NLP Chatbot for CogX Website

MAFS 6010S Project

Outliers:

LI Jiaqi
LIN Tuoyu
LIU Genghuadong
ZHANG Zehao
ZHOU Quan

01 INTRODUCTION

CogX

CogX is a research and development company in applied cognitive science, offering individualized programs, courses and professional development for educators. The website of CogX provides a platform for the public to see how their programs target cognition to enhance learning ability and as well as for the individuals and organizations to partner with CogX.

However, the lack of retrieval function makes it difficult and inconvenient for visitors to filter rich information when browsing the CogX website.

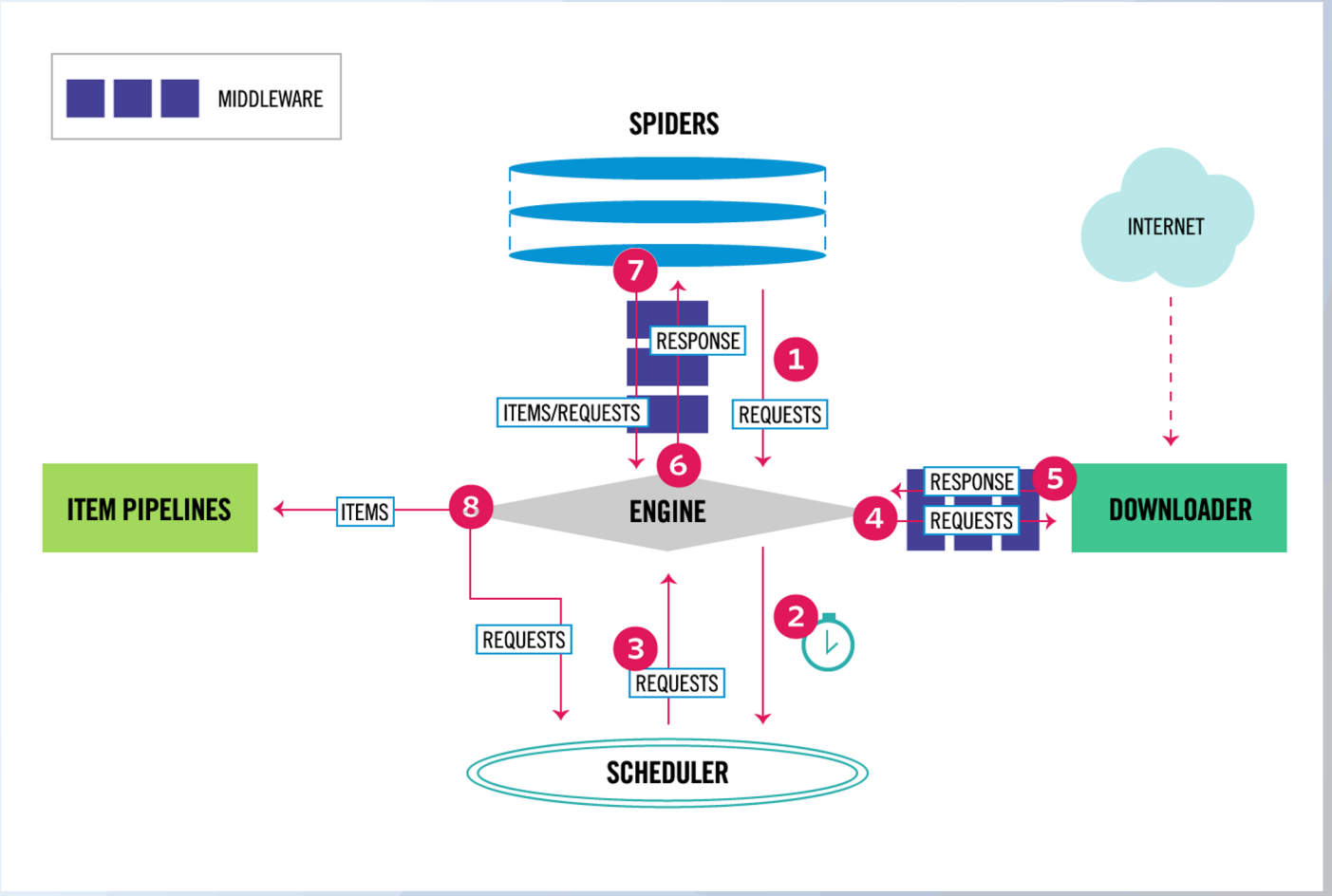
NLP chatbot for the CogX website

- Get information from CogX website and learn knowledge by itself
- Recognize user's question and answer it correctly
- Reply to daily greetings

02 ALGORITHM

Scary

Scrapy is a fast and high-level screen scraping and web scraping framework for Python, used to scrape web sites and extract structured data from pages.



CrawlSpider

CrawlSpider is a derived class of Spider in Scrapy. The design principle of the Spider class is to crawl only the web pages in the *start_url* list. However, the CrawlSpider class defines some rules to provide a convenient mechanism to follow up links and obtain links from the crawled web page results in order to continue to crawl, which greatly simplify the writing of reptiles.

TF-IDF Algorithm

TF-IDF (term frequency–inverse document frequency) is a commonly used weighting technique for information retrieval and text mining.

(1) TF is the term frequency (Term Frequency)

$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where $n_{i,j}$ is the number of times this word appears in file d_j, and the denominator is the total number of times that all words appear in file d_j.

(2) IDF (Inverse Document Frequency)

$$idf_i = log \frac{|D|}{|\{j: t_i \in d_j\}| + 1}$$

where $|D|$ is the total number of files in the corpus, $|\{j: t_i \in d_j\}|$ is the number of files that contained the term t_i.

