INTELLIGENT (TASK-ORIENTED) CONVERSATION ASSISTANT FOR COURSE SELECTION

Project Progress

THE UNIVERSITY OF SYDNEY

Information Technology Capstone Project

COMP5703/5707/5708

Group Members

1. Shengyuan Sun (460257820)
2. BiYing Wang (470067400)
3. Rui Chen (470111585)
4. Quan Chen (470199228)

TABLE OF CONTENTS

Table of Contents

1. PROGRESS & ACHIEVEMENTS

1. Summary of Progress

In this section, first we will list the four major modules of the Cassandra project, and explain the specific progress and unfinished parts of each part.
The Cassandra project consists of four parts, natural language understanding, dialogue log, dialogue management and natural language generation. In addition, because of lacking training set, there are additional training data generation tasks.

For the training data generation, the method is to generate and manually label the label through the template. The members will design the template by the confirmed intent, and generate the training data by replacing the keywords with the unified generation format. Currently there are 26 intents and 30000 training datas.

The first main part of the project is natural language understanding, which includes text preprocessing, intent classification, and slot filling. In text preprocessing, we used three steps which are convert to lower case, replace digital word to number, remove stop word and POS tagging for analysis by using the NLTK package. In order to correlate the input data, we use the common word vector from wiki-news to perform word embedding on the word. In the current design, we use the seq2seq model to represent the text meaning. Intent classification and slot filling share the same BI-LSTM encoder. The intent classification is to directly use an argmax layer to output the intent, and the slot filling decoder is obtained through the attention mode. Finally, there is string matching to identify the entity, we have used a predefined entity dictionary to match the entities in the input.

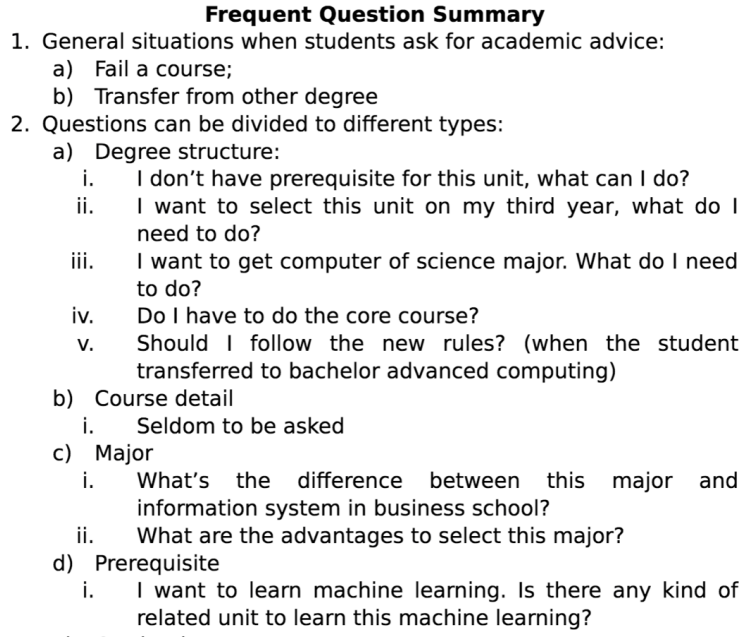
The second main body of the project is the dialog log system. The current log system of the project is mainly used to record all the information between the user and the chatbot, including the slot, intent, and complete sentences input from the user. Record the query, answer and answer template of the chatbot selection.

The third main part of the project is dialogue management. In this part, the members design a relational database by analyzing the structure of CUSP. The fuzzy query can match related information and define the middle from intent, slot and entity to query. Logic, which ultimately returns the answers the user needs from the database.

The last part is natural language generation. At present, our design is based on the template generation method, and then the grammar logic merges the templates to get the final output.

1.2 Raw Question Generation

In Cassandra project architecture, the model should be trained by concrete data that is labelled by human. The training dataset should include thousands of sentences, which could be represented by the combination of raw questions, slots that are annotated by hand and the intents which also are marked by our team members. However, the problems we faced at the beginning of the project are we do not have any raw questions that should be collected from actual interviews or questionnaires. We have interviewed the librarian or receptionists and they have provide some Course & Unit of Study Portal (CUSP) information that our team did not know before.



Frequent Question Summary

1. General situations when students ask for academic advice:
   a) Fail a course;
   b) Transfer from other degree
2. Questions can be divided to different types:
   a) Degree structure:
      i. I don't have prerequisite for this unit, what can I do?
      ii. I want to select this unit on my third year, what do I need to do?
      iii. I want to get computer of science major. What do I need to do?
      iv. Do I have to do the core course?
      v. Should I follow the new rules? (when the student transferred to bachelor advanced computing)
   b) Course detail
      i. Seldom to be asked
   c) Major
      i. What's the difference between this major and information system in business school?
      ii. What are the advantages to select this major?
   d) Prerequisite
      i. I want to learn machine learning. Is there any kind of related unit to learn this machine learning?

Figure 1 Frequently Asked Questions

The Figure 1 displays a list of questions that we summarized from interviewing the reception staff. This screenshots show part of the question list that we got. For example, most frequently asked questions includes "What is the difference between this major and information system in business school". This questions involve a general questions that tasked oriented system could not be easy to answer that. The difference that between general questions and domain-limited questions are general questions could be answered without training or obtaining specific domain knowledge. For this reason, some too general questions was ignored by our system when we initially design our system. Therefore, the solutions that our team comed up with have two methods. One is dependency parsing to help generating raw questions, and another method our team got was using the traditional template method to generate questions.

1.2.1 Dependency parsing

Dependency parsing is a method that extracts dependency parse structure of sentences. According to Clark, dependency parsing could achieve as high as 97.74 % in POS and 96.61% in UAS evaluation (Clark et al., 2018). The dependency parsing

method could help our team finding the root, some keywords which is called head and its modifier. This structure could enable us to easily find the the structure of sentences' grammar and entity. Then we could bring the ideas that generated from the result of dependency parsing to generate our template which could represent more normal questions.

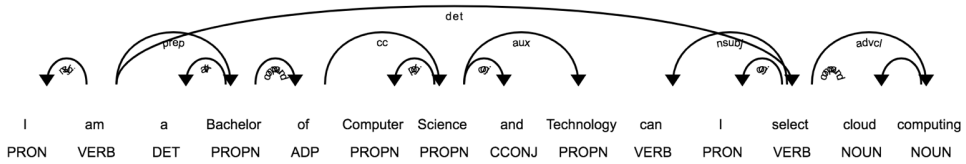


Figure 2 Dependency parsing sentence

This figure shows what we have done in terms of the part of dependency parsing for a raw questions. The part of speech (POS) part has been done automatically by calling a name library called "Spacy". The labels at the bottom of this figure is the result of POS tagging. Meanwhile the bridge lines above the raw questions are dependency relations which normally start from a head to point its modifier. This dependency part could enlighten us to design our slot based question templates.

The implementation of this part is that we firstly collect thousands of general questions from websites, and almost all of these general questions come from www.tagquestions.net.

744	Most memorable Holiday moment?
745	Your favorite holiday decorations
746	Have you ever been caught under a mistletoe?
747	Hands down, what's your all-time favorite holiday food and holiday sweet treat?
748	What did you do in last year that you've never done before?
749	What was last year's New Year Resolution, how did you do?
750	What would you like to have accomplished last year that you didn't get to?
751	What was your biggest achievement of the year?
752	What was the best thing you bought?
753	Did you travel anywhere memorable?
754	Tell us a valuable life lesson you learned last year.
755	If you could look back on one memory this year you wish you could relive, what would it be?
756	If you could change anything about the year you had, what would it be?
757	Share something you're looking forward to this year.
758	Where did you meet?
759	Where was your first date?
760	What was your first impression of each other?
761	How long have you been together?

Figure 3 General questions

The figure 3 demonstrates a list of general questions that we collected online. The domains are general rather than course related. The purpose of this we did is that the parsing result of this general questions could guide us to generate template in more detail. Therefore, we have done dependency parsing for this question list. The figure 4 will shows these dependency parsing structures.

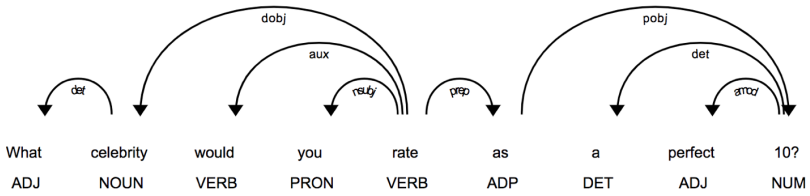


Figure 4 General question dependency parsing

The outcome of this part of workload is we have thousands of this questions structure which we could use to generate the questions templates.

1.2.2 Question Templates

The question templates are the only solution that we could use in this stage to solve the shortage of training dataset. This project we are doing do not have questions set like the reviews in forum, Facebook or twitter. Therefore, collecting data or creating our training dataset becomes the hardest part of our project. After finishing the dependency parsing, we basically grasped the structure of questions that general users who frequently ask no matter they are related course selection domain or not.

