Project Report CS476

Members

1. Fouad Majd Alkadri |218110075
2.Abdullah Rajoub | 218110141
3. Abdulaziz Alowain |219110119
4. Ibrahim Khurfan| 218110082

Chatbot

What is a Chatbot?

At the most basic level, a chatbot is a computer program that simulates and processes human conversation (either written or spoken), allowing humans to interact with digital devices as if they were communicating with a real person. We will focus on the written aspect only.

Student Names:

Student-Name	Role	Student-ID
Fouad Alkadri	Assistant	218110075
Ibrahim Khurfan	Team Leader	218110082
Abdullah Rajoub	Assistant	218110141
Abdulaziz Alowain	Assistant	219110119

Topic Choice:

The topic that we've chosen is coffee shop Chatbot.

Goal:

The coffee chatbot has 3 main objectives as inputs:

1. Take orders
2. Take questions
3. Take Complaints

As and output for each input:

1. Will initiate an order and send it to the coffee branch.
2. Will answer questions based on a predefined set of questions and if the answer is not there it will reconnect the customer with the call center.
3. Will record complaints and send it for the quality assurance team to solve it.

Distribution Table:

Names:	Tasks:
Fouad Alkadri	• Topic Choice. • Evaluate the Measures and Error Analysis Methods. • Relevant Models. • Elementary experiments. • Applied Models and Error Analysis. • Techniques used for Reducing the error: • Code resources.
Ibrahim Khurfan	• Topic Choice • The Dataset available. • Dataset Specification • Exploratory data analysis (EDA). • Elementary experiments:
Abdullah Rajoub	• Topic Choice. • The Problem Proposed Approach and Tools to be Used. • Scope of the Project. • Dataset Specification. • Exploratory data analysis (EDA). • Elementary experiments:
Hammad Ismaeel	• Problem and challenges in Chatbot.
Abdulaziz Alowain	• Topic Choice. • Interface Used Work Plan. • Related Work. • Group Management. • Relevant Models. • Elementary experiments.

Table of Contents

1. The Problem

Problems we are trying to solve are that customers waste a lot of time waiting in line at a coffee shop to make an order and need to waste money calling customer service to ask questions or to give feedback, so in order to make it easier and less expensive for the customers to do all of these things, and make them do it with a click of a button we are going to make a virtual agent (chatbot) for the coffee shop where the chatbot can do the following tasks:

❖ 1.1 Taking orders from customer

- The customer will initiate a conversation with the chatbot
- Chatbot detects the order intent.
    - If the customer has provided what they want to order already chatbot will add it to the cart and ask the customer if they want anything else or if they order or not.

❖ 1.2 Take questions from customers

- The customer will initiate a conversation with the chatbot
- Chatbot detects the question's intent
    - the chatbot will check the question if it is already stored in the database with its answer, and provide the answer associated with it.

❖ 1.3 Take Complaints from customers

- The customer will initiate a conversation with the chatbot
- Chatbot detects the complaint intent:
    - the chatbot will check the complaints if it is already stored in the database with its steps to deal with complaints
    - The chatbot will store the customer complaint and send it to the customer service team to review it and solve the customer problem.

    - The chatbot will understand any items that are not listed in the coffee shop menu.

❖ 1.4 Providing suggestions to the user:

- The chatbot will be able to provide a suggestion of good food/drinks combos when prompted

❖ 1.5 What is excluded

- ❖ The chatbot will not provide audio or video calls.
- ❖ The chatbot will not support any language other than **English.**

2. The Dataset available

For our chatbot, we are going to use external existing coffee shop datasets to shorten the development time and increase the accuracy of our chatbot, most of our datasets will be taken from Kaggle website, and we are going to combine these datasets and modify some of them to fit the coffee shop menu items. Finally, the datasets will be in **English**.

3. Proposed Approach and Tools to be Used

The tool that will be used is Rasa. Rasa is an open-source machine learning framework for building AI assistants and chatbots. Mostly you don't need any programming language experience to work in Rasa. Rasa can be used to build contextual AI chatbots, meaning that the chatbot will be capable of providing responses that fit the context. For example, if I'm ordering a coffee, and the chatbot wants to confirm my order, the chatbot should respond with "do you want to confirm your coffee purchase? "

Rasa consists of the following components:

NLU: it is the part of the chatbot used for entity identification and intent classification. It enables your chatbot to understand what is being said. It takes the input in unstructured human language form and extracts entries and intents.