(3) TF-IDF (TF * IDF)

The high word frequency in a particular file, and the low file frequency of the word in the entire file collection, can produce a high-weight TF-IDF. Therefore, TF-IDF tends to filter out common words and retain important words.

Cosine similarity

Cosine similarity uses the cosine value of the angle between two vectors in the vector space as a measure of the difference between the two objects. The closer the cosine value is to 1, it indicates that the included angle is closer to 0 degrees, that is, the two vectors are more similar, which is called "cosine similarity".

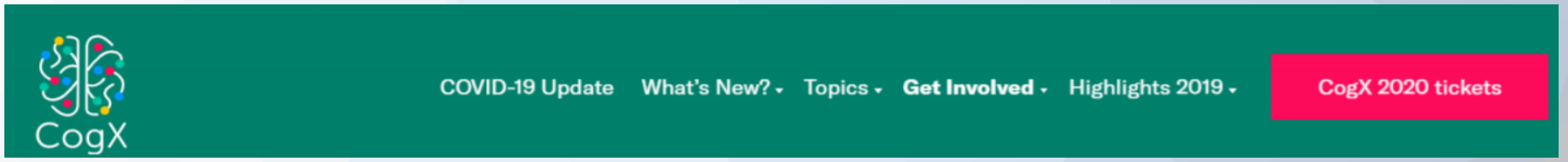
In general, assuming that $\vec{x}$ and $\vec{y}$ are two n-dimensional vectors, then the cosine of the angle between $\vec{x}$ and $\vec{y}$ is equal to:

$$cos\theta = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}} = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||}$$

where $\vec{x} = \{x_1, \ldots, x_n\}$ and $\vec{y} = \{y_1, \ldots, y_n\}$.

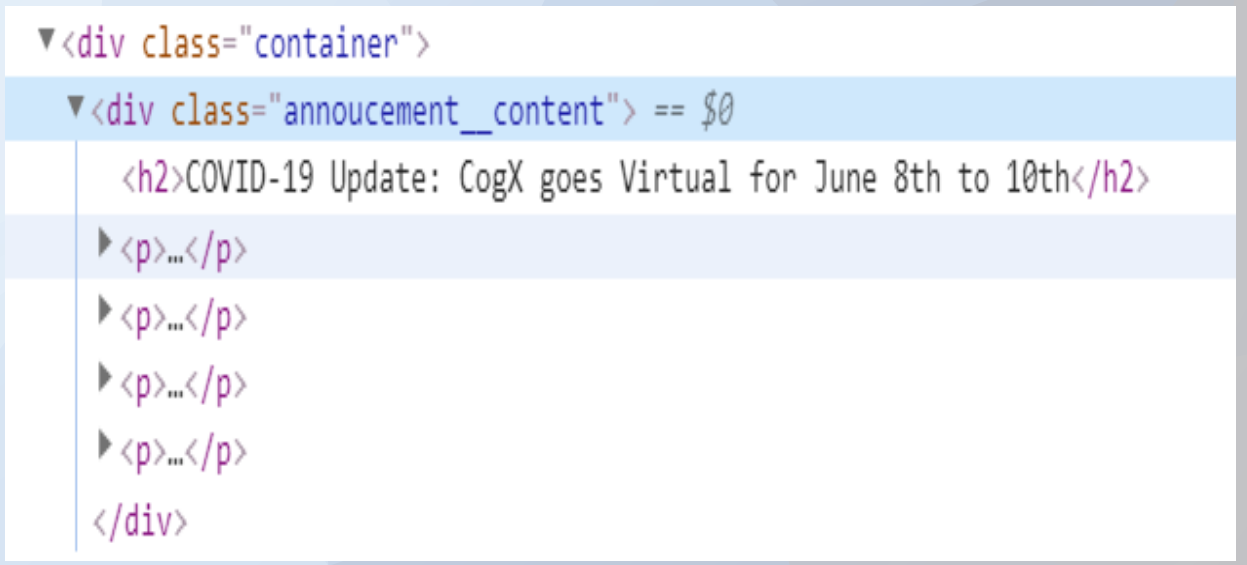
03 EXPERIMENT

Crawl Raw Text Data



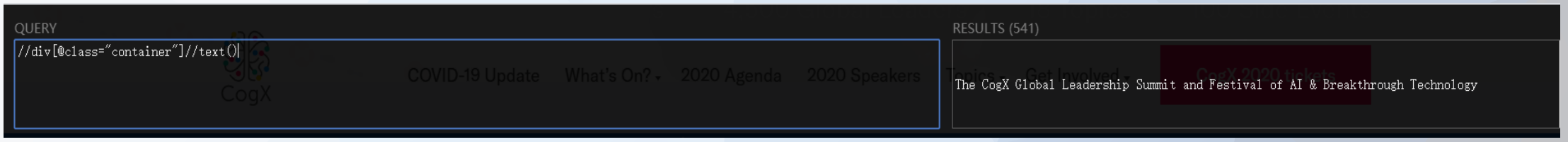
First, we open the homepage of the website. This page includes some summary and introduction of its subsections. We need to store all the information from the home page.

All text content is stored under the "div [@ class = "container"]" tag.



Crawl Raw Text Data

We locate the text content through the xpath.



In this way, we get the text content of the CogX homepage. However, our goal is to make a chatbot that response details about CogX, so we may go further to sub-sections to get details.