We have design more than one hundred questions templates with some blank that should be filled with the data in our database. These blanks will be labelled with slots name which indicates what kind of entities could replaced the blank. These blanks actually be defined by the format "#1 slot-A; #2 slot-B". Therefore, after all the templates are clearly defined, these hash symbols and its combined numeric symbol would be replaced by one of the content in the slot list like "BOS I am a (#1) can I select (#2) EOS". The Figure below shows an entity list that could match the slots "B-CourseID" and "CourseName".

```
1   CourseID,CourseName
2   COMP5206,Information Technologies and Systems
3   INFO5990,Professional Practice in IT
4   INFO5992,Understanding IT Innovations
5   INFO6007,Project Management in IT
6   CISS6022,Cybersecurity
7   COMP5045,Computational Geometry
8   COMP5046,Natural Language Processing
9   COMP5047,Pervasive Computing
10  COMP5048,Visual Analytics
11  COMP5216,Mobile Computing
12  COMP5313,Large Scale Networks
13  COMP5318,Machine Learning and Data Mining
14  COMP5328,Advanced Machine Learning
15  COMP5329,Deep Learning
16  COMP5338,Advanced Data Models
17  COMP5347,Web Application Development
18  COMP5348,Enterprise Scale Software Architecture
```

Figure 5 Entity List or Slot

We have done the randomly replacement operation that replace the slots with actual entities that in our database to form the raw questions. This method seemly mechanical, but it was the only way we could implement.

We have done the randomly replacement operation that replace the slots with actual entities that in our database to form the raw questions. This method seemly mechanical, but it was the only way we could implement.

	Intent	Table Name	Slots (#1;#2;#3;#4;#5;#6;...)	Question Template[0,0,0,#1,0,0,#2)
2	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseCode	BOS I am a (#1) can I select (#2) EOS
3	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseName	BOS I am a (#1) can I select (#2) EOS
4	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseName	BOS I am a (#1) can I select course (#2) EOS
5	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseName	BOS I am a (#1) can I select unit (#2) EOS
6	In_Related_Course	unitsCourseRelation	#1 B-units.courseCode;#2 B-courseRelations.course	BOS I want to know whether I can select (#1) I am a (#2) student EOS
7	In_Related_Course	unitsCourseRelation	#1 B-units.courseName;#2 B-courseRelations.course	BOS I want to know whether I can select (#1) I am a (#2) student EOS
8	In_Related_Course	unitsCourseRelation	#1 B-units.courseName	BOS Could I select (#1) EOS
9	In_Related_Course	unitsCourseRelation	#1 B-units.courseCode	BOS Could I select (#1) EOS
10	In_Related_Course	unitsCourseRelation	#1 B-units.courseCode	BOS Could you tell me if I can select (#1) EOS
11	In_Related_Course	unitsCourseRelation	#1 B-units.courseName	BOS Could you tell me if I can select unit (#1) EOS
12	In_Related_Course	unitsCourseRelation	#1 B-units.courseCode;#2 B-courseRelations.course	BOS Is (#1) in (#2) course structure EOS
13	In_Related_Course	unitsCourseRelation	#1 B-units.courseName;#2 B-courseRelations.course	BOS Is (#1) in (#2) course structure EOS
14	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseCode	BOS Does (#1) include (#2) EOS
15	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseName	BOS Does (#1) include (#2) EOS
16	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseCode;#3 B-year	BOS Does (#1) offer (#2) (#3) EOS
17	In_Related_Course	unitsCourseRelation	#1 B-courseRelations.course;#2 B-units.courseName;#3 B-year	BOS Does (#1) offer (#2) (#3) EOS
18	In_Related_Course	unitsCourseRelation	#1 B-units.courseName;#2 B-year	BOS Could I select (#1) in (#2) EOS
19	In_Related_Course	unitsCourseRelation	#1 B-units.courseCode;#2 B-year	BOS Could I select (#1) in (#2) EOS
20	In_Related_Course	unitsCourseRelation	#1 B-units.courseName;#2 B-year;#3 B-courseRelations.course	BOS Can I choose (#1) (#2) I am a (#3) student EOS
21	In_Related_Course	unitsCourseRelation	#1 B-units.courseCode;#2 B-year;#3 B-courseRelations.course	BOS Can I choose (#1) (#2) I am a (#3) student EOS

Figure 6 Raw Question Templates

The final output of this raw questions is that we have generated thousands of questions that shown in figure 7.

```
267404  267402,I want to know whether I can select IT Advanced Topic A I am a Graduate Diploma in Information Technology Management
        student,['B-courseRelations.course' 'B-units.courseCode' 'B-year'],In_Related_Course
267405  267403,Does Graduate Certificate in Information Technology offer COMP9120 2019,['B-units.courseCode' 'B-courseRelations.course'],In_Related_Course
267406  267404,Can I choose Digital Media Fundamentals 2010 I am a Graduate Certificate in Information Technology Management student,['B-units.courseCode' 'B-yea
        'B-courseRelations.course'],In_Related_Course
267407  267405,Can I choose COMP5707 2017 I am a Software and Music Studies student,['B-units.courseName' 'B-year' 'B-courseRelations.course'],In_Related_Course
267408  267406,Do Software Mid-Year B-and INFO5060 include 2010 ,['B-units.courseName' 'B-courseRelations.course'],In_Related_Course
267409  267407,Can I choose CISS6022 2004 I am a Software and Science student,['B-units.courseName' 'B-year' 'B-courseRelations.course'],In_Related_Course
267410  267408,Do Software B-and COMP9110 include 2018 ,['B-courseRelations.course' 'B-units.courseCode' 'B-year'],In_Related_Course
267411  267409,Do Software and Arts B-and COMP5425 include 2011 ,['B-courseRelations.course' 'B-units.courseCode' 'B-year'],In_Related_Course
267412  267410,Does Software Mid-Year offer COMP5618 2008,['B-units.courseCode' 'B-courseRelations.course'],In_Related_Course
267413  267411,I am a Graduate Diploma in Information Technology can I select course Empirical Security Analysis and Engineering,['B-courseRelations.course'
        'B-units.courseCode' 'B-year'],In_Related_Course
267414  267412,Do Software and Medical Science B-and INFO5301 include 2009 ,['B-courseRelations.course' 'B-units.courseCode' 'B-year'],In_Related_Course
267415  267413,Can I choose Information Technology Capstone Project 2014 I am a Graduate Diploma in Information Technology student,['B-units.courseName' 'B-year'
        'B-courseRelations.course'],In_Related_Course
267416  267414,Can I choose Managing Business Intelligence 2017 I am a Graduate Diploma in Health Technology Innovation student,['B-units.courseCode'
        'B-courseRelations.course'],In_Related_Course
267417  267415,Can I choose Information Technology Short Project 2003 I am a SoftwareandScience student,['B-units.courseName' 'B-year'
        'B-courseRelations.course'],In_Related_Course
267418  267416,Can I choose Introductory Biostatistics 2007 I am a Software and Commerce student,['B-units.courseName' 'B-year'
        'B-courseRelations.course'],In_Related_Course
267419  267417,Can I choose Machine Learning and Data Mining 2010 I am a Graduate Certificate in Information Technology Management student,['B-units.courseName'
        'B-courseRelations.course'],In_Related_Course
267420  267418,Can I choose Parallel and Distributed Computing 2018 I am a Graduate Certificate in Information Technology student,['B-units.courseCode'
        'B-year'],In_Related_Course
267421  267419,Does Software offer Multimedia Retrieval 2000,['B-units.courseCode' 'B-year' 'B-courseRelations.course'],In_Related_Course
267422  267420,Does Master of ITandMaster of IT Management offer Computer and Network Security 2003,['B-units.courseName' 'B-year'
        'B-courseRelations.course'],In_Related_Course
267423  267421,Does Software and Commerce offer INFO5301 2012,['B-units.courseName' 'B-year' 'B-courseRelations.course'],In_Related_Course
267424  267422,Can I choose COMP5617 2005 I am a Software Mid-Year student,['B-courseRelations.course' 'B-units.courseCode' 'B-year'],In_Related_Course
267425  267423,Does Graduate Diploma in Computing offer Wireless Engineering 2008,['B-units.courseName' 'B-year' 'B-courseRelations.course'],In_Related_Course
267426  267424,Can I choose ELEC5510 2018 I am a Master of Health Technology Innovation student,['B-courseRelations.course' 'B-units.courseName'
        'B-year'],In_Related_Course
267427  267425,Does Master of Health Technology Innovation offer IT Capstone Project - Individual 2003,['B-units.courseName' 'B-year'
        'B-courseRelations.course'],In_Related_Course
267428  267426,Is Database Management Systems in Software Engineering and Science course structure,['B-courseRelations.course' 'B-units.courseCode'
        'B-year'],In_Related_Course
267429  267427,Does Software Engineering offer Professional Practice in IT 2009,['B-courseRelations.course' 'B-units.courseName' 'B-year'],In_Related_Course
267430  267428,Does Software Engineering and Medical Science offer COMP5046 2014,['B-courseRelations.course' 'B-units.courseCode' 'B-year'],In_Related_Course
267431  267429,Does Software and Science offer Signals 2009,['B-courseRelations.course' 'B-units.courseCode'],In_Related_Course
```

Figure 7 Raw Questions

In Figure 7, the first column is the index of the questions, and the second is the raw questions that we have create according to the template. The third and fourth columns are the slots used for generating the template and the intent which we labelled in the templates. Therefore, by now we have the raw questions with each question was carefully annotated with slots and intents. This outcome could be transported to our model for training.