Core: It is also referred to as a dialog management component. It is the part of the chatbot that is concerned with decision-making. How should I respond to a specific input?

We will take a deeper look at the approach used for each of these two components:

3.1 Creating The NLU

In this step, we must do conversation design. Conversation design includes:

- ❖ Identifying your target users.
- ❖ Understanding what they will use your assistants for.
- ❖ Crafting the most typical conversation they will have with your assistant.

Our target users are coffee shop customers. They will be using our assistant to do the following:

- Place a new order.
- Make complaints.
- Ask for available menu items
- Ask for available offers

To craft a typical conversation, we used our experience with in-person customer service and how a typical conversation between us (customers) and a coffee shop employee goes on. Some things to consider while crafting the typical conversations are:

- We will not include all variations of responses in NLU training data.
- Only a sample example is 5-15 per intent, for harder cases like complaints we are going to use 20+ examples to make sure that we get the right intent.
- The example given should have a single intent (e.g., want can have purchased as intent and complain as intent).
- The examples given under intent should be diverse in vocabulary and grammatical structure.

3.2 Processing pipelines:

A processing pipeline is a sequence of processing steps that extracts text features that allow the module to learn the underline pattern from the provided example. In the beginning, we are going to use one of the pre-configured pipelines. This pre-configured pipeline is called **pretrained_embedding_spacy**. This library represents each word as a vector of values. These vectors of words are used to compare how closely two words are similar to each other in meaning (semantic) and grammar (syntactic).

Figure 1

As shown in Figure 1, the word cheeseburger and hamburger are closer to each other (their vectors are closer). The word Ferrari is far away from both.

Some advantages of using **pretrained_embedding_spacy**:

- Faster training and iteration
- Less training data is required to achieve good model performance

It is worth noting this model can handle intent classification and entity extraction too.

NOTE: The approach provided is not comprehensive and will probably change throughout the project. This is just the initial approach that we have in mind.

4. Scope of the Project

4.1 Features that are included in the project

- The chatbot will be able to provide a list of items on command.
- The chatbot will be able to remove the order from the customers.
- The chatbot will be able to take orders from customers.
- The chatbot will be able to give checkout of the order.
    - Order: I want to order coffee: Response: what type of coffee do you want?
        - Americano: Ok, nice choice, but what size?
        - Large: Ok, noted sir. Anything else?
- The chatbot will be able to take complaints.
- The chatbot will be able to respond to greetings.
- The chatbot will provide helpful contact information on command (e.g., support team email).
- The chatbot will provide suggestions to the customer.

4.2 Features not included in the project

- The chatbot will not be able to respond to any non-English and non-Arabic input.
- No speech recognition feature for voice input.

4.3 Interface Used

We will use a **Chatbot that'll be integrated with the Front-end of the webpage (HTML)** which provides all the needed services for Rasa customer services with chatbot capabilities.

5. Work Plan

Week 1-8

- Pick NLP application (Chatbot)
- Pick NLP topic (Coffee Shop Chatbot)

