THE UNIVERSITY OF HONG KONG

DEPARTMENT OF COMPUTER SCIENCE

Final Year Project Report

[Industrial Based]

AI Bot to Make the Best Decision for the Customer

COMP4801 Final Year Project

Supervisor: Dr. Kenneth Wong

Team members:

Chen Xusheng, Michael 3035028520

Huang Kai, Kevin 3035086340

Tan Zhanwen, Francis 3035028518

Client: Microsoft Hong Kong Limited

Abstract

As an industry giant in IT products, Microsoft produces a large number of digital products that are quite alike in many aspects, which makes it difficult for customers to distinguish among similar models and select the most suitable one. This project is an industrial-based project initiated by Microsoft Hong Kong Limited, which aims to solve the aforementioned problem by developing an AI Chat Bot that communicates with customers using online conversations in natural languages and recommend them with the particular product that fits their needs best.

<u>Acknowledgements</u>

The progress of the project *AI Bot to Make the Best Decisions for the Customer* could not have been made without the help of following people. We would like to express our most sincere gratitude to them.

Dr. Kenneth Wong, supervisor of this project, for giving directions and support throughout the designing and implementation of this project.

Dr. Anthony Tam, our second examiner, for giving us precious suggestion on the project and help us with all the communications.

Mr. Samson Lee, contact person from Microsoft, for helping us with the roadmap and providing useful materials.

Microsoft Hong Kong, for providing instruments and business data for our implementation and model training.

Table of Contents

List of Figures 5

List of Tables 6

Abbreviations. 7

1 Introduction 8

2 Background 10

2.1 Natural Language Processing 10

2.2 Machine Learning. 11

3 Previous work. 12

3.1 AI Bots 12

3.2 Customer Service. 13

4 The Goal of Our Project 14

4.1 The Tradeoff between Accuracy and Concision 14

4.2 The Requirement of Our Project. 16

4.2.1 Objective 16

4.2.2 Scope 17

5 Background: Microsoft APIs 18

5.1 Microsoft Bot Framework 18

5.2 Microsoft Cognitive Service APIs. 19

5.2.1 Language Understanding Intelligent Service (LUIS). 19

5.2.2 Translator API. 21

5.3 Microsoft Azure Machine Learning Studio 22

6 Challenges. 23

7 Overview. 25

7.1 Main Components. 26

7.1.1 The Bot Framework. 26

7.1.2 The LUIS model. 26

7.1.3 The Azure Machine Learning Model. 27

7.1.5 Database 27

7.2 Logic Flow of the bot. 28

7.2.1 The archived Bot. 28

7.2.2 Recommendation Based on Input. 29

7.2.3 User Profile Based Recommendation. 30

8 Design and Implementation. 31

8.1 LUIS Module 31

8.2 Text Translator API. 34

8.3 The Machine Learning Model 35

9 Evaluation. 39

9.1 The performance of LUIS Model. 39

9.2 Machine Learning Studio Model Evaluation. 40

9.3 User Study. 41

10 Discussion. 42

11 Division of Labour. 43

12 Conclusion. 44

Reference 45

List of Figures

Figure 1	Machine Learning Examples	P.12
Figure 2	Apple Online Support	P.13
Figure 3	Microsoft Online Support	P.13
Figure 4	Microsoft Bot Framework	P.18
Figure 5	Language Understanding Intelligence Service	P.20
Figure 6	Sample Usage of Translator	P.21
Figure 7	Example Machine Learning Model.	P.23
Figure 8	Main Components of the Bot.	P.25
Figure 9	Sample LUIS labelling	P.32
Figure 10	LUIS Dashboard	P.33
Figure 11	The Logic for Connecting with Translator	P.34
Figure 12	The Recommender Engine	P.36-37
Figure 13	Recommender Engine as Web Service	P.38
Figure 14	LUIS Performance on Gaming Device	P.39
Figure 15	LUIS Performance on 2-in-1	P.40

List of Tables

Table 1	Performance of Recommender Engine	P.41
Table 2	User Study Result	P.42

Abbreviations.

AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
DBMS	DataBase Management System
KNN	K-Nearest Neighbors
LUIS	Language Understanding Intelligent Service
NLP	Natural Language Processing
RAM	Random Access Memory
SDK	Software Development Kit
SQL	Structured Query Language
SVM	Support Vector Machine

1 Introduction

The development of technology gives people more choices on various electronic devices, but it also increases their confusions while selecting the right devices. Many customers can hardly distinguish the differences among tens of models and find out the one most suitable for them[1][2]. Moreover, the transfer from on-site shopping to online also makes the condition worse by reducing the possibility for a customer to seek for help from shop assistants.

In order to solve this problem, Microsoft Hong Kong and the University of Hong Kong started the project *AI bot to make the best Decision for the Customer*s. The goal is to build an Artificial Intelligence chatting bot that can help customers to make their purchase selections. The bot will be accessible from Skype, Facebook and Microsoft's online store. Customers can talk to the bot directly and get recommendations. The user experience will be similar to talking to a human shop assistant.

Different from the bots widely used on the internet, ours will stand out because of three reasons. First, current bots usually communicate with users by giving multiple choices at each step and letting users to select from them. It is hard for a customer to clearly describe his requirement in this rigid way. Moreover, if a customer can answer each question, that means he already have a clear idea about what to buy, and thus no need to talk to the bot at all. Instead, we leverage the technology of natural language processing, such that the user can really “talk” to the bot and explain his requirement little by little. We will introduce more about this technique in §2.1.

Second, current bots usually have a pre-defined decision tree. It is hard for a designer to take every scenario into consideration because of the limitation of human mind and the increasingly fast evolution of technology. Instead, we leverage the technology of machine learning. The bot will learn to deal with different conditions from a huge amount of real-world data. The bot can also update itself while running, which can help it to keep it up to date.

Third, our bot can be accessed from various platforms, including Facebook Messenger, Skype and Microsoft online store. User can access the bot from anywhere using anyway. The purchase history and preference will be recorded for registered users in order to give customized recommendations.

The remaining of this report is organized as follows. §2 introduces background on natural language processing and machine learning. §3 examines the existing works in this area. §4 covers the goal of the project. §5 introduced the APIs that are going to be used. $6 listed out the challenges we have met. §7 shows the overview of the bot. §8 detailed the design and implementation of the project. §9 evaluates the performance of the bot. §10 discuss about the limitation of our bot. §11 shows the division of labor and §12 concludes.

2 Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is the study of letting computers understand human languages[3]. Without NLP, human language sentences are just a series of meaningless symbols to computers. Computers don’t recognize the words and don’t understand the grammars. NLP can be regard as a “translator”, who will translate human languages to computer understandable information.

Traditionally, users need to follow well-defined procedures accurately, in order to interact with computers. For example, in Linux systems, all commands must be precise. A single replace of one character or even a space can have significant difference. However, the emergence of NLP is changing the way of interacting. Apple Siri [4] and Microsoft Cortana [5] have made it possible to give command in everyday languages and is changing the way of interacting.

In this project, we are going to use the API provided by Microsoft called Language Understanding Intelligent Service (LUIS)[6]. It contains a bunch of well-developed REST APIs. We will take time to read the documentations and more details will show in future reports.

2.2 Machine Learning.

Machine Learning (ML) is an area of computer science that "gives computers the ability to learn without being explicitly programmed"[7]. The parameter of the formulas is calculated from the data, rather than defined by the programmer. Two most common usage of ML is Classification and Regression. As shown in figure1[8], Classification means to categorize different types of data, while Regression means to find a way to describe the data. Basic ML program will have two stages, *fitting* and *predicting.* In the fitting stage, the program will be given a large set (at least thousands) of data. The program will try to adjust its parameter based on some statistical models, in order to make it “fit” the input data best. In the predicting stage, the program will give a prediction for a new input based on the parameters it just calculated out. For example, the famous Iris flower dataset [9] contains the measurement of several features of three different species of flowers, such as the length of sepals and petals. A well-defined ML program can learn the pattern behind this feature and give prediction accordingly.

In this project, we will use the Microsoft Azure Machine Learning Studio [10]. It is a platform for implementing and testing ML models. The backend is on Microsoft’s cloud service, which make the calculation much faster than personal computers.

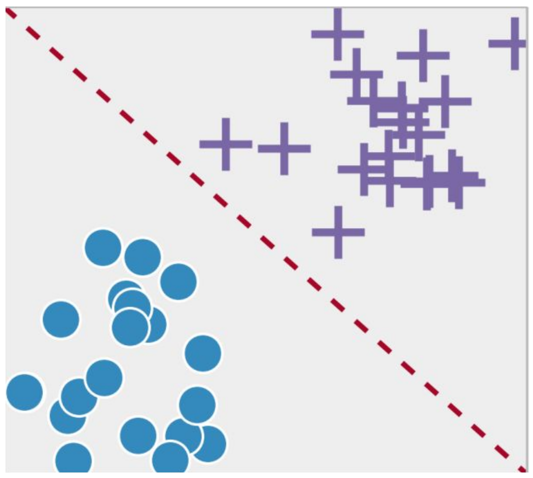


Figure1a: Classification

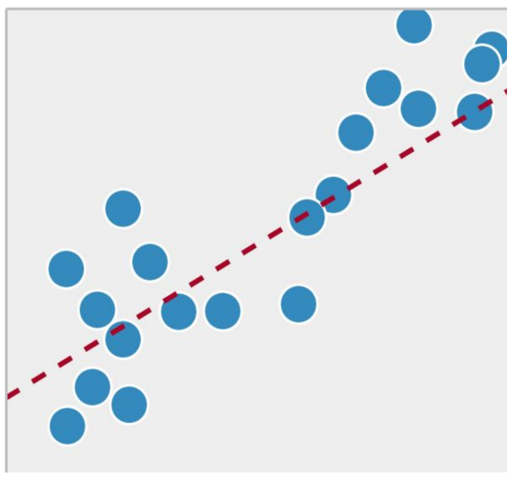


Figure1b: Regression

3 Previous work.

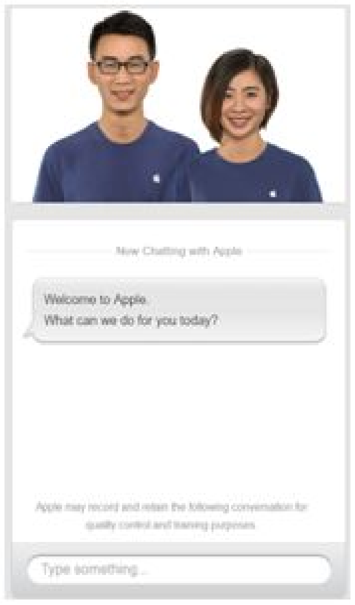
3.1 AI Bots

AI bot (or Chatbot) is a computer program that conduct conversation via audio or textual messages, and is widely used now. It typically use Natural Language process techniques to analyze input sentence and generate outputs. There are currently two trends in the development of AI bots. The first one is to be as *Human-Like* as possible and try to pass the Turing Test. One of the most famous contests in this area is the annual Loebner Prize.