1.2.3 Intent Design and Slot Design

Before we generate the raw questions, there is an important step that we need to carefully design and reach an agreement within our team. The problem at this point is how to design the intent and our slot. For intent design, it should include what is the source of intent, which indicates where can we find the initial intent; it also includes how to design the intent format, or more specific should we design the intents based on the CUSP structure or just according to what we come up with, or should we cluster them. For the slots designing, the fineness could be one problem that we needed to decide, because if we decide to use the CUSP structure to guide our designing, then we have to decide which level could be regarded as intents and which level could be treated as slots.

The intents we have designed have two levels. The first level we called it "main intent" and another level of intent we called it "sub-intent". The reason why we created intents in this way is that many questions could point to relative general questions whose answers represent a higher level structure. For example, if a student asks that "what is the lecturer of Machine Learning this semester", the we could labelled this question has the intent with "lecture". However, in another situation, we have to handle the questions such as "Tell me assessment information about Natural Language Process". This question introduces another case that we have to return the entire information that we retrieved from database by using fuzz search. The result could be a table that contains several available information or several attributes with concrete value in it. The figures below shows the CUSP structure and intents design for this project.

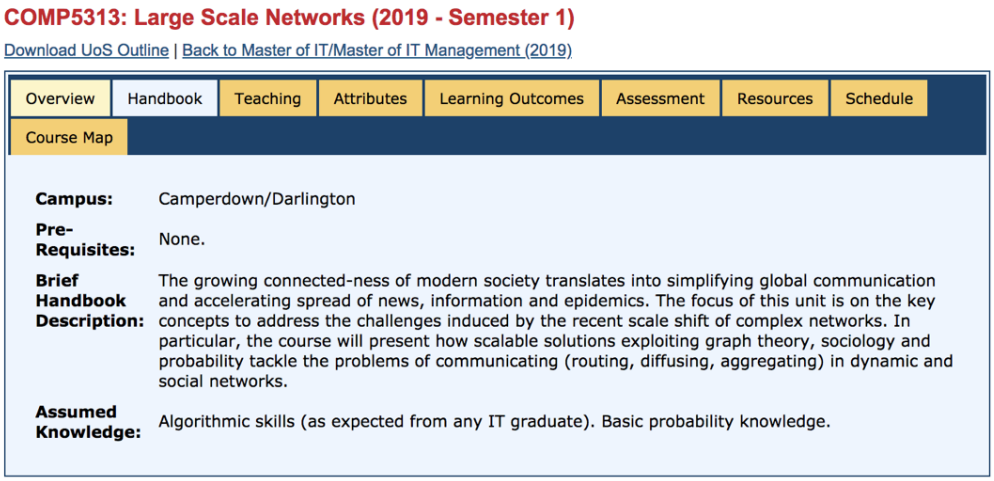


Figure 8 CUSP Structure

Handbook	prerequisites
Teaching	tutors
Assessment	assessmentMethods
Assessment	assessmentMethods
Assessment	grading
Overview	onOffer
Teaching	lecturer
Overview	level
Overview	level
Teaching	lecturer
Overview	faculty
Overview	sessionOptions
Overview	sessionOptions
Overview	faculty

Figure 9 Main intents and Sub-intents

The figure 8 shows the structure that the cusp organized and based on this structure we built our database table structure. That is why we have to design intents in two level structure, the main intents directly accord with the database structure in order to retrieve information from database. The figure 9 shows the outcome we actually designed with this issue.

1.2 Text Preprocessing

1.2.1 Tokenization

Tokenization, in natural language processing, is one of the simple processes to preprocessing the text, the purpose of tokenization is to simplify the text as the input for the following functions of the system. The most common steps of general text tokenization are: converting all letters to lower or upper case， converting numbers into words or removing numbers, removing punctuations, accent marks, and other diacritics, removing white spaces, expanding abbreviations, removing stop words, sparse terms, and particular words and text canonicalization.

In the Cassandra project, the first step is to preprocess the input text, it is important to choose the necessary steps and make sure the result is useful for the intent classification and slot filler part. For Cassandra project, The purpose of preprocessing the text is to omit the length of the sentence, leaving the more important part of the sentence to omit the less useful part of the sentence.

First, we need to convert all letters to lower case. In this step, we first unify the text format so that we can follow the string matching and database operations.

E.g

The user's input is "Hi, I want to know the lecturer address of COMP5426."

The output of this step is "hi, i want to know the lecturer address of comp5426."

converting all letters to lower or upper case

```
[ ]  1 input_str = "Hi, I want to know the lecturer address of COMP5426."
     2 input_str = input_str.lower()
     3 print(input_str)
```

```
hi, i want to know the lecturer address of comp5426.
```

Figure 10 convert all letters to lower case

In this sentence, we need to unify "COMP" into a lowercase format so that we can match the background database.

Second, converting numbers words into numbers, for the convenience of back slot matching, so in the preprocessing we need to convert numbers words into numbers.

E.g

"comp five four two six" appears in the user's question,

So in the ability of people to understand, we know that users want to express "comp5426", but it is more convenient to query numbers in the database than words, so you need to turn such digital words into numbers in the process of processing.

Third, removing stop words, when the user enters a sentence, although these words are necessary grammatically, they are not necessary for understanding the sentence. So in order to shorten the length of the sentence as much as possible, we will remove stop words in this step.

E.g

The user's input is "hi, i want to know the lecturer address of comp5426."

The output is ['hi', 'want', 'know', 'lecturer', 'address', 'comp5426']

In this sentence we can identify some components that appear frequently but have no meaning to the whole sentence, such as “to” and “the”.

```
[ ]  1 import nltk
     2 from nltk.corpus import stopwords
     3
     4 input_str=['hi', 'i', 'want', 'to', 'know', 'the', 'lecturer', 'address', 'of', 'comp5426']
     5
     6 stop_words = set(stopwords.words('english'))
     7 result = [i for i in input_str if not i in stop_words]
     8 print (result)
```

```
['hi', 'want', 'know', 'lecturer', 'address', 'comp5426']
```

Figure 11 removing stop words

To ensure the stop words set can conclude all stop word, It is necessary to select a open source stop words package. For this project, we downloaded the NLTK stop word package.

```
[ ]  1 import nltk
     2 nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

Figure 12 download stop words

1.2.2 Part of speech tagging

POS refers to Part of speech tagging. The part-of-speech tagging is also a basic module in natural language processing, laying the foundation for syntactic analysis and information extraction. It is generally necessary to first segment the statement and then perform part-of-speech tagging.

The part-of-speech tagging algorithm is also divided into two categories, a dictionary search algorithm based on string matching and a statistical-based algorithm. The jieba participle combines two algorithms. For the words identified after the word segmentation, the word part is directly searched from the dictionary. For unregistered words, the HMM hidden Markov model and the viterbi algorithm are used to identify.

In Cassandra project, POS does not have the necessary role in the current structure. We mainly use POS for text analysis. In the later answer generation and the next stage of the raw question generation we may perform chunk parsing and dependency parsing on the text.

We introducing the NLTK(Natural Language Toolkit) to finish this step. It is necessary to download two helper package for POS.

```
[ ]  1 import nltk
     2 nltk.download('punkt')
     3 nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

Figure 13 download POS package

And the main part of POS:

```
from textblob import TextBlob

input_str = "Hi, I want to know the lecturer address of COMP5426."
result = TextBlob(input_str)
print(result.tags)
```

```
[('Hi', 'NNP'), ('I', 'PRP'), ('want', 'VBP'), ('to', 'TO'), ('know', 'VB'), ('the', 'DT'), ('lecturer', 'NN'), (
```

Figure 14 POS example

1.2.3 Word Embedding

Word embedding is a method of expressing words better by using a vector method. This method turns words into vectors that can justify the relationship between words. The main word embedding methods include word2vec and FastText.

Word embedding is a necessary step before trying to use intent classification. The easiest way to use word embedding is to give each word a label, or use one-hot encoding to turn a word into a vector, but none of these methods can reflect the relationship between words, so we intend to use the word2vec or fastText method.