Crawl Raw Text Data

We enter the tag of one of the items in the main menu, such as “what ’s on”, we can see its url and the sub-menu urls of its sub-columns.

```
<a href="https://cogx.co/whats-new/">What’s On?</a> == $0
▼<ul class="sub-menu">
 ▶<li id="menu-item-19957" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-19957">…</li>
 ▶<li id="menu-item-19981" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-19981">…</li>
 ▶<li id="menu-item-20032" class="menu-item menu-item-type-post_type menu-item-object-stages menu-item-20032">…</li>
 ▶<li id="menu-item-22398" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-22398">…</li>
 ▶<li id="menu-item-22587" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-22587">…</li>
 </ul>
```

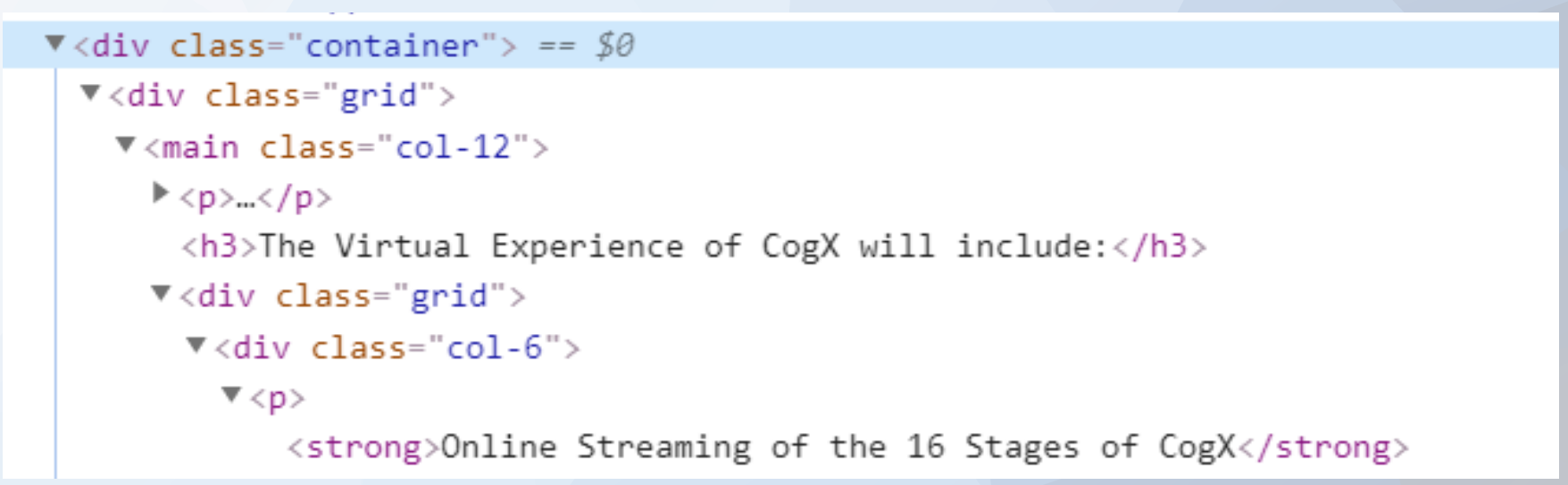
Open one of the <li> tags, we can see the name and url of the sub-menu.

```
▼<li id="menu-item-19957" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-19957">
   <a href="https://cogx.co/virtual-cogx/">Virtual CogX</a>
 </li>
```

Crawl Raw Text Data

After we download the entire url structure of the website, we can enter each sub-menu in turn to get the required text data. The page structure of these sub-menu pages are similar.

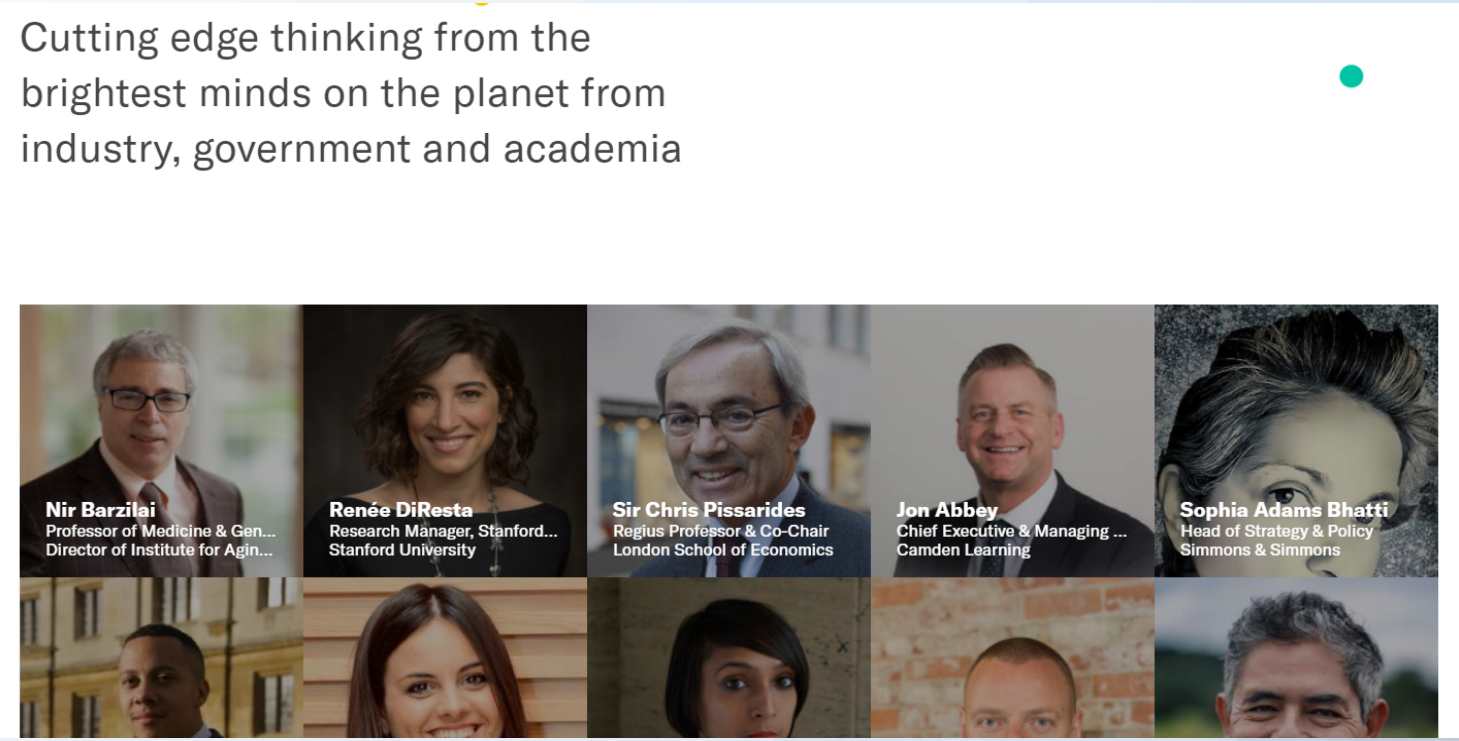
The first one is https://cogx.co/virtual-cogx/. The same as the CogX homepage, the text data of this subpage is contained under the div [@ class = "container"] tag