Another trend is trying to provide help to users, with users knowing that definitely they are talking to a bot and the bot can help them finish some specific jobs. Apple's Siri, Microsoft's Cortana and Google now can be categorized as this type. Because the conversation topics are limited in a relatively narrower area, they can achieve an acceptable accuracy and can be used in daily life.

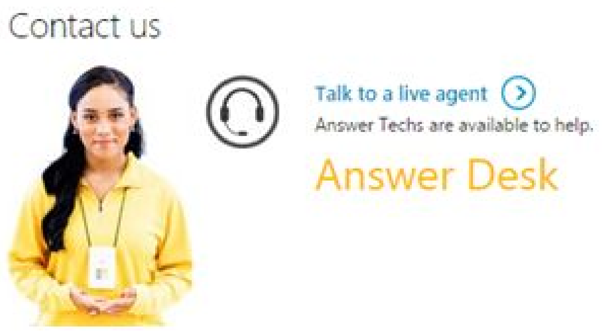
3.2 Customer Service.

Currently, even big companies like Apple and Microsoft still haven't use AI bots in customer services. *Figure 2* [2] and *Figure 3* [3] shows the customer service interfaces for Apple and Microsoft


Now Chatting with Apple

Welcome to Apple.
What can we do for you today?

Apple may record and retain the following conversation for quality control and training purposes.


Contact us

Talk to a live agent

Answer Techs are available to help.

Figure 2: Apple Online Support Figure 3: Microsoft Online Support.

For both of them, they are actually annual work. There human beings on the other side of the chat box. As a result, the service hour is limited to regular office hours (9am-5pm and 9am-9pm for Apple and Microsoft accordingly). Moreover, one human being cannot serve multiple customers at a time and takes around 20 seconds latency for each question.

AI bots are not widely used in customer service for several reasons. One of the most reasons is the accuracy. It will be annoying that one customer comes to ask for some emergency help but the bot gives a wrong answer or cannot understand his idea after several rounds. Moreover, when people are talking, their expressions can vary tremendously, different from giving simple commands like “*set an alarm”* . Our project will be restricted in product recommendation, and may have a acceptable accuracy is training data is enough.

4 The Goal of Our Project

As an industrial based project, the goal of the project comes from our client: Microsoft Hong Kong. Mr. Samson Lee is the Microsoft side’s contact and is responsible for the project but the requirement actually comes from the business side, the actual user of this bot.

4.1 The Tradeoff between Accuracy and Concision

The ultimate goal of the project is to build an easy to use talking bot that can talk with customer and provide recommendation based on their requirements. However, what is “easy to use” is actually negotiable.

There is a tradeoff between the *recommendation accuracy* and the *concision of conversion*. If we want to make a more accurate recommendation, the only way is to ask more question to the customer and gather more requirements. Imagine when you go to a shopping mall and ask the shop assistant to recommend some items for you. She will ask a lot of questions about your usage, your expectation of price and even your preference of color before he can make recommendations for you.

This is the same for the bot. Although the machine learning behind the bot can extract useful information from past customers by learning the feature and relationships, the bot cannot give useful recommendation if it does not know about the customer. In order to make a good recommendation, besides a good machine learning algorithm and the abundant training data, the bot must have the data about the customer. For a registered customer, some of the information may be already saved in the database，but for a new customer, the only way is asking questions.

However, asking too many questions will make the conversation really lengthy. Different from a face to face talking, online chat can never be that efficient. The input sentence need to be parsed and analyzed, causing multiple times of network round trip time, making the response latency really large. Moreover, from the customer's perspect, typing is also much slower than talking.

It will be annoying if the customer simply want some basic recommendation, but the bot is asking a lot of lengthy questions, without prompt response, making the conversation last for minutes. Furthermore, the customer may not know how to answer some questions (e.g., “what cpu frequency do you want, xx GHz or yy GHz”) or may not want to answer some questions (e.g.,”what is your monthly income”) because of privacy reasons. An experienced shop assistant can discover the embarrassment immediately but the bot, on the other side in the network can never identify it, until the customer typed in “I don’t know” or “I don’t want to answer this question”.

4.2 The Requirement of Our Project.

4.2.1 Objective

As a result, during the first half of our project, we were working with Mr.Samson and we have done a prototype of an “ask-everything” bot. The bot tries to get all the information, including the preference on CPU, tentative usage, expect price range, etc., and make recommendation if all the information is collected. This was a prototype that integrates all the APIs and web service, as a deployable web service.

However, we had a meeting with Microsoft’s business team in Jan 2017, and our requirement was changed to a more user friendly way. Rather than asking all the requirements and information one by one to the customer, this time we just let the customer to directly point out their intent by sentence like “ I want a computer to play large 3D games” and give recommendation sparsely. The devices are categorized into different groups, and the recommendation logic is just a simple link between intent and groups.

4.2.2 Scope

In this project, we are going to build an LUIS model for Cantonese language, because Cantonese is the most widely used language in Hong Kong. The logic is the same for other languages, but because a lacking in training data samples, we are going to only use Cantonese in our project.

For this project, in order to improve accuracy, we are going to restrict the query types to *"Purchase"* and *"Complaint"*. They will be handled differently but be analyzed in the same language model. For a purchase intention, the bot will analyze what type of device the customer wants to buy and give recommendations. For a complaint recommendation, on the other hand, the bot will only save the contents of complaint into database for a specialist to handle. In this project, we are not going to give specific response to complaint.

The device for recommendation is should be a dynamic database reflecting a real time price and specification of all the available devices. However, due to some confidential reason, Microsoft cannot provide this link to us, but give us a static table file containing some sample devices, together with type and price, and let us create a static database based on the table.

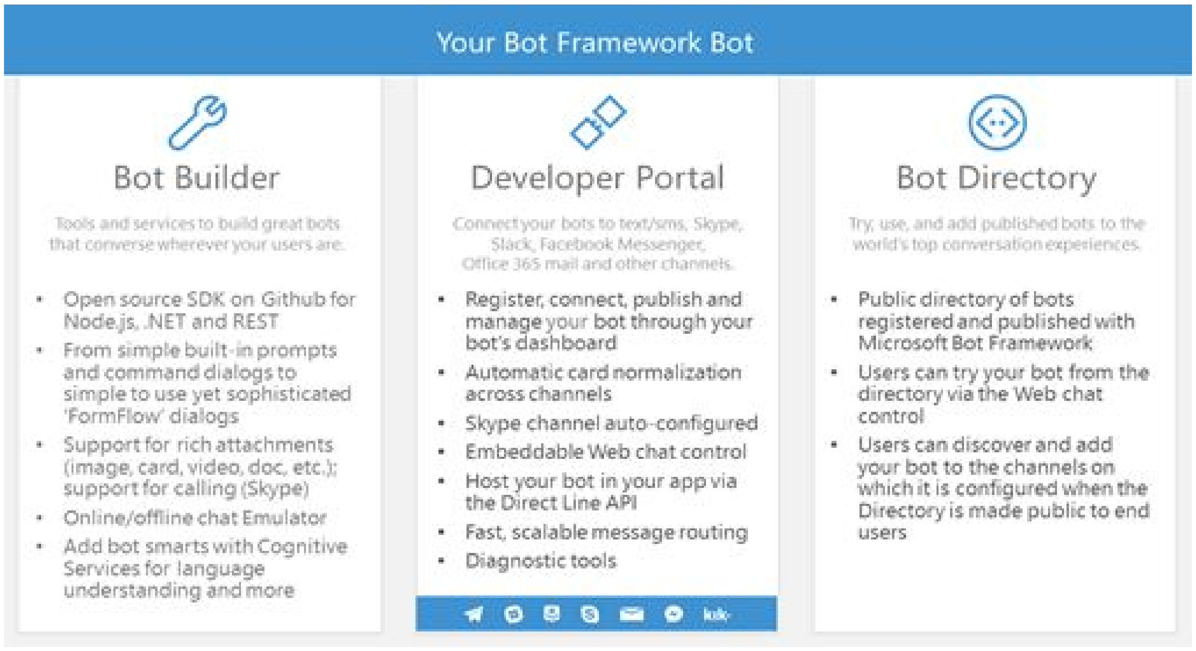
The bot is going to be deployed in three different places, including Bot framework portal, Skype and Facebook messenger. Other platforms, e.g., Kik, Slack are optional and depends on future usage of Microsoft side.

5 Background: Microsoft APIs

This project will be highly rely on Microsoft's APIs and be built in C# .Net framework. This section will introduce some key APIs used by us to build our bot.

5.1 Microsoft Bot Framework

Our bot is built on top of Microsoft's bot framework. It is a SDK provided by Microsoft to build general chat bots. With built-in classes to handle conversation. With the help of the SDK, we can focus on the logic of handling messages but forget about how the message is transferred between the chat box and the backend engine. It also defined a common gateway to connected with different front-end interfaces, such as Skype or Facebook.



Your Bot Framework Bot

Bot Builder

Tools and services to build great bots that converse wherever your users are.

