <div data-bbox="992 1462 1129 1516"> <p>➔ <b>DOI</b></p> </div>
	<div data-bbox="448 1516 992 2000"> <p><b>Abstract</b></p> <p>Coronavirus Disease 2019 (COVID-19) is an emerging disease with a rapid increase in cases and deaths since its first identification in Wuhan, China, in December 2019. Limited data are available about COVID-19 during pregnancy; however, information on illnesses associated with other highly pathogenic coronaviruses (i.e., severe acute respiratory syndrome (SARS) and the Middle East respiratory syndrome (MERS)) might provide insights into COVID-19's effects during pregnancy.</p> <p><b>Key Words</b></p> <p>novel coronavirus; 2019 novel coronavirus; 2019 nCoV; SARS-CoV-2; -novel coronavirus; 2019-nCoV; severe acute respiratory syndrome; SARS; Middle East Respiratory Syndrome; MERS; pregnancy; fetus; newborn; pneumonia; preterm birth; maternal death; fetal death; vertical transmission; perinatal infection; sepsis</p> </div> <div data-bbox="992 1516 1129 2000"> <p>➔ <b>Abstract and Key Words</b></p> </div>

对应的HTML结构如下：

```
    ><div class="Publication" id="publication">...</div>
  ▼<h1 id="screen-reader-main-title" class="Head u-font-serif u-h2 u-margin-s-ver">
    ▶<div class="article-dochead">...</div>
    ▼<span class="title-text">
      "Coronavirus Disease 2019 (COVID-19) and
      Pregnancy: What obstetricians need to know"
    </span>
  </h1>
  ▼<div class="author-group" id="author-group">
    <span class="sr-only">Author links open
    overlay panel</span>
    ▼<a class="author size-m workspace-trigger"
    name="bau1" href="#!">
      ▼<span class="content">
        ▼<span class="text given-name">
          ::before
          "Sonja A."
        </span>
      </span>
    </a>
  ▼<div class="DoiLink" id="doi-link">
    <a class="doi" href="https://doi.org/10.1016/
    j.ajog.2020.02.017" target="_blank" rel="noreferrer
    noopener" aria-label="Persistent link using digital
    object identifier" title="Persistent link using
    digital object identifier">
      https://doi.org/10.1016/j.ajog.2020.02.017</a>
  ▼<div class="Abstracts u-font-serif" id="abstracts">
    ▼<div class="abstract author" id="abs0010" lang=
    "en"> == $0
      <h2 class="section-title u-h3 u-margin-l-top u-
      margin-xs-bottom">Abstract</h2>
      ▼<div id="abssec0010">
        ▼<p id="abspara0010">
          "Coronavirus Disease 2019 (COVID-19) is an
          emerging disease with a rapid increase in
          cases and deaths since its first
          identification in Wuhan, China, in December
          2019. Limited data are available about COVID-
          19 during pregnancy; however, information on
          illnesses associated with other highly
          pathogenic coronaviruses (i.e., severe acute
          respiratory syndrome (SARS) and the Middle
          East respiratory syndrome (MERS)) might
          provide insights into COVID-19's effects
          during pregnancy."
        </div>
      </div>
    </div>
  </div>
```

```

</div>
▼<div class="Keywords u-font-serif">
  ▼<div id="kwrds0010" class="keywords-section">
    <h2 class="section-title u-h3 u-margin-l-top u-margin-xs-bottom">Key Words</h2>
    ▼<div id="kwrds0010" class="keyword">
      ▼<span>
        "novel coronavirus"
      ::after
    </span>
    ::after
  </div>
  ▶<div id="kwrds0015" class="keyword">...</div>
  ▶<div id="kwrds0020" class="keyword">...</div>
  ▶<div id="kwrds0025" class="keyword">...</div>
  ▶<div id="kwrds0030" class="keyword">...</div>

```

如上图所示，标题存储在<span class="title-text">中。作者常常有多个，这里只爬取前几个作者，每一个作者的信息都存储在<a>标签下。文章的摘要存储在<div class="abstract author">中，通过`text()`可以爬取到文章摘要。关键词的爬取与作者的爬取比较相似，关键词全部存储在<div class="Keywords u-font-serif">中，通过遍历可以提取出所有的关键词。