Fasttext skipgram

```
from gensim.models import FastText
import time
import os
```

```
save_dir = "./modelfor_fasttext/"
if not os.path.exists(save_dir):
    os.makedirs(save_dir)
```

```
start = time.time()
fasttext_cbow = FastText(sentences, size=300, window=7, min_count=3, workers=workers,sg=0, seed=seed)
print("Training FastText CBOW took {} seconds".format(time.time()-start))
```

```
Training FastText CBOW took 11.462303161621094 seconds
```

Figure 15 FastText skipgram word embedding

Word2Vec skip gram

```
save_dir = "./modelfor_skipgram/"
if not os.path.exists(save_dir):
    os.makedirs(save_dir)
```

```
start = time.time()
w2v_sg = Word2Vec(sentences=sentences, size=300, window=7, min_count=3, workers=workers, sg=1, seed=seed)
print("Training Word2Vec Skip-Gram took {} seconds".format(time.time()-start))
```

```
Training Word2Vec Skip-Gram took 5.295786380767822 seconds
```

```
w2v_sg.save(os.path.join(save_dir, "w2v_sg.bin"))
```

Figure 16 Word2Vec skip-gram word embedding

After word2vec on the problem set, we found that the results were not available in our model because the number of words in our training set was too small. Therefore, we decided to use an open source word vector library. For the specific course name, the teacher name data is solved by the stream matching method.

Figure 17 download wiki-news-300 word vector

```python
import numpy as np
from gensim.models import KeyedVectors

class Word_embedding_Model:

    def __init__(self):
        # Creating the model
        model_loc = "/Users/sunshengyuan/PycharmProjects/capstone/week8/wiki-news-300d-1M-subword.vec"
        self.en_model = KeyedVectors.load_word2vec_format(model_loc)
        self.PAD = np.zeros(300).tolist()
        self.UNK = np.ones(300).tolist()
        self.SOS = np.zeros(300).tolist()
        for i in range(300):
            self.SOS[i] = 2

        self.EOS = np.zeros(300).tolist()
        for i in range(300):
            self.EOS[i] = 3

    def get_word_vector(self,word):
        try:
            result = self.en_model[word]
            # Print out the vector of a word
            #print("Vector components of a word: {}".format(self.en_model[word]))

            return result.tolist()
        except:
            #print("Vector components of a word: {}".format(self.UNK))
            return self.UNK
```

Figure 18 example of load word vector

1.3 Intent classification

The first problem the team have to solve in the project is the natural language understanding problem, which is NLU. To design an intent classification to handle user intent. There are many implementations of intent classification, here the implement only discuss methods based on the seq2seq model, such as Bi-model with a decoder (Wang et al., 2018), Intent Gating & self-attention (Li et al., 2018). Atten.-Based (Liu and Lane, 2016), BLSTM (Zhang et al., 2016), Capsule Neural Networks (Zhang et al., 2018).

Understanding user intent is the first step in the next steps. From the user's intent, we then choose which logic to use to get the desired result based on the logical logic of the backend. Based on the intent category and the raw question training set designed above. We decided to implement this model using the seq2seq method, the model is divided into two parts: encoder and decoder.

In encoder part, after review few method ,the Bi-LSTM structure is the most popular one structure used for encoder. At the beginning, it is need define two LSTM cell for forward and backward. We keep the output prob equal to 0.5, This is an effective regularization method that can effectively prevent overfitting.

```
encoder_f_cell_0 = LSTMCell(self.hidden_size)
encoder_b_cell_0 = LSTMCell(self.hidden_size)
encoder_f_cell = DropoutWrapper(encoder_f_cell_0,output_keep_prob=0.5)
encoder_b_cell = DropoutWrapper(encoder_b_cell_0,output_keep_prob=0.5)
```

Figure 19 encoder LSTM cell design

Then we use the bidirectional_dynamic_rnn to train the input data to the finial hidden layer state and the output of the structure.

```
(encoder_fw_outputs, encoder_bw_outputs), (encoder_fw_final_state, encoder_bw_final_state) = \
    tf.nn.bidirectional_dynamic_rnn(cell_fw=encoder_f_cell,
                                    cell_bw=encoder_b_cell,
                                    inputs=self.encoder_inputs_embedded,
                                    sequence_length=self.encoder_inputs_actual_length,
                                    dtype=tf.float32, time_major=True)
encoder_outputs = tf.concat((encoder_fw_outputs, encoder_bw_outputs), 2)

encoder_final_state_c = tf.concat(
    (encoder_fw_final_state.c, encoder_bw_final_state.c), 1)

encoder_final_state_h = tf.concat(
    (encoder_fw_final_state.h, encoder_bw_final_state.h), 1)

self.encoder_final_state = LSTMStateTuple(
    c=encoder_final_state_c,
    h=encoder_final_state_h
)
```

Figure 20 encoder design

The decoder part for intent classification is more simpler compared with the slot filler. The output only one value, then it can be return by a argmax layer. The input of the this layer is the final hidden layer state.

```
intent_W = tf.Variable(tf.random_uniform([self.hidden_size * 2, self.intent_size], -0.1, 0.1),
                       dtype=tf.float32, name="intent_W")
intent_b = tf.Variable(tf.zeros([self.intent_size]), dtype=tf.float32, name="intent_b")

# intent
intent_logits = tf.add(tf.matmul(encoder_final_state_h, intent_W), intent_b)
# intent_prob = tf.nn.softmax(intent_logits)
self.intent = tf.argmax(intent_logits, axis=1)
```

Figure 21 decoder for intent classification

1.4 slot filler

First, Cassandra is a closed-domain dialogue system. The task-oriented systems has two distinct characteristics. The first is that each problem has a clear slot requirement, and the second feature is that the dialogue has a clear purpose. For the slot filling part we also use the seq2seq model and share the same encoder with the intent classification. Here we only introduce the difference between the decoder parts.

Same as the intent classification decoder, the input is the state of the last hidden layer of encoder, the structure of the decoder divide to the training mode and prediction mode, at first we will define the decoder helper, The helper is actually how the decoding stage can get the input at the next moment according to the prediction result. For example, the actual value of the previous moment should be directly used as the next moment input during the training process. The greedy method can be used to select the value with the highest probability as the prediction process. The next moment and so on. So helper can be roughly divided into training helper and predictive helper.

In the training phase, using the combination of Training Helper + Basic Decoder, this is generally fixed, of course, you can also define the Helper class yourself. The prediction phase calls Greedy Embedding Helper + Basic Decoder combination for greedy decoding.

```
training_helper = tf.contrib.seq2seq.TrainingHelper(inputs=decoder_inputs_embedded,
                                                    sequence_length=self.decoder_targets_length,
                                                    time_major=False, name='training_helper')
predict_helper=decoding_helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(embedding=embedding,
                                                                          start_tokens=start_tokens,
                                                                          end_token=end_token)
```

Figure 22 decoder helper

Currently, Attention mode plus LSTM is one of the effect structure for decoder, to implement this structure first we need define the Attention machine

```
memory = tf.transpose(encoder_outputs, [1, 0, 2])
attention_mechanism = tf.contrib.seq2seq.BahdanauAttention(
    num_units=self.hidden_size, memory=memory,
    memory_sequence_length=self.encoder_inputs_actual_length)
```

Figure 23 Attention machine

And Define the decoder stage to use the LSTMCell and then encapsulate the attention wrapper

```
cell = tf.contrib.rnn.LSTMCell(num_units=self.hidden_size * 2)
attn_cell = tf.contrib.seq2seq.AttentionWrapper(
    cell, attention_mechanism, attention_layer_size=self.hidden_size)
out_cell = tf.contrib.rnn.OutputProjectionWrapper(
    attn_cell, self.slot_size, reuse=reuse
)
```

Figure 24 Attention LSTM cell

Final apply the dynamic decoder by Tensorflow to produce the output. Call dynamic_decode for decoding, decoder_outputs is a namedtuple which contains two items (rnn_outputs, sample_id)

Rnn_output: [batch_size, decoder_targets_length, vocab_size]

Sample_id: [batch_size, decoder_targets_length], tf.int32

```
        decoder = tf.contrib.seq2seq.BasicDecoder(
            cell=out_cell, helper=helper,
            initial_state=out_cell.zero_state(
                dtype=tf.float32, batch_size=self.batch_size))
        # initial_state=encoder_final_state)
        final_outputs, final_state, final_sequence_lengths = tf.contrib.seq2seq.dynamic_decode(
            decoder=decoder, output_time_major=True,
            impute_finished=True, maximum_iterations=self.input_steps
        )
        return final_outputs
```