- Open source SDK on Github for Node.js, .NET and REST
- From simple built-in prompts and command dialogs to simple to use yet sophisticated 'FormFlow' dialogs
- Support for rich attachments (image, card, video, doc, etc.); support for calling (Skype)
- Online/offline chat Emulator
- Add bot smarts with Cognitive Services for language understanding and more

Developer Portal

Connect your bots to text/sms, Skype, Slack, Facebook Messenger, Office 365 mail and other channels.

- Register, connect, publish and manage your bot through your bot's dashboard
- Automatic card normalization across channels
- Skype channel auto-configured
- Embeddable Web chat control
- Host your bot in your app via the Direct Line API
- Fast, scalable message routing
- Diagnostic tools

Bot Directory

Try, use, and add published bots to the world's top conversation experiences.

- Public directory of bots registered and published with Microsoft Bot Framework
- Users can try your bot from the directory via the Web chat control
- Users can discover and add your bot to the channels on which it is configured when the Directory is made public to end users

Figure 4: The Microsoft Bot Framework.

The framework is only responsible for transferring conversation contents. After receiving an object represent an input from user, we can get the input string or images together with the metadata such as timestamp or user id. We can freely analyze the input sentence and create another object representing output and send back to the user.

During our test in HKU’s network on its own bot portal, the average latency is second level, a simple echo-server takes around 2-3 seconds to get response. While testing in Skype interface, the latency is around 2-6 seconds.

5.2 Microsoft Cognitive Service APIs.

Microsoft Cognitive service APIs are a bunch of services developed by Microsoft to enable Artificial intelligence in programs. Most of the services are still in the beta version, the APIs are not stable and documents are not complete, which becomes one of the challenges in our project.

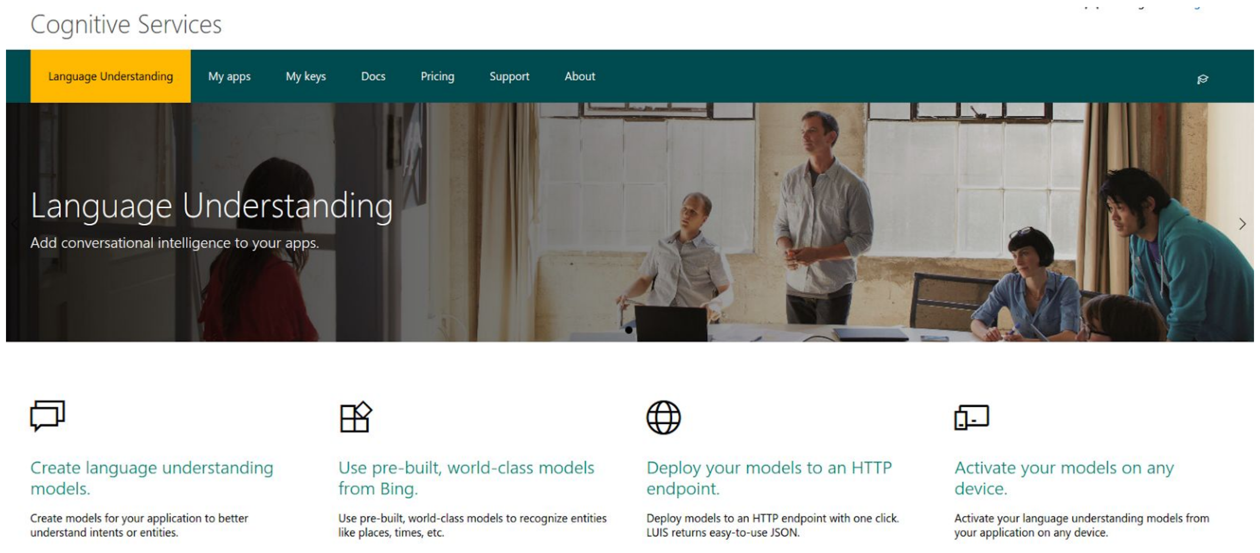
In this project, we used the Language Understanding Intelligent Service (LUIS) to analyze user’s natural language inputs and use Translator API as associate to the LUIS.

5.2.1 Language Understanding Intelligent Service (LUIS).

LUIS is a service built by Microsoft based on machine learning algorithms. Users can create language model on the platform, train it with labeled data and deploy the model as a web service. There are two important concept in LUIS, namely “*intent*” and “*entity*”.

Intent describes what intention the speaker want to convey while saying the sentence. Each input sentence will be categorized into one "intent", which is similar to the concept of "class" in supervised learning area.

Entities are the useful information detected in the input sentence. Each entity will be represented as a pair of "type" and "value", with "value" being a specific substring of the input and "type" showing what kind of useful information the substring shows. For example, in our project scope, if customer says "I want to buy a laptop for text editing". The intent can be categorized as "purchase" and the entities can be two pairs as *{product: "laptop", usage: "text editing"}.* In LUIS, the machine learning algorithm and training logic is handled by the platform, while users need to define intent and entities clearly and provide sufficient labeled training data.



Language Understanding

Add conversational intelligence to your apps.

Create language understanding models.

Create models for your application to better understand intents or entities.

Use pre-built, world-class models from Bing.

Use pre-built, world-class models to recognize entities like places, times, etc.

Deploy your models to an HTTP endpoint.

Deploy models to an HTTP endpoint with one click. LUIS returns easy-to-use JSON.

Activate your models on any device.

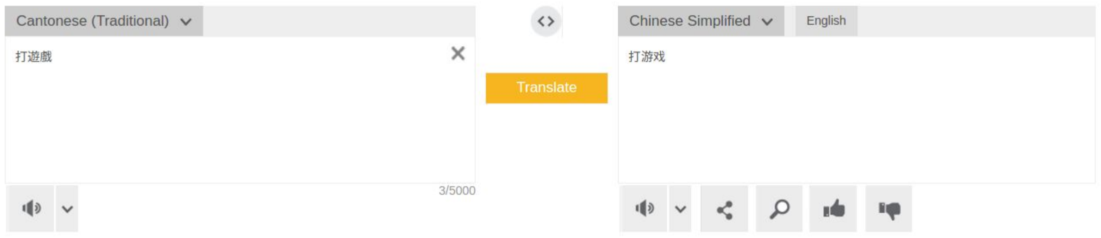
Activate your language understanding models from your application on any device.

Figure 5: The LUIS Service Page.

Besides basic idea, LUIS has two special types of entity: *hierarchical entity* and *composite entity*. Hierarchical entity employ the idea of *generic* and *specific* entities. Child entities are of different types while sharing the same characteristics. For example, in our project, we have a parent entity called “feature” and several children entities like “cpu type”, “memory size” and “disk size”. Composite entity shows the associative pattern among entities, which can be thought of as a hasA relationship. For example, the sentence “I want 3 medium pizza” has three entities: “number”, “size”, “item”, while we can define a composite entity as the type of “order” to put them together.

5.2.2 Translator API.

The translator API is a pre-built web service that can do translation from one language to another. It is not a generic necessity for us to use translator in our project because our scope is limited to Cantonese only.



打遊戲

打游戏

Figure 6: Sample Use of translator.

However, one limitation in LUIS forced us to it. Current LUIS only supports Simplified Chinese Character, and had a very poor performance if we put in Traditional Chinese Characters. As a result, the only possible way for us is to use Simplified Chinese in the backend but expose an interface in Traditional Chinese. When a customer input a sentence in Traditional Chinese, the sentence will be handled by the translator first. The figure above shows an sample usage of the translator API.

5.3 Microsoft Azure Machine Learning Studio

Microsoft Azure Machine Learning Studio is a platform for users to build machine learning predictive models. It has a drag-n-drop graphical interface to represent the logic flow of a machine learning model.

A trained model can be deployed as a web service, taking data as input and give predication based on the trained model. For example, the figure above is a sample model used to compare two models for detecting hand-writing English Characters.