Therefore, we can download the data of these pages in the same way as the home page.

Crawl Raw Text Data

The second page is https://cogx.co/2020-speakers/



This page lists speakers who will speak at the CogX forum in 2020.

Due to the lack of a box for storing text data in this page, it does not have a “div [@ class = "container"]” tag. However, we can get the text data of this page through the xpath of “// main // text ()”.

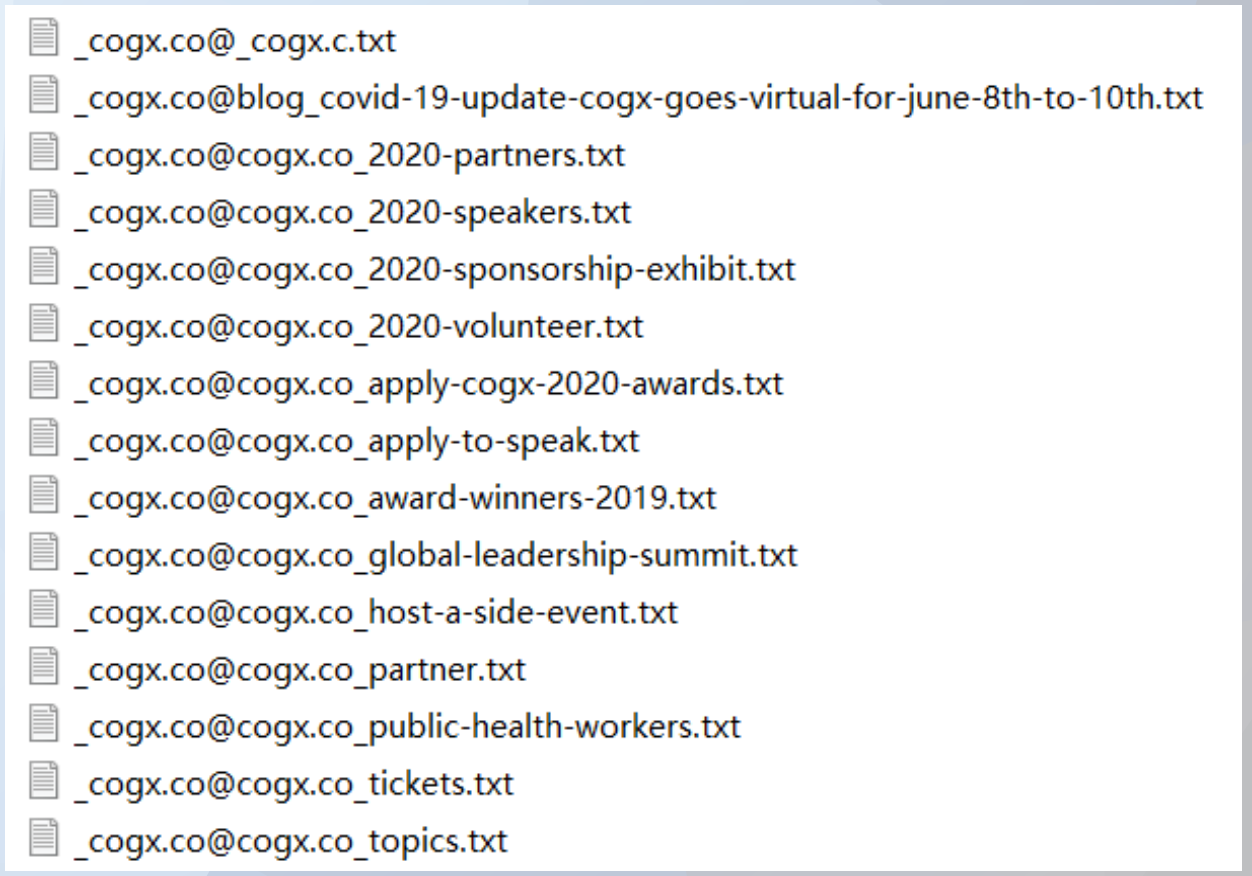
After understanding how this website is organized, it's easy to crawl data from it.

Design of Crawler

Because of the easy structure of CogX website, we only need to use Scrapy module of Python to crawl text data from it, and then save the text data into txt files with corresponding classification.

Since we do not need crawl pictures or videos, it is not likely to be banned for our IP. Thus, we may not need to handle anti-spider mechanism.

The picture is part of text datasets obtained by our crawler.



Creation of Query-Response Database

Since our chatbot is a closed-field robot based on searching and matching methods, we need to process the text data, and organize it into the corpus with the form of query-response pairs.