Figure 25 dynamic decoder plus attention

The final output of training:

```
[Epoch 46] Average train loss: 0.0
Input Sentence        :  ['in', 'which', 'film', 'does', 'a', 'recently', 'widowed', 'man', 's', 'son', 'calls', 'a', 'radio', 'talk', 'show', 'in', 'an',
Slot Truth            :  ['O', 'O', 'O', 'O', 'B-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot'
Slot Prediction       :  ['O', 'O', 'O', 'O', 'O', 'B-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-
Intent Truth          :  Plot
Intent Prediction     :  Plot
Intent accuracy for epoch 46: 0.7996926229508197
Slot accuracy for epoch 46: 0.8207318419124349
Slot F1 score for epoch 46: 0.8210318376561737
[Epoch 47] Average train loss: 0.0
Input Sentence        :  ['what', 'is', 'the', 'name', 'of', 'the', 'alfred', 'hitchcock', 'film', 'where', 'james', 'stewart', '<UNK>', 'a', 'murder', 'w
Slot Truth            :  ['O', 'O', 'O', 'O', 'O', 'O', 'B-Director', 'I-Director', 'O', 'O', 'B-Actor', 'I-Actor', 'B-Plot', 'I-Plot', 'I-Plot', 'I-Plot'
Slot Prediction       :  ['O', 'O', 'O', 'O', 'O', 'O', 'B-Director', 'I-Director', 'O', 'O', 'B-Actor', 'I-Actor', 'I-Actor', 'B-Plot', 'I-Plot', 'I-Plot
Intent Truth          :  Plot
Intent Prediction     :  Plot
Intent accuracy for epoch 47: 0.7838114754098361
Slot accuracy for epoch 47: 0.8197677123948524
Slot F1 score for epoch 47: 0.819723440827112
[Epoch 48] Average train loss: 0.0
Input Sentence        :  ['im', 'thinking', 'of', 'a', 'documentary', 'of', '<UNK>', '<UNK>', 'living', 'in', 'the', 'arctic', 'circle', 'which', 'was', '
Slot Truth            :  ['O', 'O', 'O', 'O', 'B-Genre', 'O', 'B-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'O', 'O', 'O', 'O', 'B-
Slot Prediction       :  ['O', 'O', 'O', 'O', 'B-Genre', 'O', 'B-Relationship', 'O', 'I-Plot', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-Director', 'I-Di
Intent Truth          :  Plot
Intent Prediction     :  Actor
Intent accuracy for epoch 48: 0.7838114754098361
Slot accuracy for epoch 48: 0.8180846150054609
Slot F1 score for epoch 48: 0.8183101121289099
[Epoch 49] Average train loss: 0.0
Input Sentence        :  ['in', 'what', '2006', 'disney', 'movie', 'does', 'robert', '<UNK>', 'jr', 'kidnap', 'an', 'animal', 'in', 'order', 'to', 'find',
Slot Truth            :  ['O', 'O', 'B-Year', 'B-Genre', 'O', 'O', 'B-Actor', 'I-Actor', 'I-Actor', 'B-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-P
Slot Prediction       :  ['O', 'O', 'B-Year', 'O', 'O', 'O', 'B-Actor', 'I-Actor', 'I-Actor', 'B-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot', 'I-Plot',
Intent Truth          :  Plot
Intent Prediction     :  Plot
```

Figure 26 final output of intent classification and slot filler

1.5 Dialog State Tracker

The dialog state tracker is a stack structure part in the entire system which enable the system to stock and do stacking job or have the dispatch functionalities. The dialog state tracker contains two parts, one is called “log system” which simply store the structuralized dialog stream. It will store both directions of a session of communication. The directions include the dialog stream from user to Cassandra system and the sentences that our system reply, which indicates the stream from our Natural Language Generator(NLG) to clients. Both this two directions information will be recorded in a structural way. In this case, it will be recorded in a XML files temporarily. Our team decided that the XML files could accelerate the operations in the system.

The log system has some input and output format, which could be used for further tracking the communication flow. That is significant for our tracking the history dialog that we could used for debugging or evaluating our system. The figure below shows the input format that our system received at the beginning of this part and its output which are several concreate XML files with the format defined in the figure.

```
log system{

  input{
    sentence:{"What is the courseCode of Machine Learning"|"The courseCode of Machine
learning is COMP5048"}
    slots:{"O O O O B-CourseCode O B-CourseName"}
    intent:{"Overview-tutors"}
  }

  output_XML{
    session:{"sessionID"}
    sessionURL:{URL_of_Connection}
    users:{"user"|"chatbot"}
    time:{sys.time}
    sentence:{"What is the courseCode of Machine Learning"|"The courseCode of Machine
learning is COMP5048"}
    slots:{"O O O O B-CourseCode O B-CourseName"}
    intent:{"Overview-tutors"}
  }

}
```

Figure 29 input and output format of log system

The log system would be retained in the servers memory. Meanwhile, any internal or external connection would be recorded in this system for further tracking. Another tracker in this system is dialog state tracker. The dialog state tracker could be used to maintain the context of a session during the communication. The dialog state tracker keeps the previous dialogues with their intents and slots. This design could help our system handle the situation like some sentences could be classified into none intent case. For example, when a user ask a question like “I want to know the lecturer of Machine learning”, this sentence will be fit with a intent that to indicate the system to query the lecturer of of this course. However, the user could ask with the following questions such as “What about Natural Language Process”. In this following sentence, users do implicated express the intent that they wish to find the lecturer of course Nature Language Process. However, from the respective of machines, they would not detect the intent that behind the sentence if there has not a clearly state tracker. Therefore, the dialogue state tracker is used for this purpose. It will record the previous state and store them dynamically. Then when this situation happen, the system will track back to the previous state to use the intent in of that sentence, and treat this intent as the the intent of this unknown-intent sentence. The original code is list in the below figure. It clearly shows that the system will track the dialogue state and get result from it. As for the missing slots, it also based on the similar rules.

```
round=1

user:
sentence:  ['I', 'want', 'to', 'know', 'the', 'assignment', 'of', 'INFO9117']
intent:  assignment
slots:  ['O', 'O', 'O', 'O', 'O', 'O', 'O', 'unit_code']


chatbot:
sentence:  [['about', '<unit_code>', 'there', 'are'], ['<grading>', '<method>', '<description>', '<policy>'], ['wha
t', 'you', 'want', 'to', 'know', 'first']]
intent:  assignment
slots:  ['unit_code']
keywords: ['assignment']


round=2

user:
sentence:  ['grading']
intent:  None
slots:  ['O']


chatbot:
sentence:  [['<unit_code>', 'have', '<num_of_grading_type>', 'type'], ['<grading_type_name>']]
intent:  assignment_grading
slots:  ['unit_code']
keywords: ['grading']
```

Figure 30 The round state tracker

The figure 11 that we have done is a simulation to test this component of system. In this case, the communication between user and our system could be tracked one by one. That is a way to detected original intents or slots. If the missing slots or missing intents happen, the components will automatically fill the blank with lasted used intent or slots that stacked in the tracker.

1.6 Database

1.6.1 Introduction

Database is essential for both storing data and retrieving. CUSP is comprised by structured course which are different with each other. It is crucial to design a flexible database to hold all data with considerable expansibility and cooperate with retrieving process. Well embedded data structure design would reduce the efforts on logical pre- and post-processing on slot-intent classification model's output and benefit to query template.

1.6.2 Benefits

Less efforts in logical process could be achieved if database design is adaptable. By applying the well-defined database, query template can be mapped efficiently under different condition when question comes.

1.6.3 Database design

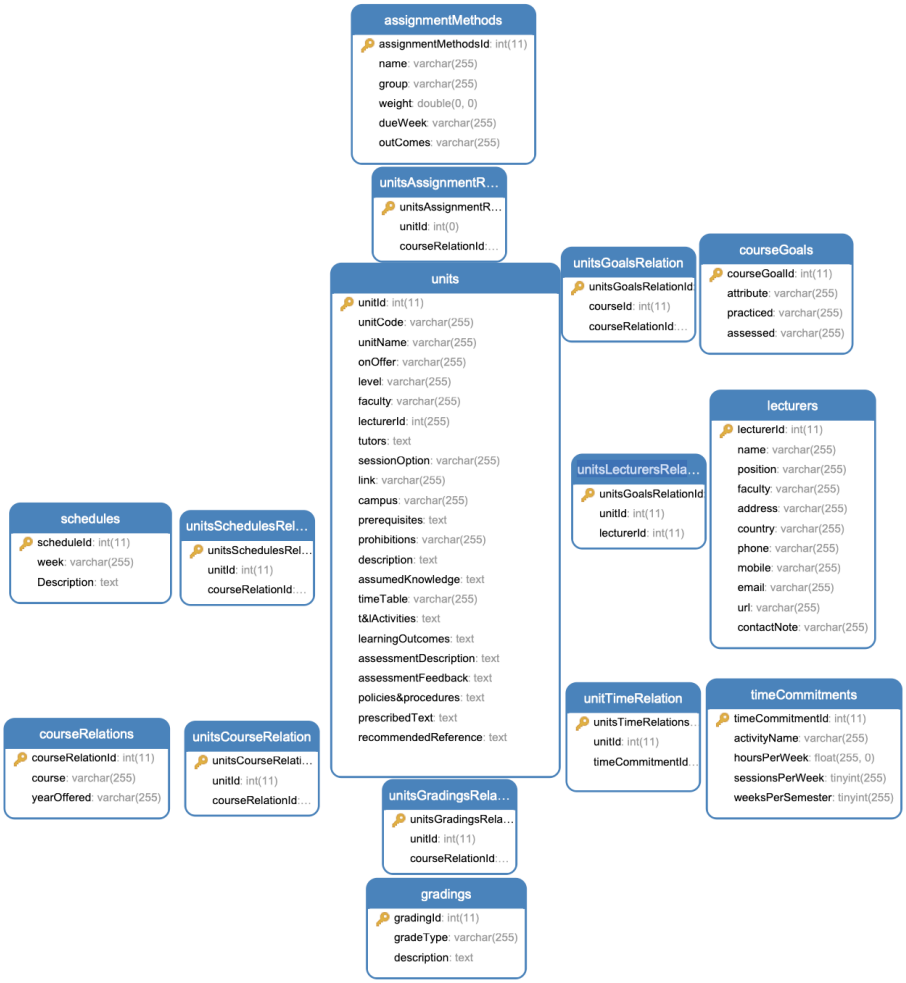


Figure 31 Final database design

1.6.4 Benefits

Because database is designed, established by MySQL and connected with Peewee, the interface of database is supplied by Peewee. There are several query codes which will be called after logical process when question comes. The output this process is the specific data in database.