分析完网页结构之后就可以写这一部分的代码啦。这里设置了`timeout`，用来控制时间。

```

Defne getpaper

[10] ▶ Mi
url_list = []
title_list = []
abstract_list = []
main_authors = []
doi_list = []
keyword_list = []
paper = []
def getpaper(paper_href, headers, timeout = None):
    base_url = 'https://www.sciencedirect.com'
    for single_href in paper_href:
        com_url = base_url + single_href # should all be strings
        try:
            single_resp = requests.get(com_url, headers = headers, timeout=30)
            print('successfully get')
            single_html = etree.HTML(single_resp.content)
            single_title = single_html.xpath('//span[@class="title-text"]/text()')[0]
            single_abstract = single_html.xpath('//div[@class="abstract author"]/div/p/text()')
            single_author = single_html.xpath('//div[@class="author-group"]/a/span/text()')
            single_doi = single_html.xpath('//div[@class="DoiLink"]/a/text()')[0]
            single_keyword = []
            for keyword in range(1,4):
                keyword_xpath = '//div[@class="Keywords u-font-serif"]/div/div['+str(keyword)+']/span/text()'
                single_keyword.append(single_html.xpath(keyword_xpath)[0])
            single_paper = {"url":com_url, "title":single_title, "abstract": " ".join(single_abstract),
                           "author":single_author, "doi":single_doi, "keyword": single_keyword}
            paper.append(single_paper)
            print('complete')
        except:
            pass
    return paper

```

至此，我们的爬取任务就差不多完成啦！最后运行主函数，并以json格式输出，看看有没有错误。

```
Get paper

[17] > Mi

if __name__ == '__main__':
    global_url = 'https://www.sciencedirect.com/search/advanced?
    take=coronavirus%20OR%20corona%20virus%22%20OR%2022019-nCoV%22%20OR%2022SADS-CoV%22%20OR%2022SARS-CoV%22%20
    OR%2022MERS-CoV%22%20OR%2022%20F%20%20Csevere%20Acute%20Respiratory%20Syndrome%22%20OR%202019-nCoV%20Middle%20East%20
    Respiratory%20Syndrome%22%20&articleTypes=REV%20&show=100&ent=true&years=2020%2C2019&
    lastSelectedFacet-years'
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
        76.0.3809.87 Safari/537.36'
    }
    ncov_paper_list, ncov_paper_href = getelsevier(url = global_url, headers = headers, page=2)#Because there
    are only 2 pages about ncovpaper, so we set 2 here.
    paper_json = getpaper(ncov_paper_href[0], headers = headers, timeout = 5)
    with open("C:/Users/Administrator/Desktop/Intership/data.json", "w") as dump_f: #output
        json.dump(paper_json, dump_f)

successfully get
complete
successfully get
complete
successfully get
complete
successfully get
complete
```

看来运行的还是不错的！就是速度比较慢.....这个爬虫的效率并不是非常高的，希望大家能够在这个爬虫的基础上进行改进啦。

## Azure Cognitive Cloud Search进行存储

首先介绍一下我们输出的json文件的结构。为了便于存储以及查阅，我将数据写出为json array，在python里面就是存储了字典的list，每一个字典对应了每一篇paper。现在我们将其导入到azure blob存储。

在这里选用Azure Cognitive Cloud Search功能，主要是因为其内在的AI功能。相信大家一定遇到过这样的情况——在网上查阅文献，有时候文献中的生词很多很多，真的不太清楚这篇文献在讲什么。假设我们现在负责一个数据库，客户要求我们找到某篇论文，并且用中文告诉客户这篇文章的概要。使用Azure Cognitive Cloud Search，我们可以直接获得这些文档的中文摘要。下面就来演示Azure Cognitive Cloud Search的数据库建立。

## 导入数据

Azure Cognitive Search服务兼容很多的数据类型，包括Azure虚拟机中的SQL服务器，SQL Database，JSON blobs, csv等等。这里建立索引较为方便的是JSON blobs以及SQL Database，所以我在这里主要使用了JSON blob的方式。