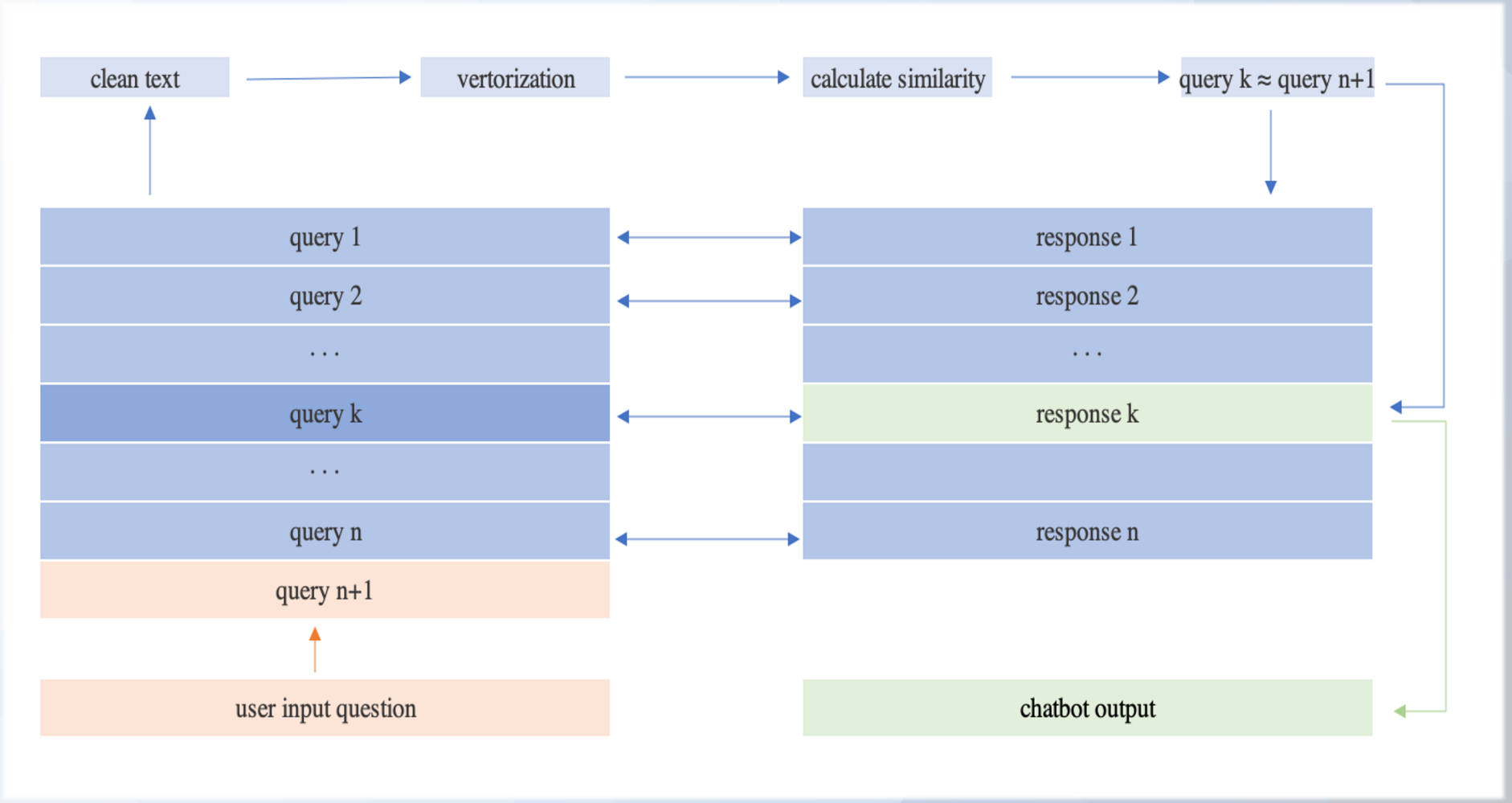
This is necessary because the chatbot based on searching and matching methods is to find the most suitable response from the existing large number of candidate responses.

	question
226	who winners 2019Best AI Product In Education
227	who winners 2019Best AI Product In Entertainment
228	who winners 2019Best AI Product In Fashion
229	who winners 2019Best AI Product in HR
230	who winners 2019Best AI Product In Insurance
231	who winners 2019Best AI Product In Next Generation Infrastructure
232	who winners 2019Best AI Product In Legal
233	who winners 2019Best AI Product In Marketing and Adtech
234	who winners 2019Best AI Product in Retail
235	who winners 2019Best AI Product In Security
236	who winners 2019Best AI Product In Telecom
237	who winners 2019Best Consumer Chatbot
238	who winners 2019Best Innovation In Artificial General Intelligence
239	who winners 2019Best Innovation in Autonomous Vehicles
240	who winners 2019Best Innovation In Chatbots
241	who winners 2019Best Innovation In Computer Vision
242	who winners 2019Best Innovation In Creative Arts
243	who winners 2019Best Innovation In Data Protection And Privacy
244	who winners 2019Best Innovation In Gaming

	response
226	Century Tech
227	Spotify
228	Edited
229	Humanyze
230	Cytora
231	Peltarion
232	Eigen Technologies
233	Codec.ai
234	Yamato
235	Darktrace
236	Voca AI
237	Yamato
238	OpenAI GPT-2
239	Wayve
240	Haptik
241	Skinanalytics
242	OpenAI (MuseNet)
243	Hazy
244	Improbable

The above graphs are part of the query-response pairs we have organized.