Week 9

- Create Proposal
- Create a simple Rasa Chatbot
- Search for datasets

Week 11

- Create stories and appropriate actions for our chatbot

Week 10

- Implement the datasets to our chatbot

Week 12

- Create stories and appropriate actions for our chatbot

Week 13

- Create stories and appropriate actions for our chatbot

Week 14

- Refine the Machine Learning Components

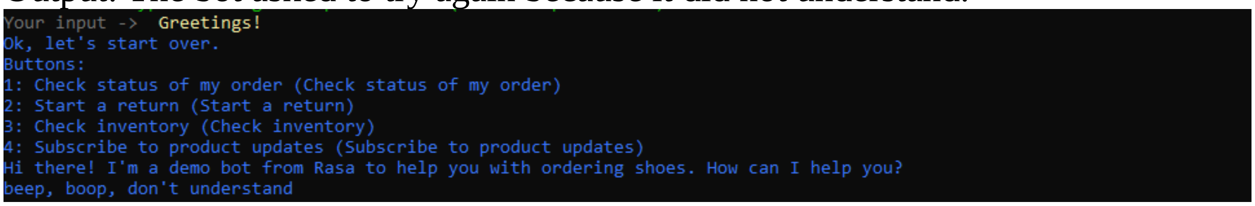
Week 15

- Test the chatbot and add final touches

6. Related Work

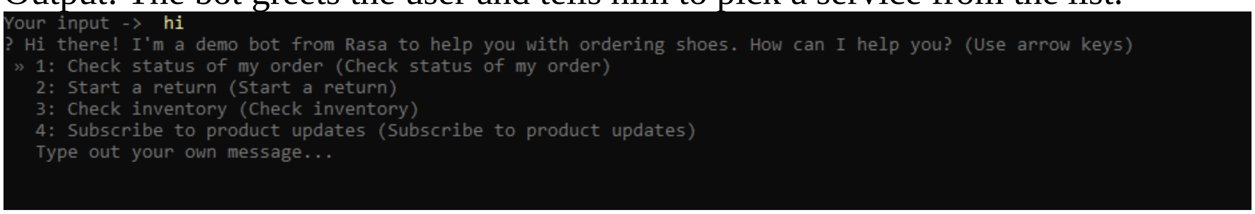
In this part of our proposal, we tried Rasa's starter pack Retail chatbot. The inputs we tried in this chatbot are:

1. Hi using unconventional ways.
   Input: "Greetings!"
   Output: The bot asked to try again because it did not understand.



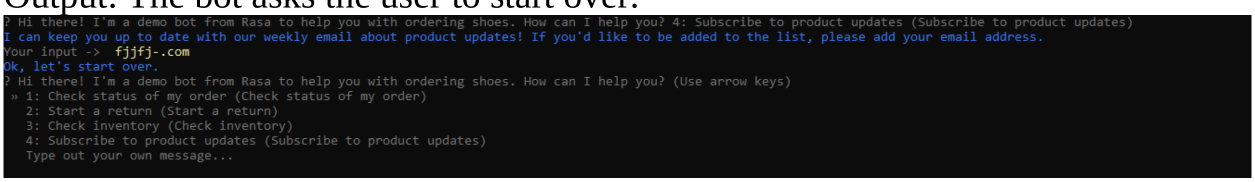
```
Your input ->  Greetings!
Ok, let's start over.
Buttons:
1: Check status of my order (Check status of my order)
2: Start a return (Start a return)
3: Check inventory (Check inventory)
4: Subscribe to product updates (Subscribe to product updates)
Hi there! I'm a demo bot from Rasa to help you with ordering shoes. How can I help you?
beep, boop, don't understand
```

2. A normal Hi.
   Input: "Hi"
   Output: The bot greets the user and tells him to pick a service from the list.



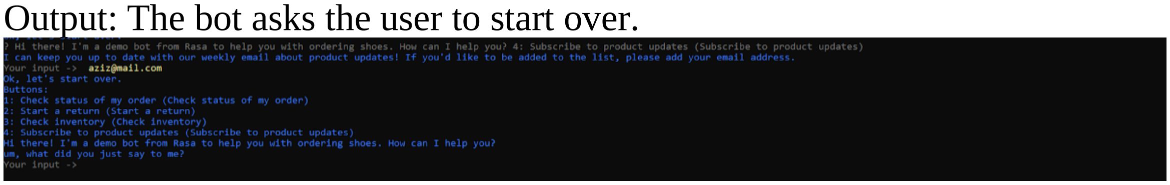
```
Your input ->  hi
? Hi there! I'm a demo bot from Rasa to help you with ordering shoes. How can I help you? (Use arrow keys)
 » 1: Check status of my order (Check status of my order)
   2: Start a return (Start a return)
   3: Check inventory (Check inventory)
   4: Subscribe to product updates (Subscribe to product updates)
   Type out your own message...
```

3. We picked the 4th service "Subscribe to product updates" and entered an invalid email (no @).
   Input: "fjjfj-.com"
   Output: The bot asks the user to start over.



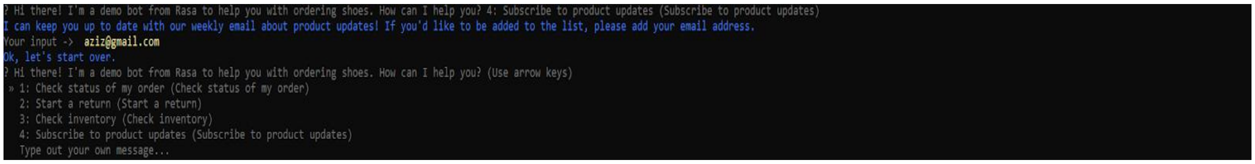
```
? Hi there! I'm a demo bot from Rasa to help you with ordering shoes. How can I help you? 4: Subscribe to product updates (Subscribe to product updates)
I can keep you up to date with our weekly email about product updates! If you'd like to be added to the list, please add your email address.
Your input ->  fjjfj-.com
Ok, let's start over.
? Hi there! I'm a demo bot from Rasa to help you with ordering shoes. How can I help you? (Use arrow keys)
 » 1: Check status of my order (Check status of my order)
   2: Start a return (Start a return)
   3: Check inventory (Check inventory)
   4: Subscribe to product updates (Subscribe to product updates)
   Type out your own message...
```

4. We picked the 4th service "Subscribe to product updates" and entered a valid email but an unknown domain (@mail).
   Input: "aziz@mail.com"
   Output: The bot asks the user to start over.