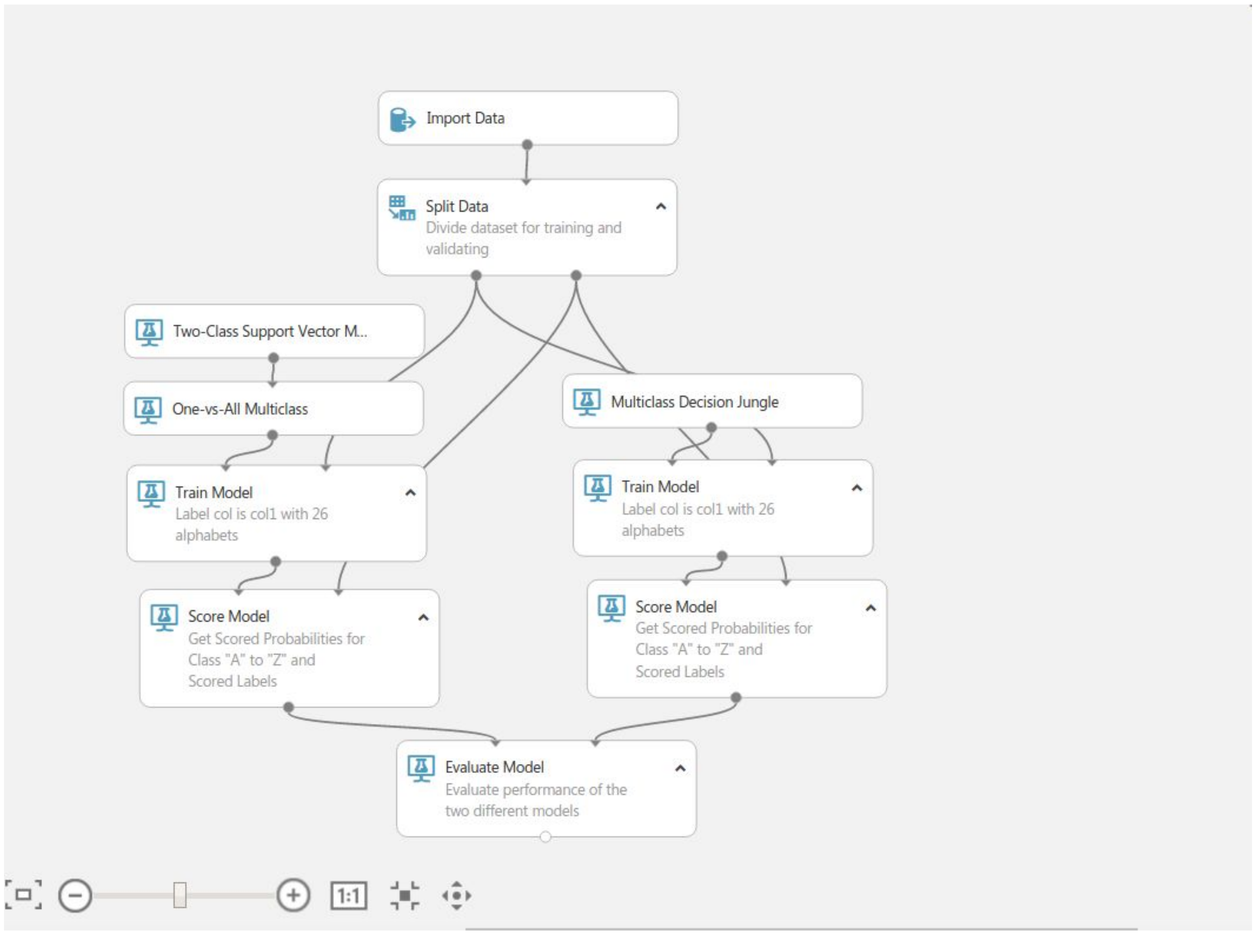

Import Data
Split Data
Divide dataset for training and validating
Two-Class Support Vector M...
One-vs-All Multiclass
Multiclass Decision Jungle
Train Model
Label col is col1 with 26 alphabets
Train Model
Label col is col1 with 26 alphabets
Score Model
Get Scored Probabilities for Class "A" to "Z" and Scored Labels
Score Model
Get Scored Probabilities for Class "A" to "Z" and Scored Labels
Evaluate Model
Evaluate performance of the two different models
1:1

Figure 7: A sample of model on Machine Learning Studio

6 Challenges.

This section covers the challenges we met throughout the project, including both software engineering aspects and technical aspects.

1. **Changing of requirements.** As an industrial project to build a product, we must follow the requirement from the user. However, because the project's goal is to be used by the business team, but it is responsible by the technical team, the requirement changed a lot in the middle after a meeting with the business team. The business team want a simple bot that can give recommendation immediately. We had to archive what we had done before and build a new one.

2. **Lacking of training data.** The quantity and quality of training data is critical to the performance to a machine learning model. However, because some confidential and privacy reasons, the business team cannot provide enough data for us, and we had to make up data by our own. For the machine learning model, we generate some fake data based on our daily life experience, which is really biased, although with a good accuracy on the fake dat.

3. **Unstable API version.** Because API service we are using are still under development, and we cannot fix to a version for the API, the API may changes overtime. Moreover, there are inconsistencies between the APIs and their documents or sample codes.

4. **Not familiar with the C# language and .NET framework.** None of the three of us has previous knowledge with C# language. Programming in a new language in such a huge framework is quite challenging for us at the beginning of the project. However, when we comes to the later phases, we are more used to that.

5. **Azure account.** This is not a technical challenge but administrative. Because all the components of the project need to be deployed as web service, we must hold it in Azure. However, the education account is quite limited and cause us several days to handle the account problem.

7 Overview.

This section shows the overview of our project, including the main components, how they work together and the work flow.

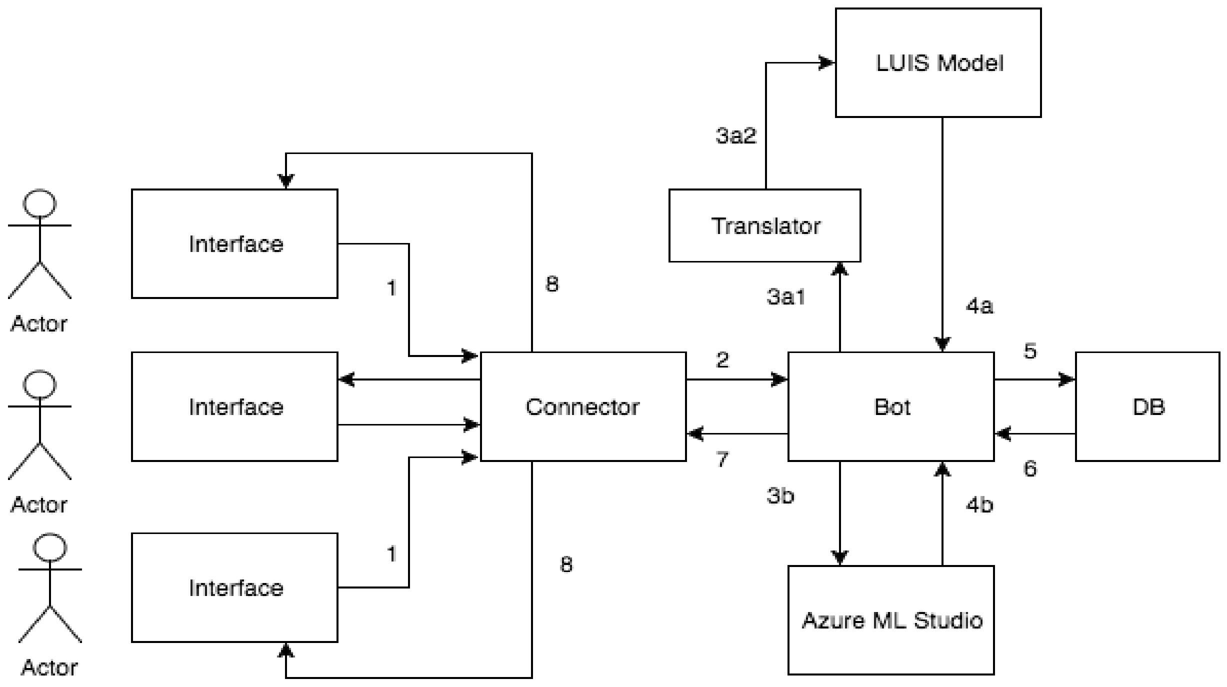


Figure 8: The workflow of our bot,

7.1 Main Components.

7.1.1 The Bot Framework.

As shown in the figure, there are six components in our project. The *interfaces* are the front end chat box for user to talk to the bot, which can be the Bot Portal, Skype, Facebook, etc. The *connector* works as a common gateway for all the interfaces. The outbound side calls different APIs to different front end, but the inbound APIs kept the same for our bot to connect. Fortunately, this connector has already been implemented by the bot framework SDK, we only need to rightly configure them.

The *bot* part contains the main flow control of our project. It is responsible for redirect the input to different models, parse the return values, and determines what to do next. It is also connected to the database to retrieve and update values.

7.1.2 The LUIS model.

The *LUIS* model is a standalone web service. We train it and deploy it in its own web interface and connect it through REST APIs. The input is exactly the sentence the user typed in and it will return the analyzed result as a json format. Our bot will parse the json format to determine how to make replies.

However, because the LUIS currently does not support Traditional Chinese Characters, we had to train the model in Simplified Chinese Characters and do a translation when a user input some sentence. The translator is a close-sourced and we cannot change the function. It will be optimal if it can give a one-to-one mapping between Traditional

Chinese Characters and Simplified Chinese Characters. However, during the test, we found that it will do some meaning based translation. As a result, for all the input sentences, we pass them through the translator first, and train the LUIS model using the output of the translator.

7.1.3 The Azure Machine Learning Model.

The *Azure Machine Learning Studio Model* contains a trained model on a recommendation engine. After meeting with the Microsoft, they asked us to implement a recommendation engine based on the customer's user profile, such as age, gender or occupation. The training data is the preference of different person's preference of different type of devices, and we are going to use the profile of a new user to give recommendations.

Unfortunately, because of the privacy reasons, we cannot get the data during the development phase. We had to make up some fake data to demonstrate the workflow, and it may get connected to the real data set if the business team accepted our bot.

As a result, the recommendation of the Machine Learning Model seems very biased. For example, it will recommend a young boy with a gaming device, not matter he really like playing games of not.

7.1.5 Database

The database contains all the device models for recommend by the bot. As mentioned previously, because we are still implementing a prototype, the database is not dynamic.

The catalog and prices will be fixed through all time. However, we try to make the codes to connect wil database as an independent module, if the bot is actually connected with a dynamic later, the code logic to be changed will be kept minimal.

The database also holds the complaint and feedback from a user. The complaint is used for the customer service people to follow up. The feedback actually means the whether the customer likes the recommendation. It can be used as training data to the machine learning module to improve the accuracy.

7.2 Logic Flow of the bot.

In this section, we are going to demonstrate the logic flow of the AI bot by two sample usage. In the first half of our project, we built a bot to collect the requirement of a customer in all perspective. Although that bot is archived, that represent some of our first understanding of the bot. I will briefly introduce the logic in this section. After that, I will use an example to show how the bot does recommendation based on user input and how the bot does recommendation based on user profile.

7.2.1 The archived Bot.

The archived bot listed some specification that the user can destinate, such as the CPU, RAM size, disk size, graphic cards, etc. Actually these fields comes from the database given by the Microsoft side. Each device has all the columns below, and thus we can select device based on some criteria.

The bot assumes that a customer knows the specific requirement of the device they want to buy. It will try to extract the requirement specification from the user's input and make recommendation accordingly. For example, if the customer says, "I want to buy a laptop with 8GB memory, price lower than 8000", the bot will know the requirement on RAM size and price, which comes from the analysis of LUIS.

However, the bot needs to ask all the fields before it can make any recommendations. In the previous example, the bot will ask questions like "Do you have preference on brand/disk/graphic cards?" one by one. This is really annoying, especially for a user without computer knowledges. We were trying to improve the user experience of that bot but the requirement changed after we met with the business team. The bot is archived after that meeting. Some features such as getting some specific requirement may be useful in the future, but that is out of the scope of this report.