Information Match and Auto-reply



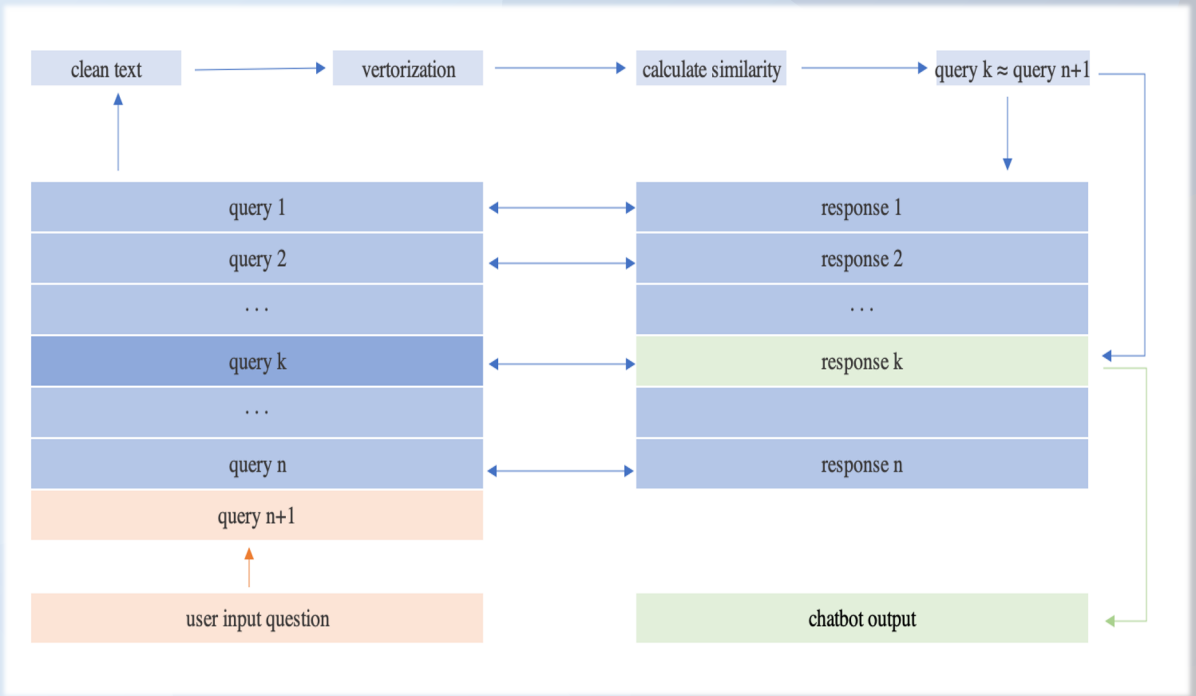
Information Match and Auto-reply

1. Read the string of input question from the user.

2. Load the prepared query database and combine with the input to form a new query database.

3. Run the text cleaning function on each of these queries to perform the text clean.

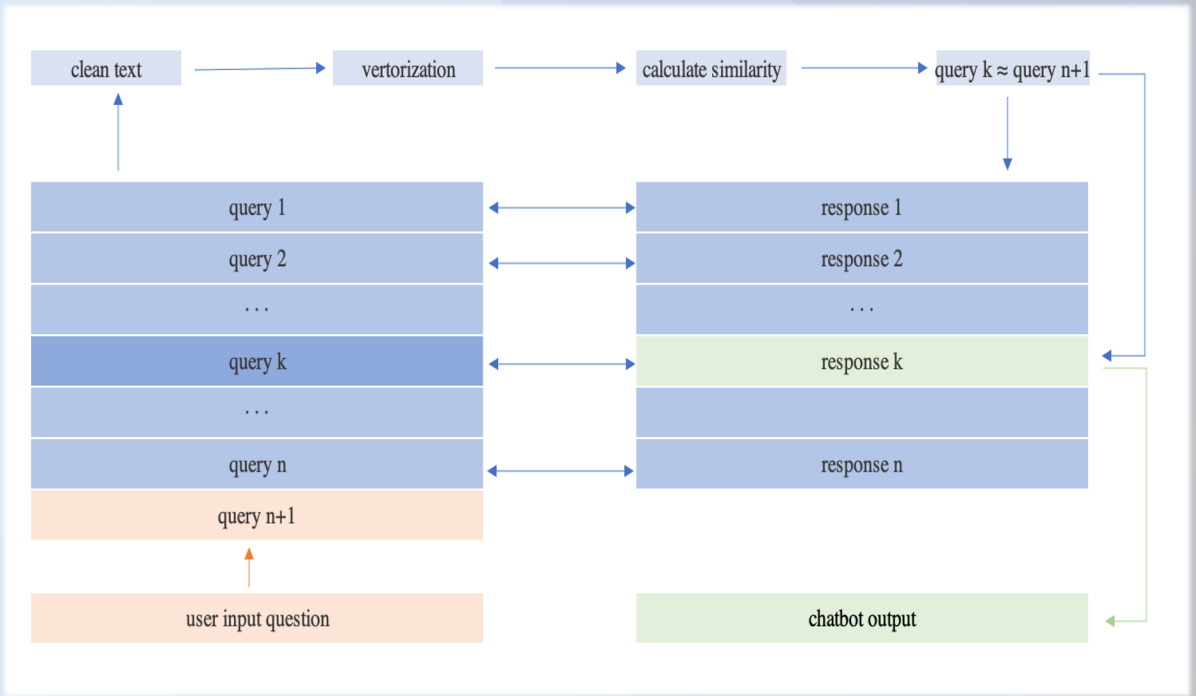
4. Performing lemmatization to standardize all the words in every query and converting all words to lower case.



Information Match and Auto-reply

4. Text vectorization:
Firstly we need to remove stop words. A customized stop word list is used. Then the TF-IDF algorithm is performed.