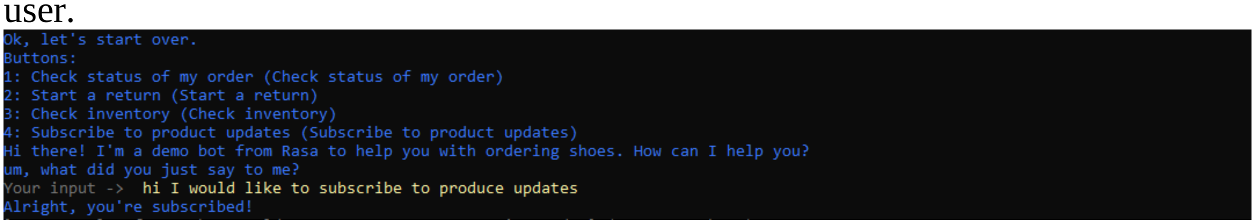


```
? Hi there! I'm a demo bot from Rasa to help you with ordering shoes. How can I help you? 4: Subscribe to product updates (Subscribe to product updates)
I can keep you up to date with our weekly email about product updates! If you'd like to be added to the list, please add your email address.
Your input ->  aziz@mail.com
Ok, let's start over.
Buttons:
1: Check status of my order (Check status of my order)
2: Start a return (Start a return)
3: Check inventory (Check inventory)
4: Subscribe to product updates (Subscribe to product updates)
Hi there! I'm a demo bot from Rasa to help you with ordering shoes. How can I help you?
um, what did you just say to me?
Your input ->
```

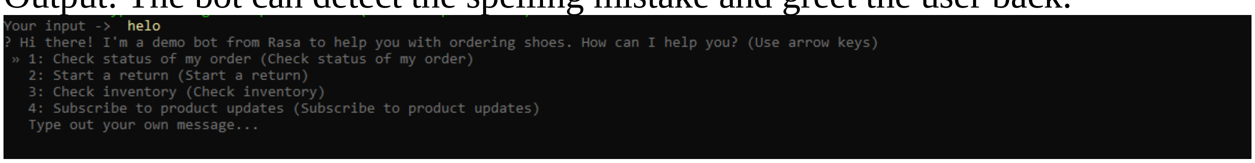
5. We picked the 4th service "Subscribe to product updates" and entered a valid email domain.
   Input: "aziz@gmail.com"
   Output: The bot asks the user to start over. This means that there is a bug with the bot.



6. We tried double intents with spelling mistakes (produce not product).
   Input: "hi I would like to subscribe to produce updates"
   Output: The bot subscribes the user to the service and does not greet the user.



7. We tried greetings with spelling mistakes
   Input: "helo"
   Output: The bot can detect the spelling mistake and greet the user back.



6.1 Conclusion

The bot was able to detect spelling mistakes and deal with double intents correctly. There are however some issues with the services of the bot. For example, even if we provide a correct email for the subscription service the bot still asks the user to try again.

7. Evaluate the Measures and Error Analysis Methods

7.1 Experiment

The experiment that we're going to conduct is about the cafe shop chatbot that will include a starter pack, so the steps that we're going through that we implement are the following:

- Build the chatbot project through the command-line interface.
  - Command Line Anaconda.

- Installing **Rasa** inside the virtual environment that we're going to use.
  - Python 3.7

- Adding the pipelines or the components that are going to be trained as a model in **config.yml** which are the following:
  - Machine learning Component.
  - Neural Network Component.
  - RexgetFeaturizer Component.
  - LexicalSyntaticFeaturizer Component.
  - CountVectorFeaturizer Component.
  - WhitespaceTokenizer Component.

