# NLP Chatbot for CogX Website

LI Jiaqi 20659572
LIN Tuoyu 20652809
LIU Genghuadong 20640870
ZHANG Zehao 20665313
ZHOU Quan 20659754

## 1. INTRODUCTION

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.

The goal of this paper is to design an NLP chatbot for the CogX website. We want to design a chatbot that get information from CogX website and learn knowledge by itself. When user asks some question related to the predefined information, the robot will recognize the key word and answer the corresponding question. When the user just says: “Hello” or other greetings, it will also give simple responses. In this way, the chatbot can provide the exact information and guide visitors to the corresponding webpage efficiently, saving time at the same time. On the other hand, the user's conversation with the chatbot will be recorded, helping CogX to analyze which topics are the most popular or which events visitors prefer to get involved so that they can optimize the website structure.

## 2. THEORY

### 2.1 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. Scrapy is versatile and can be applied in many areas, including data mining, monitoring and automated testing. With the power of the framework, users only need to customize and develop a few modules to easily implement a spider, which is used to grab webpage content and various pictures conveniently.

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.

The overall crawling process of CrawlSpider is illustrated as below:

a). The crawler file first obtains the webpage content of the url according to the starting url.
b). The link extractor will extract the links in the web content in step (a) according to the specified extraction rules.
c). The rule parser will parse the webpage content in the link extracted from the link extractor according to the specified parsing rule according to the specified rule.
d). Encapsulate the parsed data into the item and submit it to the pipeline for persistent storage.

### 2.2 TF-IDF Algorithm

#### 2.2.1 Introduction of TF-IDF algorithm

TF-IDF (term frequency–inverse document frequency) is a commonly used weighting technique for information retrieval and text mining. TF-IDF is a statistical method used to evaluate the importance of a word to a document in a document set or a corpus. The importance of a word increases proportionally with the number of times it appears in the document, but at the same time it decreases inversely with the frequency of its appearance in the corpus. The main idea of TF-IDF is that if a word appears in an article with a high frequency TF and rarely appears in other articles, it is considered that the word or phrase has a good ability to distinguish between categories, suitable for classification.

(1) TF is the term frequency (Term Frequency)
Word frequency (TF) indicates how often an entry (keyword) appears in the text. The number is usually normalized (generally the word frequency divided by the total number of words in the article) to prevent it from biasing towards long documents.

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

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)
The IDF of a particular word can be obtained by dividing the total number of files by the number of files containing the word, and then obtaining the logarithm of the obtained quotient. If the number of documents containing the term t is smaller, the IDF is larger, indicating that the term has a good ability to distinguish categories.

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

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.

### 2.2.2 TF-IDF application

(1) Search engine; (2) Keyword extraction; (3) Text similarity; (4) Text summary

## 2.3 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 $\vec{y} = \{y_1, \ldots, y_n\}$.

Cosine similarity is most commonly used in high-dimensional positive spaces. For example, in information retrieval, each term is assigned a different dimension, and a dimension is represented by a vector, and the value in each dimension corresponds to the frequency of the term appearing in the document. The cosine similarity can therefore give the similarity of the two documents in terms of their subject. Also, it is usually used for file comparison in text mining. In addition, in the field of data mining, it will be used to measure the cohesion within the cluster.

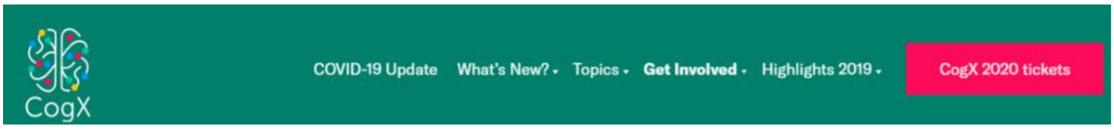
To be specific, the process of text similarity calculation based on cosine similarity is:
a). Find the keywords of two articles.
b). Take several keywords for each article and combine them into a set to calculate the word frequency of each article for the words in this set.
c). Generate word frequency vectors of two articles.
d). Calculate the cosine similarity of two vectors. The larger the value, the more similar it is.

# 3. EXPERIMENT

## 3.1 Data Acquire

### 3.1.1 Crawl Raw Text Data

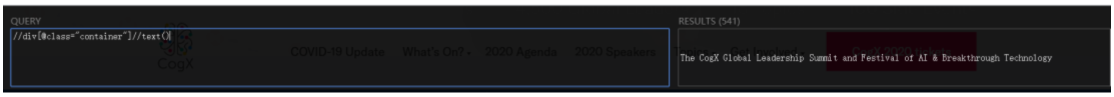
As mentioned in the previous section, we use the Scrapy framework to crawl the entire site of cogx.co. The first step in writing a crawler is to understand the architecture of the target website.




```
▼<div class="container">
  ▼<div class="annoucement__content"> == $0
     <h2>COVID-19 Update: CogX goes Virtual for June 8th to 10th</h2>
    ▶<p>…</p>
    ▶<p>…</p>
    ▶<p>…</p>
    ▶<p>…</p>
   </div>
```

First we open the homepage of the website. *https://cogx.co/* is the URL of the frontpage of CogX. This page includes some summary and introduction of its subsections. We need to store all the information from the home page. Through the element check, we found that the main elements of the page are shown in the right side figure.