7.2.2 Recommendation Based on Input.

After the meeting with the business team, they gave us some new requirements on the workflow of the bot, which is simpler but more realistic. The assumption that a customer knows all the fields seldom happen in the real world, but what they really care is the usage of the device.

In the latest version of the bot, the bot assumes that a customer will directly point out their usage of their expected device. For example, a customer may say "I want to buy a laptop for playing computer games" or "I want to buy a computer to watch movies".

The business team divide all its devices in the catalogs into five types, namely *gaming device, desktop, laptop, tablet* and *2-in-1*. The business also listed some potential usage of device and assigned one type of devices to each potential usage. The model will randomly recommend 3 devices in a type if it knows the potential usage.

For example, if the user says “I want to buy a device to play video games” (in Cantonese), the sentence will be transfer to LUIS model. The trained LUIS model will mark this sentence with the *intent* of *gaming device.* The bot got the reply from the LUIS and will recommend three random gaming device from the catalog.

7.2.3 User Profile Based Recommendation.

Different group people may have different preferences on electrical devices, which makes it possible for the bot to make recommendation based on user profiles. The bot assumes that a registered user have some of her personal information logged in the system, and based on previous result of the preferences, it can make recommendation based on the information of the new customer.

However, the user profile is confidential and private data. We cannot use these data to do testing or development. As a result, we made up some fake data to train the machine learning model, and use special input in the bot to represent log-in information.

For example, if we input “@70” in the bot. It will be parsed as a user with userID being 70 has logged in and want some recommendation. The bot will go over the database, found that the user 70 is a young boy student, and tell the machine learning model about the information. The model predicts from previous data and recommend *gaming device* to them. This recommendation is quite biased, and may be improved if we can get real training data with more information about a user.

After the recommendation is given, the bot will ask the user to give ratings as feedback. The feedback is saved back into database for future training and refining the accuracy of the machine learning module.

8 Design and Implementation.

This section covers the design and implementation of different module of the bot, which contains the design of the LUIS module, the Translator API and the Microsoft Machine Learning Sutdio.

8.1 LUIS Module

The design of the LUIS module is key to the accuracy of the Natural Language Processing and thus critical to the performance of our bot. Because the learning algorithm is close-sourced, our major work is to define the *intent* and *entity* clearly and labeling the sentence based on our design.

One key consideration is that we must make sure the return value of the LUIS is already "ready to use". The bot cannot understand what is "ram size larger than 4GB", but a composite of “type = RAM, relation = >, value = 4” is easy to implement in the bot’s logic.

For the first half of our project, we defined a LUIS model to analyze a customer’s requirement from different perspectives. Because each sentence can only have one intent, we can only use the *entity* feature to determine the analyzed result. The figure below shows some sample sentence and their labeling. We used entities and sub-entities to determine the types of requirements. We also defined a special entities called relation to deal with the “grater than” or smaller than relations. To notate the association between the relation and requirements, we used composite entities.

The LUIS model turned out to have good accuracy. However, because of the change of project scope, this model has been archived for later usage.

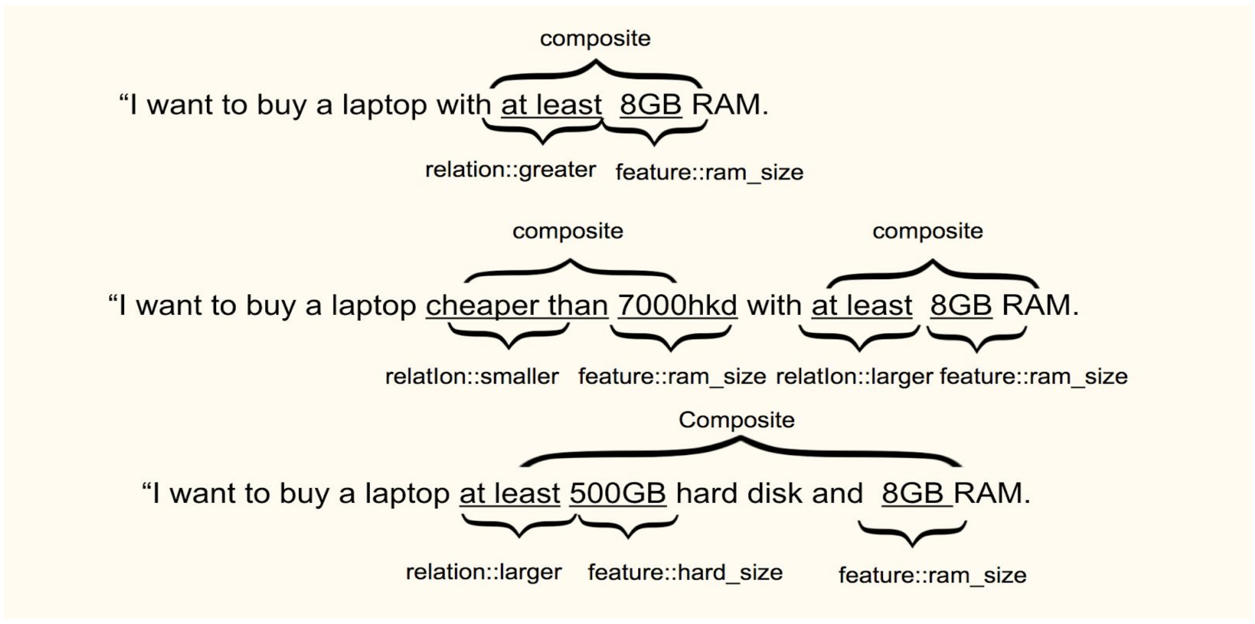


Figure 9: Samples of LUIS utterances.

After the change of scope, we defined a new LUIS model to cope with the new requirement. This time the design becomes even easier because all we need to do is to label different sentence as different intent representing by the device types. We tried to rephrase the data given by the business team and get more training data.

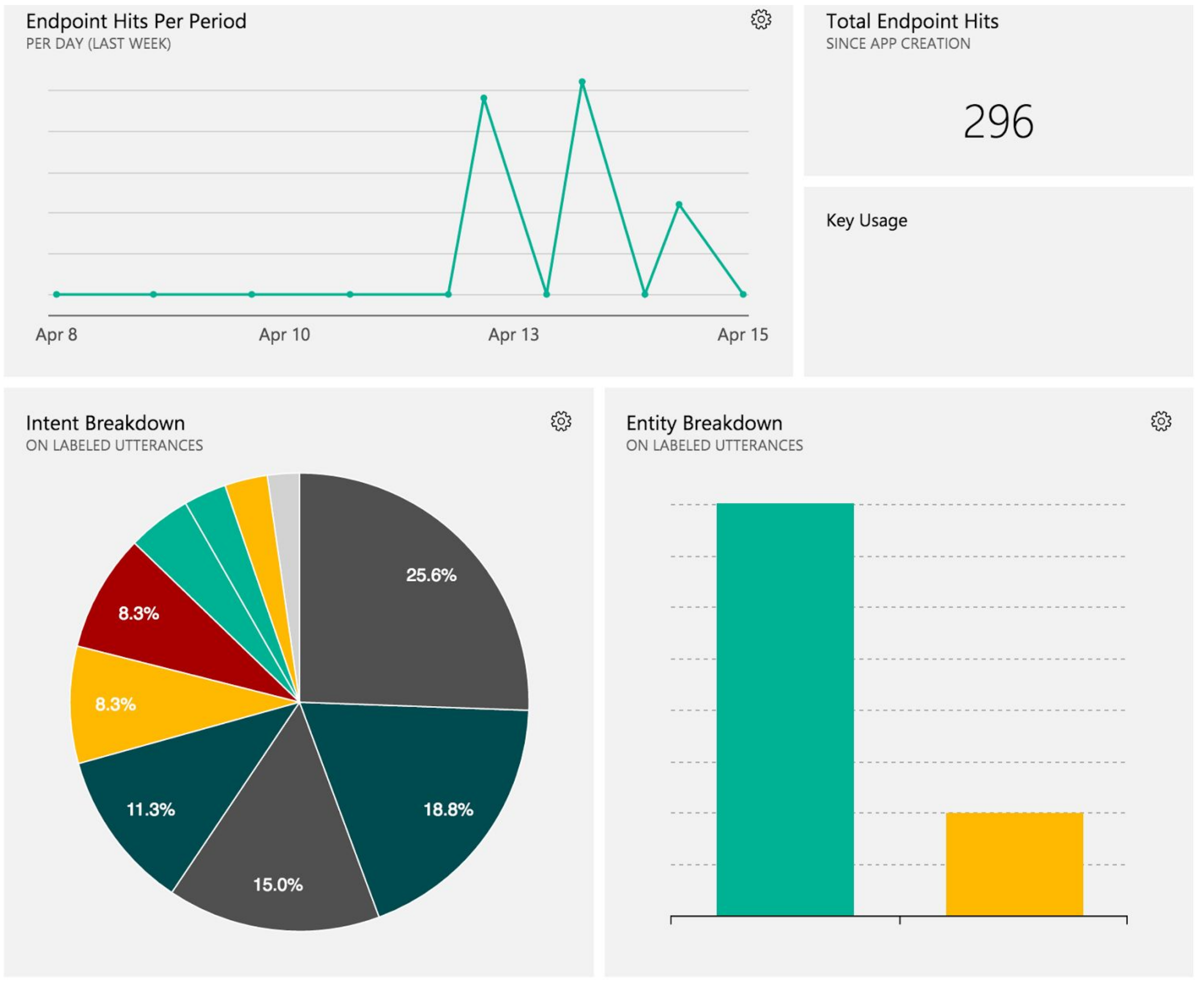


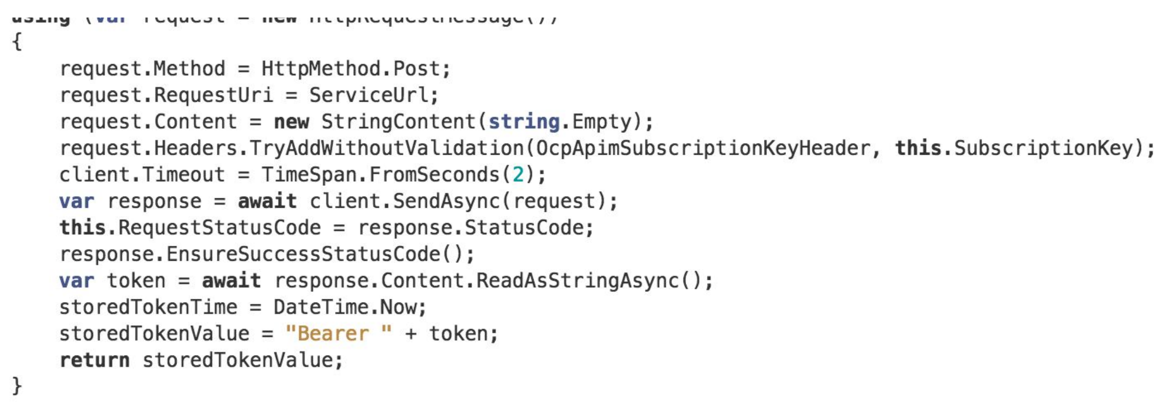
Figure 10: The LUIS dashboard

Apart from potential usage, one customer may also want to express whether she is satisfied with the recommendation. If she is not satisfied, we will try to retry understanding her intent. We added two more intents representing *satisfied* and *unsatisfied* and made some sentences to train the model.