1.6.5 Implementation

Specific keywords and slot need to be applied to map the query template code when retrieving database. In order to implement query successfully, Logical process which will be illustrated in next section is introduced before using query code.

```python
def re_units(slots,course):
    '''Retrieve information under Units table
    INPUT: slots, course(course name or code with ' ')
    RETURN: information
    '''
    query = (Units
             .select(eval('Units.'+slots[0]))
             .where(eval('Units.'+slots[2]+'=='+course))
             .dicts())
    for i in query:
        return i
```

Figure 32 One of example on executing query

There are some examples for retrieving template code and result.

```python
# Search Units' one info

# query INFO1110's session option
query = (Units
         .select(Units.sessionoption)
         .where(Units.unitcode=='INFO1110')
         .dicts())
for i in query:
    print(i)

# query 'Introduction to Programming''s on offer option
query = (Units
         .select(Units.onoffer)
         .where(Units.unitname=='Introduction to Programming')
         .dicts())
for i in query:
    print(i)
```

```
{'sessionoption': 'Semester 1, Semester 2'}
{'onoffer': 'Yes'}
```

Figure 33 Retrieve one unit's one attributes

```python
# Search Units' multiple info

# query INFO1110's session option
query = (Units
         .select(Units.sessionoption, Units.onoffer)
         .where(Units.unitcode=='INFO1110')
         .dicts())
for i in query:
    print(i)

print()

# query 'Introduction to Programming''s on offer option
query = (Units
         .select(Units.faculty, Units.description)
         .where(Units.unitname=='Introduction to Programming')
         .dicts())
for i in query:
    print(i)
```

```
{'sessionoption': 'Semester 1, Semester 2', 'onoffer': 'Yes'}

{'faculty': 'School of Computer Science', 'description': 'This unit is an essential starting point for software devel
opers, IT consultants, and computer scientists to build their understanding of principle computer operation. Students
will obtain knowledge and skills with procedural programming. Crucial concepts include defining data types, control f
low, iteration, functions, recursion, the model of addressable memory. Students will be able to reinterpret a general
problem into a computer problem, and use their understanding of the computer model to develop source code. This unit
trains students with software development process, including skills of testing and debugging. It is a prerequisite fo
r more advanced programming languages, systems programming, computer security and high performance computing.'}
```

Figure 34 Retrieve multiple attributes in one unit

```python
# test query Units info with non return
# F, coursecode not exist
query = (Units
         .select(Units.sessionoption) # need to be fill
         #.where(Unitsassignmentrelation.courseid==1)
         .where(Units.coursecode=='INFO1110')
         .dicts())

for i in query:
    print(i)
print(query.exists())
```

```
{'sessionoption': 'Semester 1, Semester 2'}
True
```

Figure 34 Retrieve quiz information in one unit

1.7 Logical system

1.7.1 Introduction

Logical process are developed to handle different kinds of problems properly under slot-intent classification model output and client's question. It checks the integrity of classified slots and intent to collect enough information for retrieving. All those problems such as slot missing, client's question is too general etc need a complete logical process to handle which makes system be more robust and stable.

1.7.2 Benefits

This part handles the detail process and solution which will keep system stability and let it be robust. For example, unit name missing occurs when client's question not cover full unit name or slot-intent classification model only label part of course name.

1.7.3 Implementation

When slot missing happen or client's question is not too general, system will collect key information by asking and guiding client to select attributes according to course structure and logical process. For example, there are some genera problem.

1. Course name slot missing.

   It may happen when model can not label all parts of course name as course name slot or client's question not cover complete course name, which lets matched course name can not be applied as keyword in Peewee template. To solve this problem, Levenshtein distance which is a method to calculate string

similarity is applied to return most similar course name in database for client. System will gather client's selection to fullfill the course name attributes in Peewee template code so the query process can be execute successfully. There are some codes illustrating the process of course name checking and choosing.

```
# Compute similarity
# Programming Languages is not complete
process.extract('Programming Languages', coursename, limit=5)
[name for name in process.extract('Programming Languages', coursename, limit=5)]
```

```
[('Programming Languages, Logic and Models', 90),
 ('Programming Languages, Logic and Models (Adv)', 90),
 ('Programming Languages and Paradigms', 90),
 ('Systems Programming', 69),
 ('Introduction to Programming', 66)]
```

Figure 36 String similarity calculation example

```
def choose_information(tochoose):
    '''Let clinet to choose
    INPUT: course_name_or_code, existed content list
    RETURN: selection course name or code
    '''
    output = 'Do you means those course?\n'
    for i, item in enumerate(tochoose):
        output = output + str(i)+ ': ' + item + '\n'
    output = output + 'Please choose the number.'
    choosed = int(input(output))
    return tochoose[choosed]

def check_existence(course_name_or_code,contentlist):
    '''Judge the where course is existed, if match return True
    else return selection
    INPUT: course_name_or_code, existed content list
    RETURN: Ture or selection course
    '''
    if course_name_or_code in contentlist:
        return True
    else:
        need_to_select = [name[0] for name in process.extract(
            course_name_or_code, coursename, limit=5)]
        return need_to_select
```

Figure 37 Functions to generate course list need choosing

2. Question is too general.

Question such as "Show me the assignment of INFO1110" is too general and keywords is extremely limited which can not match any Peewee template code. In this condition, system needs to provide subsection according to CUSP structure to guide client to choose the information they need. In above question, systems will return selection < Assessment Methods,

Assessment Description, Grading, Policies & Procedures> to customer and waiting for response because whose attributes listed in Assessment category on CUSP. It is crucial to checking whether the data under attribute is table is crucial due to system need customer make a choose on querying specific part of table or the whole table.

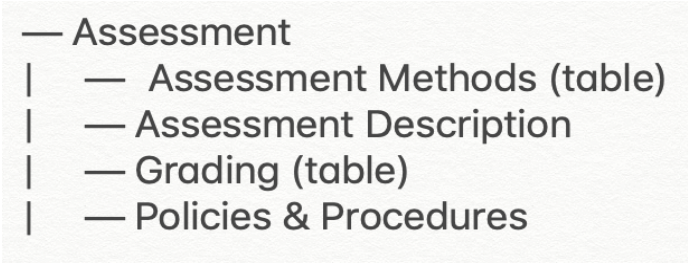


Figure 38 Assessment structure

3. Invalid question.

Invalid question represent those question which are greeting or invalid sentence which can not be used to query. Those sentence's slot and intent may be null and system need to response for those invalid question in order to guide customer to ask valid question. For example, sentences like "Hello", "What a good day", etc. are invalid and system will return "Hello, you can ask me question like this: Show me the timetable of INFO1110".

1.7.4 Outcome

This part contributes the stability and robustness in system and management according to logical processing.

2. OBSTACLES

2.1 Project Level Obstacles

2.1.1 The obstacles in the project structure

In the original project structure, we designed in the proposal report as follows, this Conversational Assistant will use Google Home API to transfer voice to the text. Then Dialogue Analyzer will do intent classification and slot matching for the text input. All processed text will input to the rule-based system to match a rule and create a SQL for searching data from database. The selected data will be delivered into Natural Language Generator to be added into answer template which will generate a complete sentence as an answer. The final answer will transfer to

voice as on output to students via Google Home API in the voice-text converter again.

This structure and process is correct and reasonable for the whole project. However, when we followed this structure, we found several challenging problems we can not solve them as our initial design.

a) The first obstacle is on training dataset found in Week 6. We found several datasets including suitable intent and slot classification. However, they can not be a direct reference to match our "course selection" topic. This obstacle is the biggest one we met. This training dataset will influence the output performance of dialogue analyzer, since the results of intent classification and slot matching are the foundations of the whole system. There is uncertainty in the model itself significantly. If we do not have enough training dataset for training. We have to meet the intent misunderstanding or slot lost problems in the test part. Moreover, it will limit the design of next parts.
To solve this problem, we tried two different ways to get more training data. The first one is using dependency passing to analyze sentence structure. This can help us to analyze training questions with a small scale. The second one is writing some sample questions as a question template. This template will include varieties of questions to cover all aspects of this system. These two methods we are both still working on, so the final results of which one better for the dialogue analyzer can not be judged at the moment.

b) The second obstacle is on rule-based system. We tried to provide a logic rule for the rule-based system, however, this logical structure is not a real rule-based system design. We got a wrong understanding of rule-based system. This part influence the input of the answer template design and the interface of detailed SQL function design.