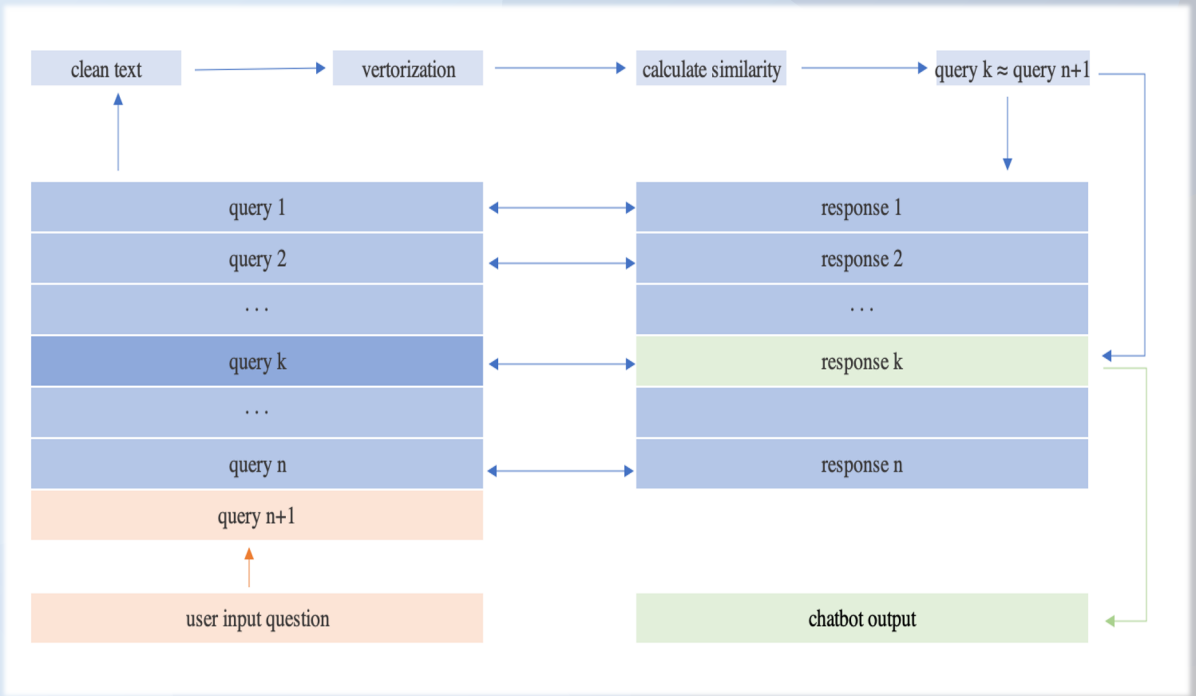
5. Calculate the cosine similarity:
We calculate the cosine similarity between the vectors from original query set and the vector of the input question.



Information Match and Auto-reply

6. Select the index with the largest cosine similarity and retrieve the corresponding answer. If the maximum cosine similarity is still below 0.5, no answer will be returned.

7. Establish a simple learning mechanism and continuously improve the chatbot's problem recognition ability during use.



04 CHATBOT DISPLAY

Functions

This Chatbot basically matches the questions with the response if they have similar relationships.

There is a database of the responses (stored in the chatbot). Every question accordingly matches a response. We set these pairs between question-response in advance.

When a person ask a question, based on the rules of cosine similarity, if this question overlap the questions stored in the chatbot database greater than 50%, we match the answer accordingly to this stored questions. If the similarity is less than 50%, the chatbot will generate the response: “sorry, I don’t know”.

Similarly, we divide the chatbot function to the technical part and the daily chat part.

The great part of the chatbot is that it covers almost all the information/answers in the website. When a person asked the questions in the website, the chatbot will probably response the correct answer. When a person talks about the daily conversation, the chatbot will have the answer prepared before.

Functions

The major drawback is that the chatbot is not smart enough. It is not flexible. We set the range of the answer. The answer is simply limited in the information of the website, about the job, achievement, education background, past experience, etch. Daily conversation is also like this.

If we ask some very flexible questions, like a joke, then this robot cannot generate the appropriate response. In other words, we have strong limitations in the range of the response. This robot is especially designed for answering the questions in the COG X website.

Examples

The coding part: we define many functions to run the chatbot, such as:

read questions:

The chatbot read the input question.

clean text:

Deal with the text clean up and remove the irrelevant information

Word2vec:

Convert the text into vectors after deleting the stop-words.

Cosine similarity

After converting the text into vectors, we use the similarity to match the correlation between the question and the question stored in the database.

In conclusion, we process the question and based on the rules to get the response of the robot

Examples

Especially in Word2Vec:

We only screenshot part of the stopwords. In order to match the question and the response, we have to convert the text into vectors so that we could use cosine similarity to judge if the questions are similar to the questions stored in the bot and generate the response.

Most importantly, in this chatbot design we delete the stop-words. Those words like “a”, “the”, “their” is not important and will affect the matching process.

So we decide to delete this stop words.

Basically, we list the stop words manually, but maybe it still cannot include all the stop words’ situation but it is probably enough.

```
def word2vec(corpus_list):
    '''
    convert the text into vectors after deleting the stop-words
    '''
    stopwords = [
                    "a",
                    "able",
                    "about",
                    "above",
                    "according",
                    "accordingly",
                    "across",
                    "actually",
                    "after",
                    "afterwards",
```

```
                "zero"
            ] + ['ain', 'aren', 'couldn', 'didn', 'doesn', 'doesnt', 'don', 'hadn', 'hasn', 'haven', 'isn', 'll',
                 'mustn',
                 'shan', 'shouldn', 've', 'wasn', 'weren', 'won', 'wouldn']
# stopwords = ENGLISH_STOP_WORDS.union(stopwords)
stopwords = stopwords
# texts is the list of all the questions, and each of the question is a str of selected words
# texts = [' '.join([word for word in doc.lower().split() if word not in stopwords]) for doc in corpus_list]

# tf-idf
vect = CountVectorizer(stop_words=stopwords).fit(corpus_list)
feature_names = vect.get_feature_names()  # words
# print(feature_names)
trans_vect = vect.transform(corpus_list)
tfidf = TfidfTransformer().fit_transform(trans_vect)
max_value = tfidf.max(axis=0).toarray().ravel()
sorted_by_tfidf = max_value.argsort()
# print([feature_names[i] for i in sorted_by_tfidf[-40:]])
# print(tfidf.toarray())
return tfidf
```