In the above webpage structure, all text content is stored under the “div [@ class = "container"]” tag. Therefore, we locate the text content through the xpath.




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.

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 .

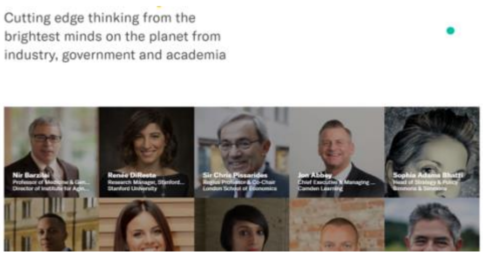
```
<a href="https://cogx.co/whats-new/">What’s On?</a> == $0
▼<ul class="sub-menu">
  ▶<li id="menu-item-19957" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-19957">…</li>
  ▶<li id="menu-item-19981" class="menu-item menu-item-type-post_type menu-item-object-page menu-item-19981">…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

```
▼<div class="container"> == $0
  ▼<div class="grid">
    ▼<main class="col-12">
      ▶<p>…</p>
       <h3>The Virtual Experience of CogX will include:</h3>
      ▼<div class="grid">
        ▼<div class="col-6">
          ▼<p>
             <strong>Online Streaming of the 16 Stages of CogX</strong>
```

Here we can download the entire url structure of the website. Then we can enter each sub-menu in turn to get the required text data. The page structure of these sub-menu pages are similar, we will explain how to crawl these text data with two example pages. 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.




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, we've seen how the site was built, and it's easy to crawl data from it.

### 3.1.2 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 text datasets obtained by our crawler are shown as below.

_cogx.co@_cogx.c.txt
_cogx.co@blog_covid-19-update-cogx-goes-virtual-for-june-8th-to-10th.txt
_cogx.co@cogx.co_2020-partners.txt
_cogx.co@cogx.co_2020-speakers.txt
_cogx.co@cogx.co_2020-sponsorship-exhibit.txt
_cogx.co@cogx.co_2020-volunteer.txt
_cogx.co@cogx.co_apply-cogx-2020-awards.txt
_cogx.co@cogx.co_apply-to-speak.txt
_cogx.co@cogx.co_award-winners-2019.txt
_cogx.co@cogx.co_global-leadership-summit.txt
_cogx.co@cogx.co_host-a-side-event.txt
_cogx.co@cogx.co_partner.txt
_cogx.co@cogx.co_public-health-workers.txt
_cogx.co@cogx.co_tickets.txt
_cogx.co@cogx.co_topics.txt

### 3.1.3 Creation of Query-Response Database

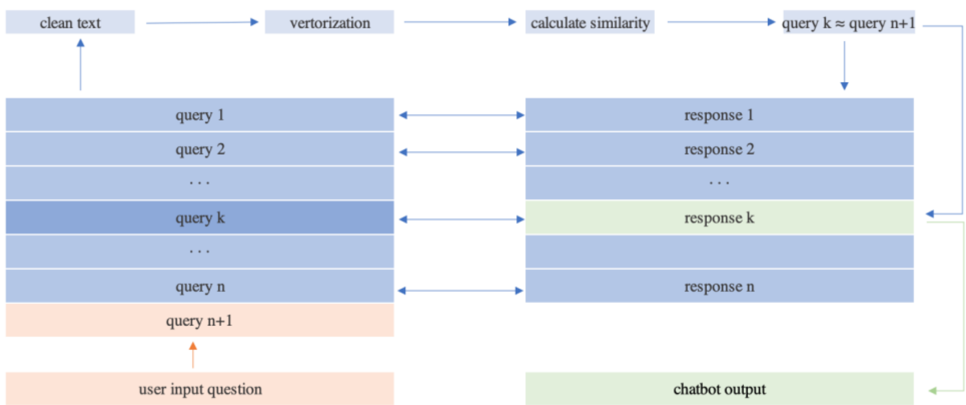
Since our chatbot is a closed-field robot based on searching and matching methods, we need to manually process the text data that has been obtained, and organize it into the corpus with the form of query-response pairs. This is necessary because the chatbot based on searching and matching methods is to find the most suitable response from the existing large number of candidate responses by searching and matching. The following are part of the query-response pairs we have organized.

| | 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 |

After processing the text data, we can start to write chatbot's algorithm.

## 3.2 Information Match and Auto-reply




a). Read the string of input question from the user.

b). Read the csv file of the query database and combine it with the input question to form a new query database.

c). Run the clean_text function on each of these queries to perform the text cleanup, which includes:

Removing special symbols, addresses, urls and other irrelevant information according to regular expressions;

Performing lemmatization to standardize all the words in every query to the corresponding word roots, which can reduce the feature dimension and improve the matching accuracy;

Converting all words to lower case.

d). Text vectorization:

Firstly, we need to remove stop words. A customized stop word list is used, because the traditional stop word list removes words such as “when”, “who”, “where”, which may express key information in chatbot. Thus, the traditional stop word list cannot be used directly.

Then the TF-IDF algorithm is used to vectorize all the cleaned corpus, which contains the input query.

e). Calculate the cosine similarity:

The previous step has obtained vectors that can represent all the queries. Then we calculate the cosine similarity between the vectors from original query set and the vector of the input question. Next we record the corresponding index of every cosine similarity value.

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

g). If the newly entered question is successfully matched to an answer, the newly entered question will be used to update the corresponding original question, so as to establish a simple learning mechanism and continuously improve the chatbot's problem recognition ability during use. The approach is as following:

Combine the string of the original question and the string of the new question into a new string, then split this string into a list. Next, convert the list into a set in order to deduplicate the key words. Finally, use the set to generate the new query string, that is, the updated question.

# 4. CHATBOT DISPLAY

## 4.1 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 user asks a question, according to the cosine similarity rule, if the question overlaps by more than 50% with the question stored in the chatbot database, we will match the answer to the stored question accordingly. If the similarity is less than 50%, the chatbot generates a response: "sorry, I don't know".

Similarly, we divided the functions of the chatbot into the technical part and the routine part. The biggest advantage of chatbot is that it can cover almost all the information/answers on the website. When users ask a question on the website, the chatbot will response the correct answer. When users talk about daily conversation, the chatbot will give the answer preset before. However, the drawback is that the chatbot is not smart enough, which means 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 because this robot is especially designed for answering questions on the CogX website.

## 4.2 Example

### 4.2.1 Introduce Part of the Coding

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

a). read questions:
The chatbot read the input question.
b). clean text:
Deal with not standardized letters and standardize the text format.
c). Word2vec:
Convert the text into vectors after deleting the stop-words.