登录到Azure Portal的首页，在Azure服务中点击存储账户，在存储账户中点击添加：



按照上图新建存储账户。创建好之后，转到存储账户，找到Blob服务，点击容器，点击+容器，即新建一个blob容器：

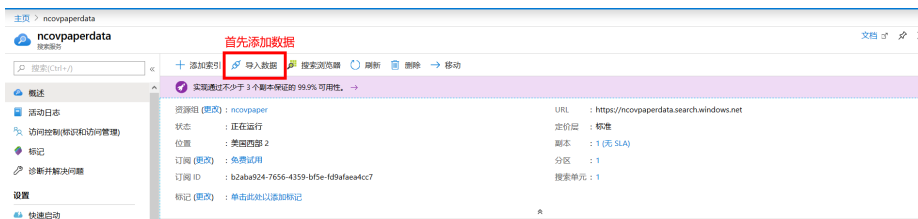




## 创建搜索服务并进行配置

点入我们以及新建好的blob容器，就可以进入上传我们爬取的json文件啦。上传好之后，回到主页，点击创建资源，并搜索Azure Cognitive Search，进入点击创建。在查看创建中，选择我们刚刚创建的资源组，并创建搜索服务。

因为我们的数据是JSON blob，所以我们可以导入数据的同时，创建索引。点击导入数据以上传数据。



进入导入数据页面需要仔细配置，首先在连接到数据部分，导入数据并等待Azure 提取数据信息：

### 导入数据

[连接到你的数据](#) [添加认知技能\(可选\)](#) [自定义目标索引](#) [创建索引器](#)

使用来自当前订阅中现有 Azure 数据源的数据创建和加载搜索索引。Azure 认知搜索 会爬网你提供的数据结构、提取可搜索内容、视需要使用认知技能扩充它并将它加载到索引中。[Learn more](#)

数据源	Azure Blob 存储	
数据源名称 *	data	刚刚导入数据的文件名
要提取的数据 ①	内容和元数据	
分析模式	JSON 阵列	选择默认或者JSON ARRAY
连接字符串 *	DefaultEndpointsProtocol=https;AccountName=ncvpaper;AccountKey ...	
容器名称 *	ncvpaper	选择刚刚创建的blob
blob 文件夹 ①	在此处输入你的文件夹名	
说明	(可选)	

下一步: 添加认知技能(可选)

注意在这里我碰到过几次无法读取，这可能跟网络有关，多尝试几次就好了，但一定要保证JSON文件的格式是正确的。



主页 > nconv数据 > 导入数据

导入数据

技能组名称 \* ①

azureblob-skillset

可以在这里建立自己的技能组

应用 OCR 并检索所有文本都会分割到 'merged\_content' 字段中 ②

源数据字段 \*


abstract

选择你想要添加AI服务的字段

扩充粒度级别 ③

源字段(默认)

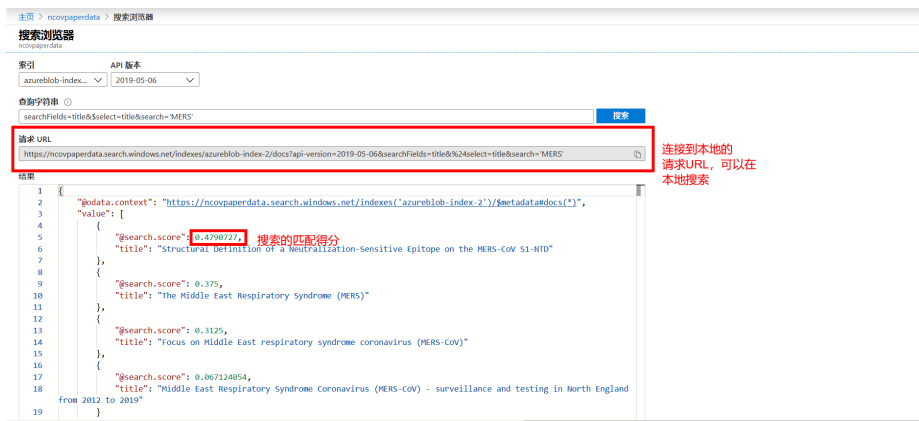
<input checked="" type="checkbox"/> 文本认知技能	参数	字段名称
<input type="checkbox"/> 提取人名		people
<input type="checkbox"/> 提取组织名称		organizations
<input type="checkbox"/> 提取地名		locations
<input checked="" type="checkbox"/> 提取关键短语		keyphrases
<input checked="" type="checkbox"/> 检测语言		language
<input checked="" type="checkbox"/> 翻译文本	目标语言 <span>简体中文</span>	translated_text