Moreover, we also included intent for greeting, so that the bot will behave naturally. Some random sentence are also included as *none* intent. The bot will simply reply with “I cannot understand” for these inputs. The figure belows shows the dashboard of the LUIS for our new model.

8.2 Text Translator API.

This section introduce how to connect with the Text Translator API. The translator API is a web service provide by microsoft. However, it is not free service and users must subscribe for the usage in Azure portal. Any client must get a token from its own subscription key before asking for service, as shown in the figure below. However, because the translator API keeps developing, none of the documents or sample codes covers this part. As a result, we spent more than one weeks try to connect to the service.



```
using (var request = new HttpRequestMessage())
{
    request.Method = HttpMethod.Post;
    request.RequestUri = ServiceUrl;
    request.Content = new StringContent(string.Empty);
    request.Headers.TryAddWithoutValidation(OcpApimSubscriptionKeyHeader, this.SubscriptionKey);
    client.Timeout = TimeSpan.FromSeconds(2);
    var response = await client.SendAsync(request);
    this.RequestStatusCode = response.StatusCode;
    response.EnsureSuccessStatusCode();
    var token = await response.Content.ReadAsStringAsync();
    storedTokenTime = DateTime.Now;
    storedTokenValue = "Bearer " + token;
    return storedTokenValue;
}
```

Figure 11: The code logic for getting translator token

8.3 The Machine Learning Model

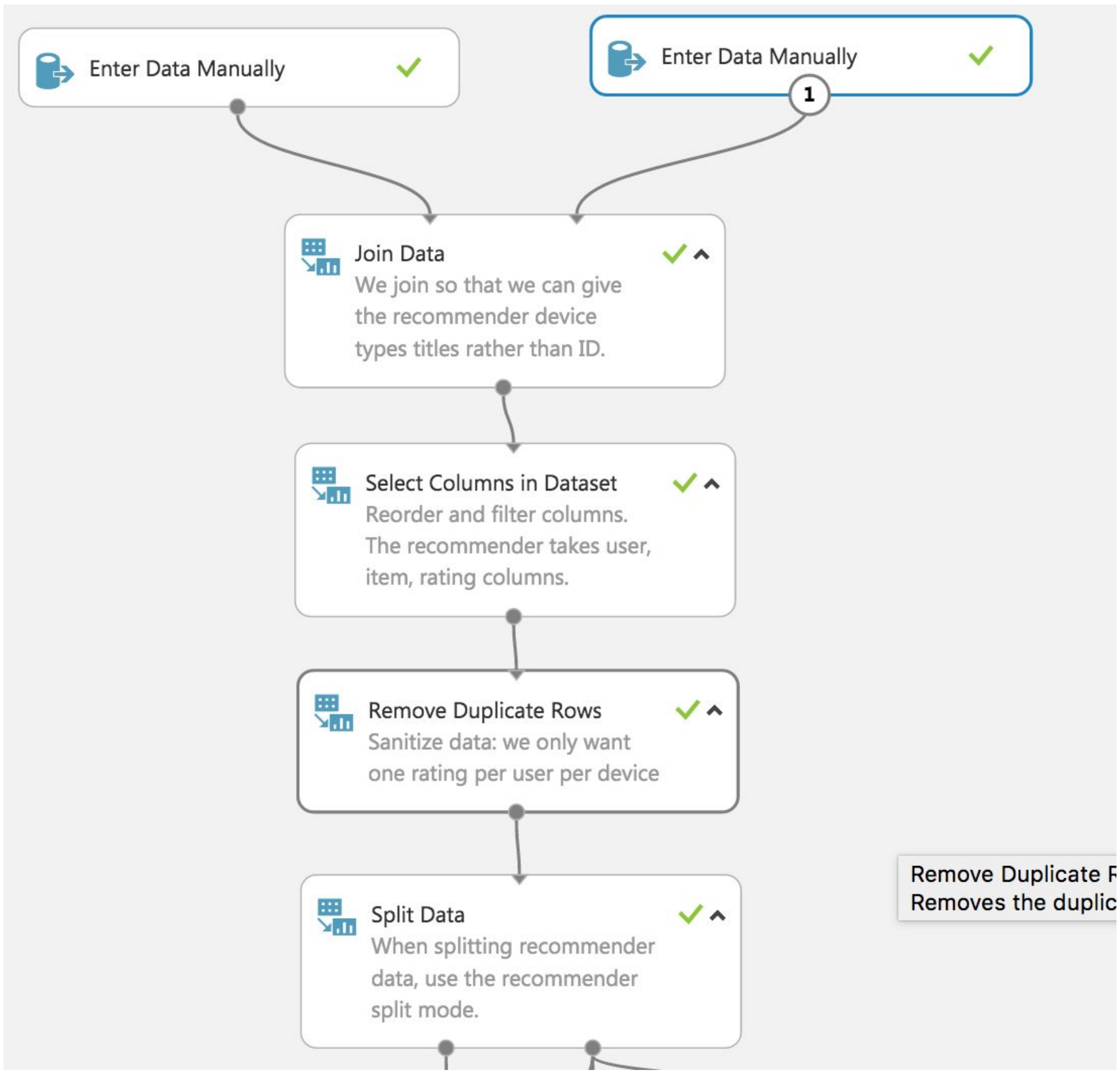
This section introduced how we found used the Microsoft Azure Machine Learning Studio to create a recommendation engine based on user profile.

First, we generate some fake data for the training. The data is divided into three tables. The first one represents the registered users, with one column representing the userID, and several columns representing profile of the user, such as gender, age, occupation, hobbies. The second table are the five types of devices provided by the microsoft. This can be replaced by a catalog with all the deviceIDs and their features. However, because the lacking of data, we only use the five types for demonstration usage.

The third table is the preference of the users to different types. To qualify the preference, we let each user give a rating, from 1 to 10 to each device. The three columns are userID, deviceID, and ratings respectively. The ratings are generally based on our everyday experience, which can be really biased but can show that the module is really working. We only generated the ratings for the first half users as training data, and assumes that the other half are the registered user that will talk to our bot and ask for recommendation.

We also monic the process of tuning learning parameters, but because the data is really biased, it can always give good performance.

The two figures below shows the workflow of the training phase of the machine learning model. (Sorry but the picture is too long and cannot be put into one page).



We join so that we can give the recommender device types titles rather than ID.

Reorder and filter columns. The recommender takes user, item, rating columns.

Sanitize data: we only want one rating per user per device

Remove Duplicate R
Removes the duplic

When splitting recommender data, use the recommender split mode.

Figure 12a: The first half of the recommender engine,

This figure represents the preprocessing of the training data. We first join the tables, remove duplicate keys and then split the data into training data and test data, with a rate of 7:3.

The figure below is the logic for training and testing of the recommender. The rating, together with the user's features are fitted in the recommender to do the training. And the three testing methods are used to evaluate the training model.