However, we found this obstacle in Week 8. The time left us to modify and complete the whole system is limited. After our several discussions, we might exchange a logical system to the rule-based system in order to process the project successfully.
The new structure we modified as follows to lead us to complete our final work.

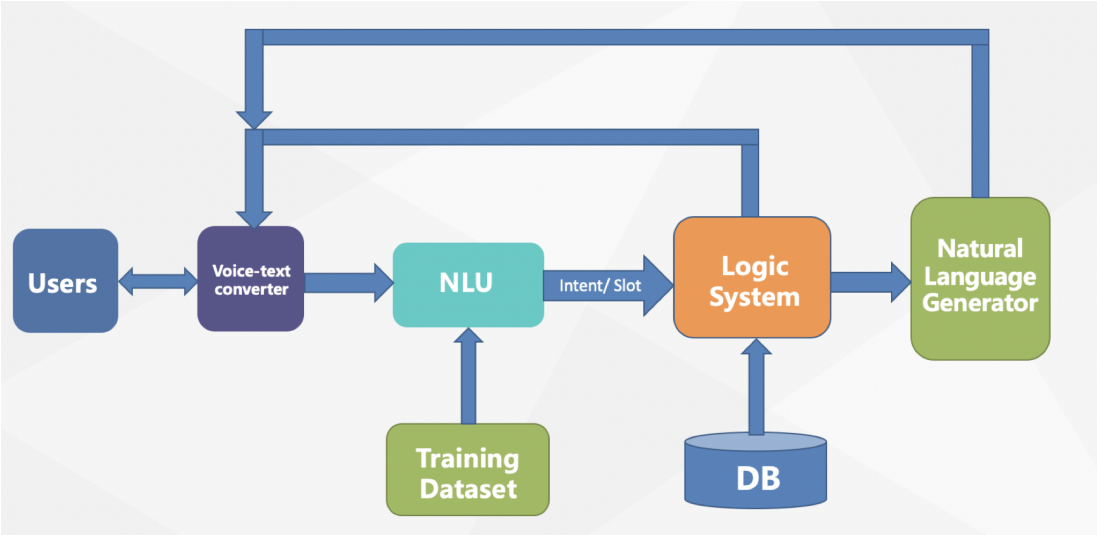


Figure 39

2.1.2 The obstacles in the project management

Although we had a positive group work attitude during the previous weeks, we still found some team cooperation problems we have to spend amount of time to solve.

In the last 9 weeks, we usually used slack and zoom to meet and discuss our group project. However, we seldom used Trello to manage our deliverables whether complete on time or not. Furthermore, because we lack of group experience of large programming project, we do not design input and output formats at initial period. When we wanted to combine each individual works together, we found we need to spend extra time to unify the format of inputs and outputs.

2.2 Implement Level Obstacles

2.2.1 Raw Question Obstacles

The most urgent and significant obstacles our team faced in this project is that we do not have any pre-collected training dataset for training. Therefore we have to firstly to collect the raw question data and then do the annotation part.

The original state of this part is that our team always cannot reach an agreement about how to create these raw questions. Initially, we were going to do several interviews or surveys which tries to gather some initial questions or real questions and then we could annotate these questions by ourselves. However, this initial plan was discarded due to the reason that we needs thousands of raw questions rather than hundreds of questions. The amount of raw questions demands are so large that we could not collect them directly from users with this limited time. Therefore, we changed the plan to a comparatively easier method that using template that we designed to automatically generate the raw question. Meanwhile, this method could help us pre-labelling this raw questions without post-processing.

The Figure 6 above displays this method that we use. However, before we got the actual question templates. Some other problems also blocked our project process. For example, how to confirm the format of templates. Because the format and the principles or rules of creating the templates are significant, and four of us cannot agreed each other, then the project was deadly delayed due to this reason. This obstacle was finally handled by serval group meeting. The final solution has been uploaded to the game. That is the final templates that we created in the google drive.

Another issue that related to the raw question generation is that the data in the database still cannot available directly. Due to this reason, some blanks or slots in the templates cannot be filled with the actual data. Because of missing of data in database, almost of us cannot how to define the slots and blanks in a template. That is why the template designing and its following step, which is raw question generation was severely postponed.

This dead delay resulted in entire delays of our projects. This dependency part is model training. Although our model has been training by other dataset such as "ATIS" which is a airline booking dataset, the actual model stille needs retrainied by our dataset. Therefore, it would postpone the timeline of this step.

2.2.2 Dialog State tracker Obstacles

The dialogue state tracker is actual a stack for tracking the states of a dialogue. In short, it is kind of log system. In the original version of our project, the tracker only trace back the latest intents of the users' asking for detecting the missing intents and missing slots, which has been discussed in the achievement part of this report. However, some other problems jumped out when we discussed the details of the state tracker. That is we need an algorithms or strategies to tracked back several level rather than the nearest level. That indicates we have to redesign the stack struct and the rules in the trackers. This redesign delayed the projects, though it has been completed by now.

Another problem that could also related to the state tracker is that when we have got some missing slot or intents, how could we replace them into already exists stackers. Do we needs to eliminates current sentences in the stack or just replace the missing part and replace them. Our team has provided the final solution though it has been implemented. That is just replacing the missing part and only with a slight migration, which get back to our original initiatives.

This part could only serve the situation that when the intents are missing. Therefore, there is not strong dependency with other part. So it would not impact other part implementation.

2.2.3 Hardware driver Obstacles

Cassandra Project will be deployed on Google Home devices, which is a home based smart device for assisting home user. This device is a home speaker that could receive audio waves and upload Google server for further operating. Cassandra is this kind of project that need to be deployed on Google Home to actually uses, though it could be only represented in a terminal version. The obstacles in this case, its that Google Home device only provide some limited product API to developers. For example, Google Assistant, which is a software ecosystem that Google Home can run. All applications that need to be run on Google Home have to comply with its software frame. In another words, our model, which is a communication program, has to be deployed on DialogFlow, which is a product that Google developed. DialogFlow has their own model training method and we could only use them rather than manually replace them. Therefore, this frame will impose some unnecessary limitation on our project and could result in the situation that we could not finally deploy our model, rule base and tracker on it.

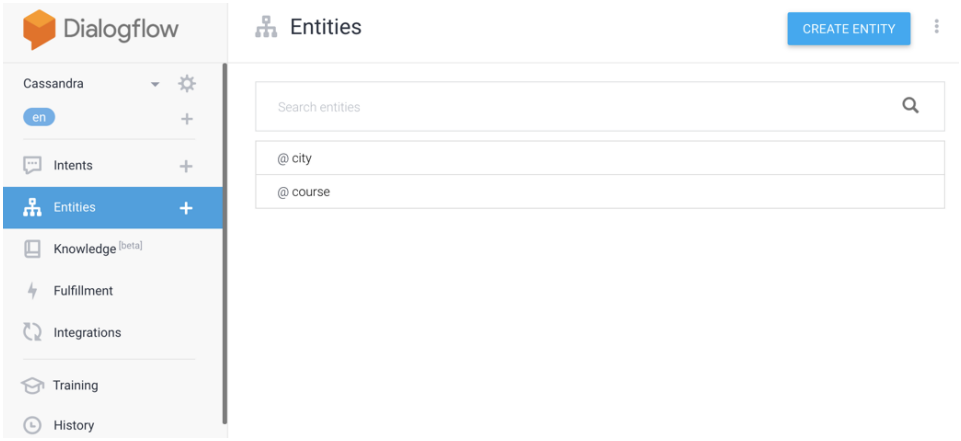


Figure 40 DialogFlow

This figure shows that the DialogFlow only define the intents and entities in this predefined structure without mutation its structure. Even the action could not be invoked our model. Therefore, this product level API could not help us too much to deploy the model.

This part does not have strong dependency relations with other part. It is optional rather than compulsory.

Other solution to this part is that we could use Google API such as, Speech to Text and Text to Speech to convert the voice that we received from the devices, and transport them to the Google Cloud, which is not convenient to use for developers but have to due to the the connection and authentication issues of Google devices. However, these Google API can only handle the audio files with some specific file

format, which involves sound sampling techniques, with which we do not familiar. Meanwhile, and the biggest obstacles of this part is that there is not, at least we do not find, a API that could help us to receive audio file from Google Home files, or we could send audio file to Google Home to let them to speak out. The API that Google Provides just for calls center or other only stream website which already have the stream audio sampling and audio playing services. Therefore, for deploying our project on concrete devices, finding available API is the big obstacle.