Chatbot Display

Output

The first output is the mixture response of the daily conversation and the conversation related to CogX.

```
You:
Hello!
CogBo:
Hi! Welcome to CogX. I'm CogBo, created by the Outliers from HKUST.

You:
Who is outliers?
CogBo:
Outliers, a team of cool boys and girls. Members of Outliers are LIN Tuoyu, LI Jiaqi, LIU Genghuadong, ZHOU Quan, ZHANG
Zehao, they gave me birth and teach me how to communicate with you like this.

You:
What can you do?
CogBo:
I like watching news abour CogX and answering your questions!

You:
Ok, now tell me something about cogx.
CogBo:
Interesting question. You may find what you need at https://cogx.co

You:
What is cogx?
CogBo:
The CogX Global Leadership Summit and Festival of AI & Breakthrough Technology is the biggest, most inclusive & forward
thinking gathering of leaders, CEOs, entrepreneurs, policy makers, artists, academics and activists of its kind - addres
sing the question "How do we get the next 10 years right?"
```

Chatbot Display

Output

The second and the third picture is the inquiry about the information in the COG X website.

```
You:
When will cogx be held?
CogBo:
Monday 8th – Wednesday 10th June 2020.

You:
Introduce the topic of space, thx
CogBo:
Interesting question. You may find what you need at https://cogx.co

You:
How many topics are there in cogx
CogBo:
There are 18 topics:
1. Leadership
2. The Cutting Edge
3. Lab to Live _ Case Studies
4. Ethics & Society
5. Economy & Future of Work
6. The HR and Ed Tech Revolution
7. The Planet and Smart-Cities
8. Space
9. Start-up to Scale-Up to IPO and Beyond
10. Createch
11. Health, Wellbeing & COVID-19
12. Research _ The Long View
13. Web 3.0 & Decentralisation
14. Cyber & Defence
15. Gen Z
16. FinTech & Future of FS
17. Next Gen Infrastructure & Cloud
18. Industry 4.0 & Sustainable Supply Chains
```

```
You:
Tell me about the topic of Leadership.
CogBo:
Focusing on the biggest questions of our time, leaders set our direction of travel

You:
What is the topic of Space talking about?
CogBo:
Get the latest in space exploration, innovation and astronomy thinking as we celebrates humanity's ongoing expansion ac
ross the final frontier and both look to the future and discuss what tangible learnings we can take from this research t
o todays challenges.

You:
Who are the speakers of the topic Web 3.0 & Decentralisation?
CogBo:
Addressing global challenges with distributed technology

You:
Who are the speakers of Cyber & Defence?
CogBo:
Mariarosaria Taddeo, Stuart Russell,  Husayn Kassai, Anthony Finkelstein, Grace Cassy, and Ivana Bartoletti

You:
Introduce Stuart Russell.
CogBo:
Professor of Computer Science UC Berkeley
```

05 CONCLUSION

Achievement

As We expected, we try to build a task-oriented Chatbot that could automatically reply *CogX*-related questions. Based on the information got from the Internet and the recognition of the question, the Chatbot could provide corresponding information that is requested in the question.

Based on the functions we designed at last, the Chatbot would help those interested in *CogX* in such ways:

- Reduce the time lag of information acquisition.
- Simplify the steps of getting information about specific issues.

Future Improvement

Corpora Pretreatment

After scrabbling of the website, we spent a lot time on the pretreatment of the corpora manually, which is not as we expected. In this way, we would have to do similar thing every time the website is updated, which is quite inefficient.

To solve this problem, we should design some rules to treat the corpora. For example, we could design some rules to extract the important information from the articles and sentences. And the type of the information should be marked. For the sentence below:

'The CogX would be held at Monday 8th - Wednesday 10th June 2020 virtually.'

The information in the sentence should be recognized as below:

Type	Information
Activity	CogX
Time	Monday 8th - Wednesday 10th June 2020
Method	virtually

Future Improvement

Memory

Now the chatbot do not have memory. In some situation, the user may ask a question that is related to the answer of the last question. For example:

-Q1:How much is the Festival Pass?
-A1: £295
-Q2:What does it include?
-A2:……

In this situation, 'it' in the second question actually refers to the 'Festival Pass', but the chatbot could not recognize without any memory. To solve this problem, we may save the information from the last question and answer, when words like it/that/them appears in the next question, corresponding words from the last question would be added to this question.

Future Improvement

Answer Generating

Answers provided by the chatbot are all written in advance. In this case, the chatbot cannot generate the answer that the users need. Or sometimes the chatbot may reply a lot of sentences but only one of them contains the useful information.

To solve this problem, we may try using a generative chatbot instead of matching chatbot now. We expect that the chatbot would use least sentences to answer the question. But in this way, we may try to teach the chatbot how to assemble sentences using the words that are extracted as useful information.

Future Improvement

Similarity Judgment

Now we use the tf-idf to judge the similarity between the users' question and the questions we set in advance. But there are some disadvantages for this method.

Tf-idf could not recognize synonyms and near synonyms. This may lead to the situation that the user's question could not be matched when synonyms are used in question.

When using the tf-idf, we ignored the sequence of the words and the meaning of the words. Only the frequency of the words is used to do matching. In this way, the accuracy and efficiency of the match would be lower.

To solve the problem, we could try using the algorithm of dual LSTM instead.

THANK YOU!