- Implement our datasets and put the examples of intents and the actions through **data/nlu.md**.
  - Sample example:

  EX 1:
  ##Intent: greet
  -hello
  -hi
  -yo
  -wassapp
  -hya

EX 2: [--] search for that pokemon, **(--)** entity.
##Intent: Find Pokemon
-is [bulbasaur] (pokemon-name) a pokemon
##Lookup: pokemon-name
-data/pokemone.txt

- Ensure that the **Domain.yml** file covers the domain of **data/nlu.md** file that is responsible for the action of the chatbot if it is either available or not.

- Implement the stories and the sequence of action examples in the **data/stories.md** file that is responsible for the flow of the story that's created by the user when they're experimenting with the chatbot.

- Training **Rasa** through this dataset by using the pipelines or components that we've provided.

- Experimenting with the chatbot that has been built and trained.

7.2 Measures

The measures that we're going to use throughout the project to evaluate that the chatbot works well are for the following reasons:

- User feedback.
    - By simply asking real coffee shop customers, if they're satisfied with the chatbot or not.

7.3 Error Analysis Methods

In this section, we'll provide scenarios and some examples of what the possible problem could occur through this. And the possible errors that might disrupt the analysis methods are the following:

- **Dataset**
    - Similar classes.
    - Not enough features.
    - Noisy dataset (High difference in inputs).
    - Empty data (missing data → Normalization).

- **Note: most of the datasets that we're going to use are preprocessed.** The solution for that problem is we might be facing are the following:

- **Dataset**
  - Normalization of the dataset.

- **Chatbot**
  - Missing to cover some cases that might the user ask about it, for example, the user says **(Hey)** and in the chatbot don't cover this domain and it'll reply to weird stuff.

Sample example:

EX 1:
##Intent: greet
-hello
-hi
-yo
-wassapp
-hya

Compiler:
User: Hey .
Chatbot: Bye.

- This example shows how the chatbot got confused and didn't know what he could do if there was no possible output for responses.

Solution:
EX 1:
##Intent: greet
-hello
-hi
-yo
-wassapp
-hya
-Hey

Compiler:
User: Hey.
Chatbot: Hi.

❖ See the confidence of the model if it works well or not.


```json
{
  'intent':{
    'name':'greet',
    'confidence':0.44503513568867775
  },
  'entities':[
    {
      'value':'neg',
      'confidence':0.9933702940854111,
      'entity':'sentiment',
      'extractor':'sentiment_extractor'
    }
  ],
  'intent_ranking':[
    {
      'name':'greet',
      'confidence':0.44503513568867775
    },
    {
      'name':'chitchat',
      'confidence':0.20129539551108508
    },
    {
      'name':'inform',
      'confidence':0.09576408290307896
    },
    {
      'name':'goodbye',
      'confidence':0.08987117551991658
    },
    {
      'name':'decline',
      'confidence':0.08840002616908385
    },
    {
      'name':'affirm',
      'confidence':0.04842063587016189
    },
    {
      'name':'restaurant',
      'confidence':0.03121354833799584
    }
  ],
  'text':'Hello stupid bot'
}
```


❖ By using the command **(-rasa shell nlu).**

○ That shows how the chatbot is chosen or predicts the output with the confidence measure of this model.

8. Group Management

We mainly use discord and WhatsApp for our communications and virtual meetings. Discord is an application like google meet. We do not have specific roles or responsibilities for each member. We instead work on discord where one member would present his screen and code and we would give him ideas. When we feel the work should be distributed as tasks, we try to give each member the tasks he wants. We use google drive to store our documents and code. The frequency of our meetings depends on the amount of work to do. For example, when we proposed, we met twice and distributed the work and then met again to finalize it.

9. Problem and challenges in Chatbot:

Chatbots are not easy to implement, design, or maintain, because processing the natural language changes quite often as is evident by the introduction of new colloquial terms and slang that age rather quickly. We'll attempt to capture all these challenges by producing a chatbot that takes Coffee orders, aka a chatbot Barista. An example of user input could be:

- Coffee large black asap. (Notice how this makes the task very challenging).
- Can I please get a large black coffee? (isn't this input an NLP dream, common human language is rarely such clean and easy).
- The usual (don't even get us started).

But the true challenge of NLP is not just understanding the language but also adapting to all the recent changes that are introduced daily. Additionally, some common problems with chatbots these days are the need to process every increasing amount of data, with similar demands in increases in runtime pose a significant problem as the computation load is increasing daily. Also, as people become more comfortable with using chatbots, the amount of input data would increase and the style of communication of such input would become more informal thus complicating the chatbot's functions.