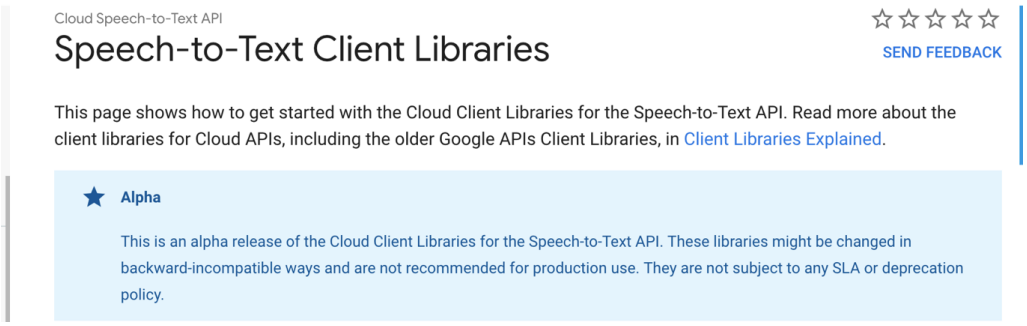


Figure 41 Google Speech to Text API (Alpha version)

This part also do not have strong dependency with other part. Therefore, it will not influence others' work.

2.3 Database obstacle

2.3.1 Previous state

Previous data table is not well defined with typos and lack of relation tables. Some of the attributes' data type is also need to be changed for efficient store data which saves memory.

2.3.2 Changes

Database needs to be change to correct store course information efficiently with less memory by applying appropriate data type. In Week 7 those problems are found when uploading data in database. There several parts of database are changed.

1. Some of the attributes in tables are changed to text data type

   Data type does not matching occurs when we upload data. To solve this problem some of the attributes such as description, assessmentFeedback, etc are change from varchar to text.

2. Lack of unitsLecturersRelation table

   Some of the course in CUSP has multiple lecturers while database is only designed in one unit-one lecturer pattern, which is wrong and not suitable. unitsLecturersRelation table are added under this condition.

3. Memory-efficiency

Some of the database table's attributes do not store long string but are assigned with text data type. Changing those attributes from text to varchar data type will save memory.

2.3.3 Solutions

There are one risk in database which is the design of database may not work very well with query template code. More efforts need to be payed on design query template code in order to solve this risk. To manage the risk, we need to identify the type of this risk and discuss with teammates with risk plan.

2.3.4 Risk dependencies

This risk may increase the complication of Peewee template code design and logical checking which both influence logical process and template matching.

2.3.5 Mitigation

In order to mitigate this risk influence, well defined logical process need to be design to match query template code. On the other hand, Peewee template code should be designed to be more flexible to match different kinds of condition.

2.4 Logical processing Obstacles

2.4.1 Changes

Logical process is new added part in system for checking, fulfilling the keyword and attributes to ensure successful query in database by guiding customer to specific question. This part is discussed in Week 8 and is in development.

2.4.2 Solutions

The risk in this part is that the logical process is too complicated and some of situation are not taken into consideration. To manage the risk, cooperation and information exchange between logical processing and database retrieving should be unimpeded.

2.4.3 Risk dependencies

This risk may influence whether system is guiding clients to information what they exactly want to know. It may also determine the Peewee template code flexibility because template code followed by logical processing.

2.4.4 Mitigation

In order to mitigate those risk, there are some methods.

1. Brainstorming the condition that will happen and consider solutions

   Asking some question taking system's process as reference to test logical process to check what types of conditions will happen. Recording those conditions and make a high level logical design.

2. More flexible logical processing

   Consider different conditions when design logical processing. Summarise those conditions and separate them to several parts for development.

3. DEVIATION TO TIMELINE

There are several dynamical changes from the week 6 to this week. The original project had planned that the raw question could be generated before week 8 which is a key time point that could guide our project further developmentation. The original timeline has been shown in figure below. However, there are some non-technical problems blocking the completion of the milestone. The first and most urgent problems that we faced are we could no get the raw questions. According to the original plan, raw question generation would depend on analysing general questions which we collect in week 4 and provided some suitable parsing or grammar structures for creating templates. However, until last week, we still did not decide the format and the intents and slots. That is the main reason this key part delay. Besides this other reason had resulted in this dead delay, that is the database design delay postponed the raw question generation. Because if we did not access the actual data, our team could not quickly replace the blanks that we retained in templates with actual course information.

The plan towards to this part will be deviated to the end of week 9, which is the end of this week. During this week, almost all the question templates has been defined and several general interfaces which is used to dynamically add templates and slots have been released. Therefore, for the raw question generation, it will be finished by the end of week 9.

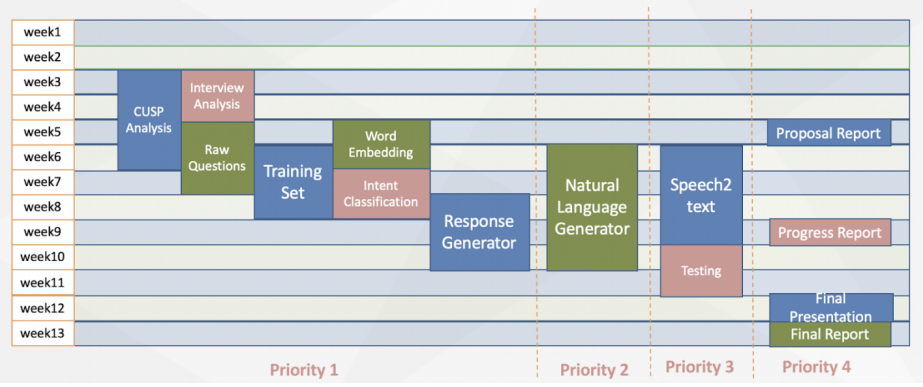


Figure 42 Original TimeLine

As for the implementation of database, several problems had been found, which could impact the developing the response generator and the natural language generator part. One prominent problem related to the database is memory-intensive query issue. The problem happen when fuzzing querying start. Meanwhile, in week 7, we found some metadata were also needed to be stored in the database due to some high level query that user arised. For example, user could ask a relatively higher level question like “could you tell me assessment information of Machine Learning”. This is kind of question that contains several attributes that needs to be returned. Due to this reason, response generator was postponed to the end of week 10.

In terms of deploying the program on Google Device such as Google Home, the speech to text components and the cloud deployment action has been shifted to the end of the week 10. The reason that we faced recently about the Google Device is that Google does not provide fully API to user to access the Google Device functions. We still not find suitable API to call. The risks of this part is that we could not present our programme on time. Therefore, we also urgently needs to migrate its implementation to the nearest week, which indicates the week 10.

Natural Language Generator (NLG) has been modified to use SQL template due to the limited time reason. Original NLG would use language model plused templates to generate. However, due to the delay of other part, the method to implement NLG part to use logical condition plused templates. These templates will be stored as several object file rather than storing in database like previous plan. Therefore, the timeline to this part is also modified to the end of week 10.

The response generator which indicates rule base system has been changed into another implementation method. That is to use logical system which the rules are statically written into the system, like a configuration file rather than dynamically add and query from rule base system. Therefore, this migration would weaken robustness and expansibility of the system. Meanwhile this part would be speeded up and planned to finished before the end of week 10.

4. MILESTONES & REPORTING

Milestone	Tasks	Reporting	Date
Week-1	Select the project topic Group a team	Client meeting to review the project	01-03-2019

Week-2	Basic understanding of the project	Client meeting to review the work plan	08-03-2019
Week-3	Project architecture design Analyzing course information structure Analyzing degree information structure	Client meeting to review Project requirements Prerequisite structure Interview Questions	15-03-2019
Week-4	Project architecture design Design CUSP database structure	Client meeting to review Interview recording	22-03-2019
Week-5	Proposal Report Due Crawling information from CUSP Create templates for questions generating Compare word2vec and fastest	Client meeting to review Group Proposal Report Database structure Interview summary	29-03-2019
Week-6	Fill keywords to templates to generate Questions Design a intent system based on knowledge database Design a slot tag system based on CUSP database Create different word vector files Compare different STT API	Client meeting to review CUSP database containing course and degree information with hierarchy structure A file of word vector (training dataset)	05-04-2019
Week-7	Fill keywords to templates to generate Questions Design an intent system based on knowledge database Bi-LSTM + Attention model Applying STT API and testing	Client meeting to review A raw question set	12-04-2019

Week-8	Labelling intent & Data preprocessing Cross-Validation/ Evaluation Design Logical system	A set of intent & a labelled training set A tiny program that can convert text to speech or contrariwise	19-04-2019
Week-9	Progress Report Due Design Asking part	Project Progress Report	03-05-2019
Week-10	Design Response part Logical processing test Applying STT API and testing Peewee templte testing A language generator	Testing	10-05-2019
Week-11	Testing		
Week-12	Final Presentation	Demo	24-05-2019
Week-13	Final Report (thesis)	Final Report	31-05-2019

5. REFERENCE:

Clark, K., Luong, M. T., Manning, C. D., & Le, Q. V. (2018). Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*.