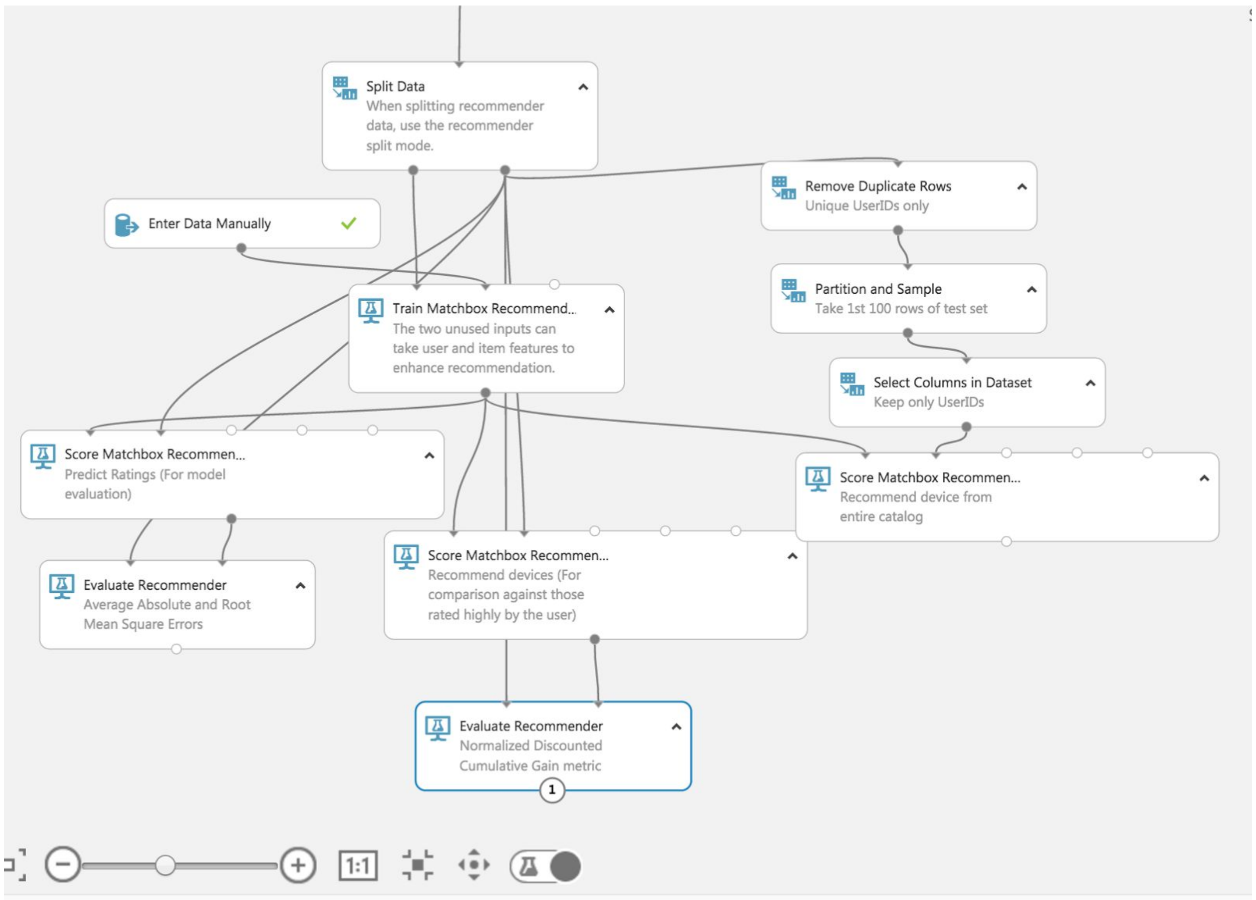
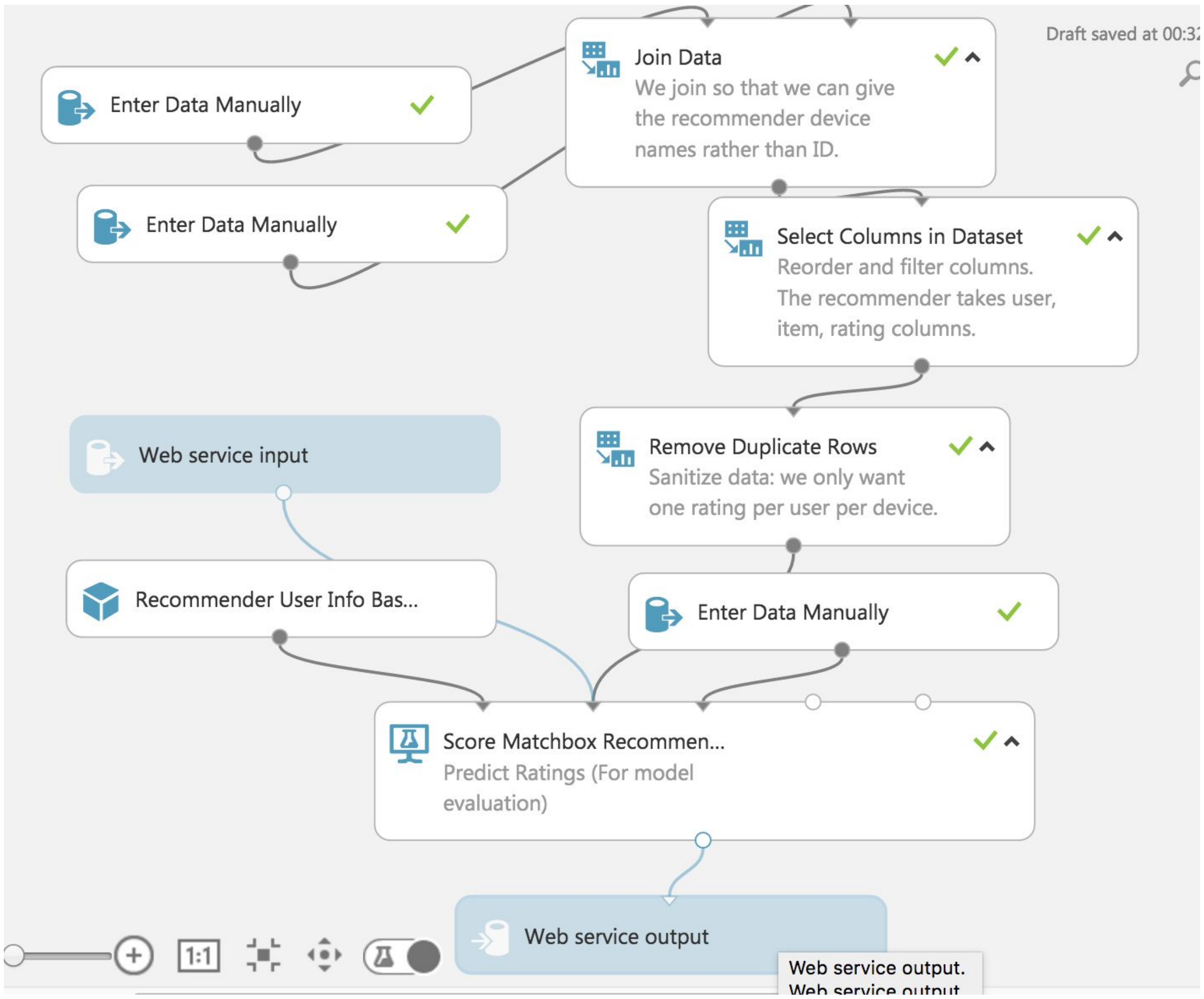

Split Data
When splitting recommender data, use the recommender split mode.
Enter Data Manually
Remove Duplicate Rows
Unique UserIDs only
Partition and Sample
Take 1st 100 rows of test set
Train Matchbox Recommend...
The two unused inputs can take user and item features to enhance recommendation.
Select Columns in Dataset
Keep only UserIDs
Score Matchbox Recommen...
Predict Ratings (For model evaluation)
Score Matchbox Recommen...
Recommend device from entire catalog
Evaluate Recommender
Average Absolute and Root Mean Square Errors
Score Matchbox Recommen...
Recommend devices (For comparison against those rated highly by the user)
Evaluate Recommender
Normalized Discounted Cumulative Gain metric
1
1:1

Figure 12b: The second half of the recommender engine,

After training the model, we deployed the machine learning model as a web service and define the input and outputs. The trained is as an input to the recommender in the left bottom corner. We also include the training data, which means if a user has already rated the devices, we do not need to do predictions but follow her preference.



Draft saved at 00:3
Join Data
We join so that we can give the recommender device names rather than ID.
Enter Data Manually
Enter Data Manually
Select Columns in Dataset
Reorder and filter columns. The recommender takes user, item, rating columns.
Web service input
Remove Duplicate Rows
Sanitize data: we only want one rating per user per device.
Recommender User Info Bas...
Enter Data Manually
Score Matchbox Recommen...
Predict Ratings (For model evaluation)
1:1
Web service output
Web service output.
Web service output

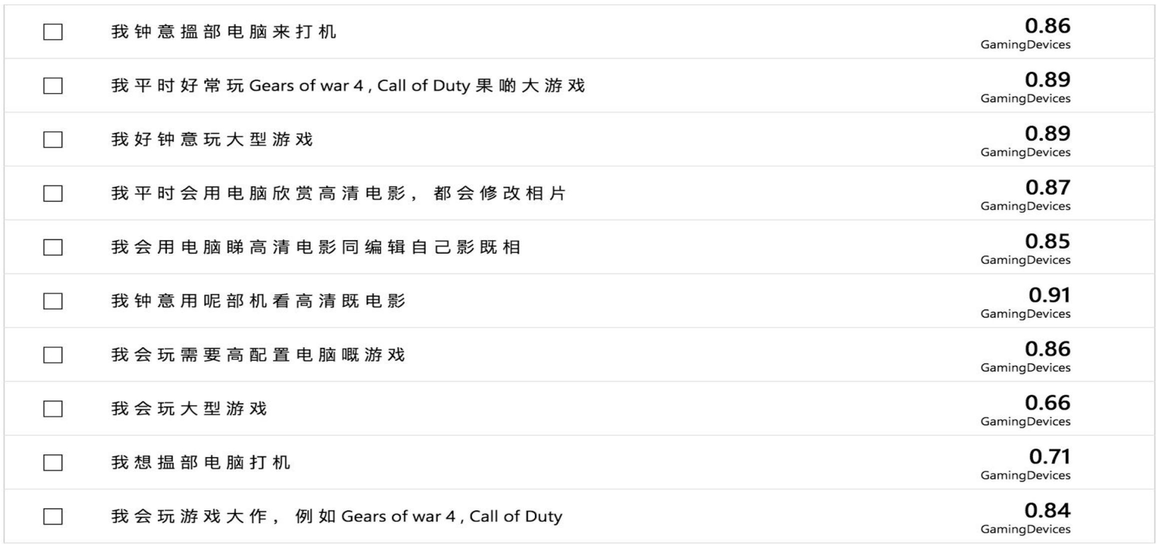
Figure 13: The deployed recommender engine.

9 Evaluation.

This section shows the evaluation of our project, mainly focus on three aspects. The performance of the LUIS model, the performance of the profile based recommendation and a user study to twenty users.