主頁 > ncovpaperdata > 导入数据								
导入数据								
字段名称	类型	可检索	可筛选	可排序	可查找	可搜索	分析器	建议器
url	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...
title	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...
abstract	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...
author	Collection(E...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...
doi	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...
<input type="text" value="keyword"/>	<input type="text" value="Collection..."/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...
 AzureSearch_DocumentKey	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		...
metadatas_storage_content_type	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		...
metadatas_storage_size	Edm.Int64	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			...
metadatas_storage_last_modified	Edm.DateTi...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			...
metadatas_storage_name	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		...
metadatas_storage_path	Edm.String	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		...
translated_text	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	中文(简体) - Lu... <input type="text"/>	...
keyphrases	Collection(E...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...
language	Edm.String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	标准 - Lucene <input type="text"/>	...

接着我们就来进行一些简单的搜索吧！

终于可以测试一下我们建立的数据库了！搜索浏览器是通过请求URL来运行的，我们输入在查询字符串中的都应当遵循Azure规定的可以使用的查询语言。这里我使用的是简单Azure查询语言，也可以使用 Lucene 等。

### 示例一 查询标题里带MERS的文章



为了搜索标题中带MERS的文章，我们可以编写这样的查询字符串：

```
1 searchFields=title&$select=title&search='MERS'
```

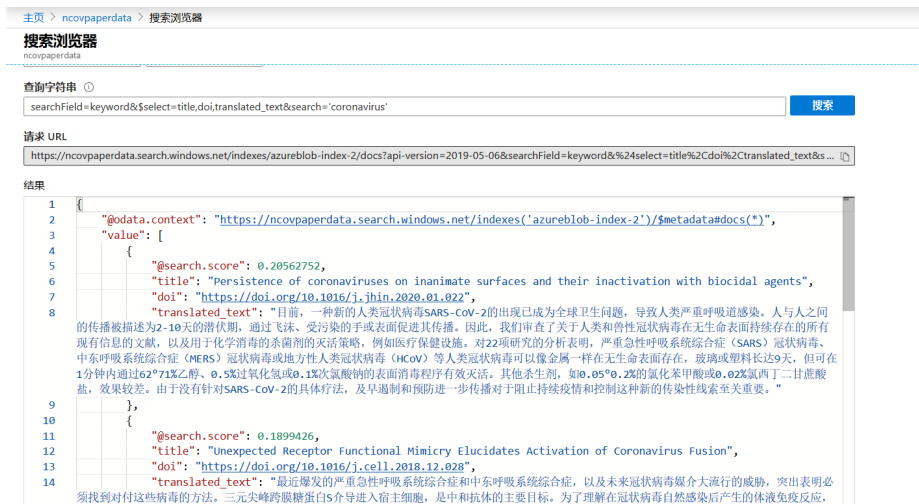
上面这个字符串的意思是，在title(标题)中搜索带有 **MERS** 关键词的数据，并且显示他们的标题。

## 示例二 查询Coronavirus相关的文章并且返回doi以及中文摘要

在这个例子中，为了了解Coronavirus，我们需要与其相关的文章，并且能够简要了解，并下载。这里我们就可以设置查询字符串为：

```
1 searchField=keyword&$select=title,doi,translated_text&search='coronavirus'
```

也就是在 **keyword** 中搜索带有coronavirus关键字的文章，返回他们的标题，doi，以及翻译后的摘要。结果如下：



可以看到，翻译后的文本可读性还是很高的。

## 结语

本数据库的搭建还是非常简易方便的，当然跟已经成型的数据库相比还是相形见绌。实际上，**Azure Cognitive Search**还有更多的有趣的功能，例如与**Power BI**的兼容，可以做出很美观的报告。或者与**AI**兼容，可以在翻译的基础上，加上语义分析等等功能。与**Elastic search**（基于**Lucene**的搜索服务器）相比，这个搜索服务更加的简洁，也更加的方便，还可以添加**AI**功能。当然，一个问题就是他的搜索速度有限，可能不如**elastic search**的分布式搜索。

总而言之，整个建立数据库的流程还是十分顺利的，使用这样的流程也可以建立其他的数据库。在这里也感谢我的导师们给予我的帮助！