```
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
```

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.

d). Cosine similarity

```
def cosine_similarity(x, y, norm=False):
    assert len(x) == len(y), "len(x) != len(y)"
    zero_list = [0] * len(x)
    if x == zero_list or y == zero_list:
        return float(1) if x == y else float(0)
    res = np.array([[x[i] * y[i], x[i] * x[i], y[i] * y[i]] for i in range(len(x))])
    cos = sum(res[:, 0]) / (np.sqrt(sum(res[:, 1])) * np.sqrt(sum(res[:, 2])))
    return 0.5 * cos + 0.5 if norm else cos
```

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.

#### 4.2.2 output of the chatting

The output of the mixture response about the daily conversation and the conversation related to CogX is shown in the Appendix 1.

# 5. CONCLUSION

## 5.1 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.

## 5.2 Future Improvement

### 5.2.1 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 |

In this way, the corpora could save the information we need. It would save a lot time if this process can be done automatically.

### 5.2.2 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, words from last question would be added to this question.

### 5.2.3 Similarity Judgment

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.

## Appendix 1

```
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?"
```

The first output is the mixture response of the daily conversation and the conversation related to COG X information.

```
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
```

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

## Appendix 2

Link of presentation video:

https://www.youtube.com/watch?v=9vyOJiV6BsM

Link of codes collection:

https://github.com/GenghuadongLIU/HKUST-MAFS-6010U-Project/

Group-member's contribution:

LI Jiaqi: corpus processing, report writing and typesetting, code and PPT validation

LIN Tuoyu: coding, report writing and PPT validation

LIU Genghuadong: corpus processing, report writing, code validation, PPT drawing

ZHANG Zehao: corpus processing, report writing, code and PPT validation

ZHOU Quan: corpus processing, report writing, code and PPT validation

## References

[1] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al.2016. Smart reply: Automated response suggestion for email. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 955–964.
[2] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In NAACL-HLT, 2018.
[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, 2017.
[4] Chongyang Tao, Wei Wu, Can Xu, Wenpeng Hu, Dongyan Zhao, and Rui Yan. 2019b. One time of interaction may not be enough: Go deep with an interaction-over-interaction network for response selection in dialogues. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1–11.
[5] Feng-Lin Li, Minghui Qiu, Haiqing Chen, Xiongwei Wang, Xing Gao, Jun Huang, Juwei Ren, Zhongzhou Zhao, Weipeng Zhao, Lei Wang, et al. 2017. Alime assist: An intelligent assistant for creating an innovative e-commerce experience. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, 2495–2498.
[6] Haifeng Wang. 2016. Duer: Intelligent Personal Assistant. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. ACM, 427–427.
[7] Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 372–381.
[8]Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 285–294.
[9] Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 496–505.
[10] Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 1118–1127.
[11] *https://engineering.linkedin.com/blog/2017/10/building-smart-replies-for-member-messages*
[12] *https://zhuanlan.zhihu.com/p/83825070*
[13] *https://stackoverflow.com/questions/34514286/auto-reply-from-the-use-of-a-predefined-word-list*
[14] *https://blog.csdn.net/xuezhangjun0121/article/details/84862486*