9.1 The performance of LUIS Model.

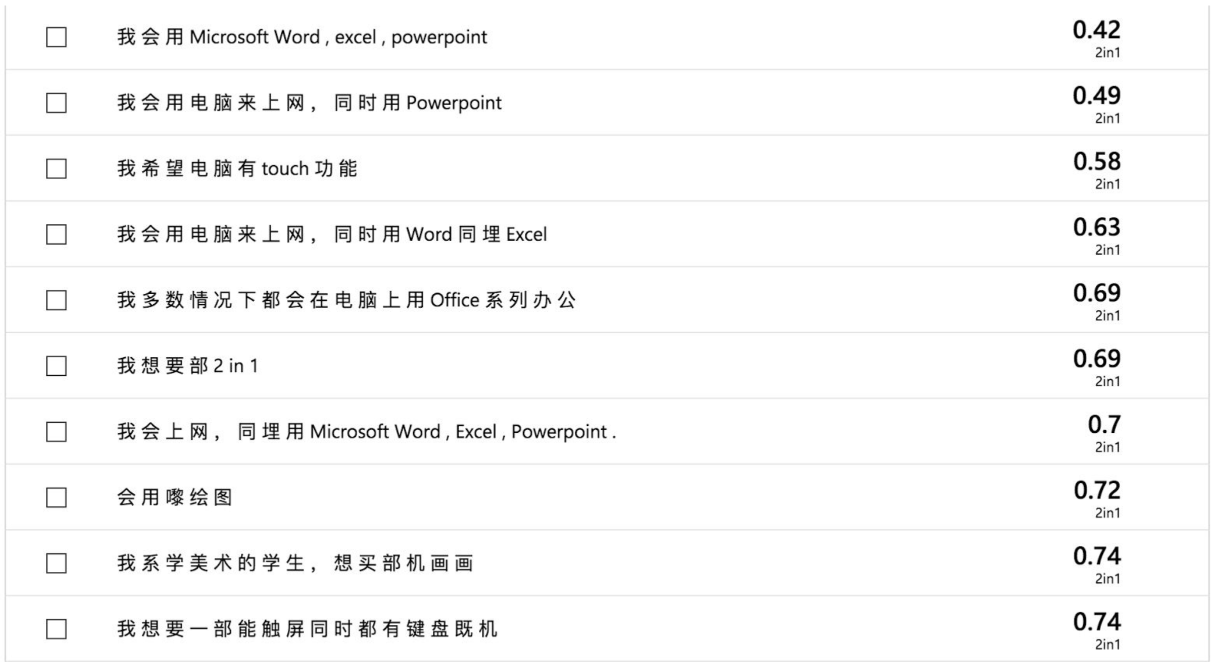
Previously, there was a page on LUIS to show the accuracy, precision and recall figures, and other statistics. However, after their recent upgrading, this feature has been removed. As a result, we can only show the screenshots of the training page to see the prediction on training dataset. The figure below is the prediction on training dataset for gaming devices. It can be extracted almost correctly because the genral feature of gaming device means “high performance”



☐	我钟意揾部电脑来打机	**0.86** GamingDevices
☐	我平时好常玩 Gears of war 4 , Call of Duty 果啲大游戏	**0.89** GamingDevices
☐	我好钟意玩大型游戏	**0.89** GamingDevices
☐	我平时会用电脑欣赏高清电影， 都会修改相片	**0.87** GamingDevices
☐	我会用电脑睇高清电影同编辑自己影既相	**0.85** GamingDevices
☐	我钟意用呢部机看高清既电影	**0.91** GamingDevices
☐	我会玩需要高配置电脑嘅游戏	**0.86** GamingDevices
☐	我会玩大型游戏	**0.66** GamingDevices
☐	我想揾部电脑打机	**0.71** GamingDevices
☐	我会玩游戏大作， 例如 Gears of war 4 , Call of Duty	**0.84** GamingDevices

Figure 14: LUIS performance on Gaming Device

However, the performance for 2-in-1 is not optimal. It is mainly because that the positioning or features of 2-in-1 is not that clear. The performance can be much better if more data are provided.



☐	我会用 Microsoft Word , excel , powerpoint	0.42 2in1
☐	我会用电脑来上网， 同时用 Powerpoint	0.49 2in1
☐	我希望电脑有 touch 功能	0.58 2in1
☐	我会用电脑来上网， 同时用 Word 同埋 Excel	0.63 2in1
☐	我多数情况下都会在电脑上用 Office 系列办公	0.69 2in1
☐	我想要部 2 in 1	0.69 2in1
☐	我会上网， 同埋用 Microsoft Word , Excel , Powerpoint .	0.7 2in1
☐	会用嚟绘图	0.72 2in1
☐	我系学美术的学生， 想买部机画画	0.74 2in1
☐	我想要一部能触屏同时都有键盘既机	0.74 2in1

Figure 15: LUIS performance on 2-in-1

As for the prediction on the sentence it never met before, we leave it for the user study part.

9.2 Machine Learning Studio Model Evaluation.

Because the data are made up by ourselves, with limited quantity and quality, we have already foreseen a bad performance on the model. However, the result matched our assumption while generating the data. For example, if I give a young boy, it will recommend gaming devices. I also assumed that programmers want a high performance desktop, the recommender also gives the right recommendation. Because we have no real data, we can not test whether the model can be generalized to unseen data.

The table below shows the result of the machine learning model, it can be seen that the model fit the training data well, no matter on predicting the ratings given by a user to an item, or on predicting the rank of preference of a user. That is because the data is biased. Although we tried to add some disturb in the data, we were still following some rules while generating them. That means actually the model must be overfitted and can hardly extend to real world. However, this can be solved if the bot is maintained by Microsoft and they can use the real data to train the model.

Mean Absolute Error (test size = 60)	1.83
Root Mean Square Error (test size = 60)	2.32
Normalized Discounted Cumulative Gain (test size = 15)	0.95

Table 1: The performance of Machine Learning Model

9.3 User Study.

We found 20 students from HKU as users of our bot, and let them give rates to the bot, on the conversation logic, recommendation satisfactory rate, and overall rating, from 1 to 9. We also showed them the five categories of devices and let them determine whether they thought they have been recommended with a the right types, as an evaluation of the performance of LUIS. The table below shows the average of the ratings.

Rating on interaction logic	8.75
Rating on recommendation result	5.45
Overall Rating	6.6
Accuracy of category.	0.6

Table 2: The User study result

From the result, the interaction logic is well accepted by the users because of the concision of the words. However, the accuracy is not satisfactory.

This can be largely improved if more data can collectected in real world as training data. The current training dataset, provided by the Microsoft is too small for a machine learning program. Moreover, the data are actually created by the business team, rather than from real world, which makes the data are also biased as that for machine learning module. Furthermore, the lacking in data size makes it hard to find relation between each sample. The LUIS can work perfectly if the input is similar to a previous training data but will be panic if the data is not similar to anyone and give some random answer.

10 Discussion.

If the bot is continually developed and maintained by the Microsoft, I want to give the following suggestions which I believe can make the bot much better.

1. **Provide more real world training data.** The training data is too sparse for our project, which greatly limited the performance of the bot.
2. **Refine the conversation logic.** Although the current conversation logic achieves a good satisfactory rate during the user study, that only means it’s not annoying. It also failed to help the customer to find the best device. The development team should communicate team to find a middle place between concision and information gain during the conversion.
3. **Combine the Azure Machine Model and LUIS**. Currently, our recommender engine and the LUIS model works separately. Although it may be true that a young boy want an Xbox, he may also has already got one and want to buy something else. It will be better if the recommendation not only rely on the user profile, but use more diverse data and the conversation for help. However, that will be determined by what user data can the Microsoft company get.

11 Division of Labour.

Chen Xusheng Michael, during the first half of the project, is responsible for design the LUIS model and train an english model (the archived one). For the second half of the project, he is responsible for getting through the connection with translator API and build the Recommender on Machine Learning Studio.

Tan Zhanwen Francis, during the first half of the project, is responsible for coding the logic of the bot. He also took over the Cantonese Language Model in the second half.

Huang Kai Kevin, during the first half of the project, is responsible for setting up database, resolve the connection issue and the complaint handling. For the second half, he helped with Francis on the Cantonese model and bot logic, together with the connection with the recommender deployed by Michael.

12 Conclusion.

Machine Learning and AI bots are the most heated topic in the current Computer Science world. However, in order to have good performance, other than a good algorithm, the training data is also critical. In this project, we built a Machine Learning based bot prototype as a standalone web service to give recommendation of electronic device for customers. We hope this bot will be continually developed by Microsoft and be online someday.

Reference

[1] Quora [Internet]. USA: Why do PC laptop manufacturers have such a confusion of differently specified models? [updated 2012 Mar 17, cited 2016 Oct 22]. Available from: https://www.quora.com/Why-do-PC-laptop-manufacturers-have-such-a-confusion-of-differently-specified-models.

[2] Kissmetrics [Internet]. USA: Are you Losing Sales by Giving Customer Too Many Choices? [updated 2012, cited 2016 Oct 22]. Available from: https://blog.kissmetrics.com/too-many-choices.

[3] Lexalytics [Internet]. USA: NLP in 5 Munutes. [updated 2016 April 27, cited 2016 Oct 21]. Available from: https://www.lexalytics.com/lexablog/2016/nlp.

[4] Apple [Internet]. USA: Siri. [cited 2016 Oct 21]. Available from: http://www.apple.com/ios/siri.

[5] Microsoft [Internet]. USA: What is Cortana. [updated 2016 Sep 10, cited 2016 Oct 21]. Available from: https://support.microsoft.com/en-hk/help/17214/windows-10-what-is.

[6] Microsoft [Internet]. USA: Language Understanding Intelligent Service. [cited 2016 Oct 20]. Available from: https://www.microsoft.com/cognitive-services/en-us/language-understanding-intelligent-service-luis.

[7] Simon P. Too Big to Ignore: The business Case for Big Data. USA: Wiley; 2013 Mar 18.

[8] Gitbooks [Internet]. Artificial Intelligence: Machine Learning. [cited 2016 Oct 20]. Available from:

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/machine_learning.html

[9] Fisher RA. The use of multiple measurements in taxonomic problems. Annals of Eugenics. 1936; 7(2): 179-188.

[10] Microsoft [Internet]. USA: Microsoft Azure. [cited 2016 Oct 20]. Available from: https://www.microsoft.com/cognitive-services/en-us/language-understanding-intelligent-service-luis.

[11] Microsoft [Internet]. USA: Microsoft Bot Framework. [cited 2016 Oct 20]. Available from: https://dev.botframework.com/.

[12] Microsoft [Internet]. USA: Creating and retrieving resources in Windows Runtime apps. [cited 2016 Oct 20]. Available from: https://msdn.microsoft.com/en-us/library/hh694557.aspx