Solution:

The solution by using Rasa which is an open-source machine learning framework for building AI assistants and chatbots.

Rasa consists of the following components:

NLU: it is the part of the chatbot used for entity identification and intent classification. It enables your chatbot to understand what is being said. It takes the input in unstructured human language form and extracts entries and intents.
Core: It is also referred to as a dialog management component. It is the part of the chatbot that is concerned with decision-making. How should I respawn to a specific input?

10. Dataset Specification

In this section, we are going to use 3 datasets gathered from Kaggle. And we've created our own dataset that will support our chatbot implementation. The Datasets are as follows:

1. **The First Dataset consists of 3 files as follows:**
   a. **Item_to_id file** contains all menu items of a coffee shop with their corresponding ID.

   b. **Conversation file:** contains all conversations that had happened between the café staff and the customer's **Questions/Answers.**

   c. **Food file:** contains the food id, how many it's used, and food rating to train chatbot as a recommender that will advise the user as a recommender for which most ordered food in cafe shop.

2. **The second Dataset consists of 1 file as follows:**
   a. **Café_data:** contains invoices for items menu that has ordered and that will help us to know what is the most ordered food item in these invoices.

3. **The third Dataset consists of many files, but our focus is on the following file:**
   a) **Ratting_and_sentiments:** This file contains Yelp ratings for several coffee shops.

4. **The fourth dataset consists of intents, stories, actions, lookup table, and entities.**

   Note: We'll study our datasets and visualize them. And based on that we'll use some of the data to feed our chatbot with the most useful information regarding concerns of cultural perceptive of our chatbot. Additional to that we've added some examples that's created from us.

10.1 First Dataset:

The first dataset "https://www.kaggle.com/datasets/sonalibhoir/cafe-chatbot-dataset/code?datasetId=693402&select=Item_to_id.csv." will consist of 3 csv files:

10.1.1 items_to_id file:

a. The file consists of two main columns:

i. ID: Data type (int).
ii. Name: Data type (String).

b. File benefits:

i. It will make us keep a record of what items are being ordered the most and what items mostly appear together.

c. File limitation:

i. It does not include all local café menu items.

10.1.2 Conversation file:

a. The file consists of two main columns:

i. Question: Data type (String).
ii. Answer: Data type (String).

b. File benefits:

i. Allows us to gain valuable information on how the conversation is formed in the café and the order of them.
ii. Allows us to gain information on what are the most frequently asked questions and in what order they are presented.

- **Ex:**
- ❖ Can I have a hot chocolate please, it has the same meaning as One hot chocolate, please?
- ❖ Allows identifying what is the best answer that is associated with a different type of questions.

- **Ex:**
- ❖ Q: Can I have one hot chocolate / A: Yes, of course, one hot chocolate is added, anything else?
- ❖ Q: I wonder if I can get one hot chocolate / A: Yes, of course, one hot chocolate is added, anything else?

c. File Limitations:

i. It is not captured from local Café shops.
ii. It will not be able to make us recognize local beverages such as “Shahii Adani or Shaii Karak”.
iii. Limited in only one language which is English.

10.1.3 food file:

a. The file contains three main columns:

i. id: Data type (int).
ii. times_appeared: Data type (int).
iii. food_rating: Data type (int).

b. File Benefits:

i. It will allow us to identify items for customers based on their popularity to appear together, which will increase the sale of the coffee.
ii. It will tell us what items are most liked and how that gets reflected in customer questions.

c. File Limitations:

i. It does not include the time and date of the order since these are crucial factors in determining the recommended order.

- **Ex:**

In winter people will drink hot drinks, also in morning side dishes differs from mid-day or late night.

10.2 Second Dataset:

The Second dataset "https://www.kaggle.com/datasets/ankitverma2010/cafe-data?select=Cafe_Data.xlsx" consists of 1 csv file:

10.2.1 Café_data file:

a. The file contains 9 main columns:

i. Date: Data type (date).
ii. Bill Number: Data type (string).
iii. Item Desc: Data type (string).
iv. Quantity: Data type (int).
v. Rate: Data type (int).
vi. Tax: Data type (float).
vii. Discount: Data type (int).
viii. Total: Data type (float).
ix. Category: Data type (string).

b. File Benefits:

i. It contains categories so we know if the item is a beverage or a side dish. This might be helpful if a customer doesn't know the exact name of a beverage, the chatbot can give a list of similarly named beverages to the customer.
ii. It contains a rate, so we know which items are the most liked.
iii. It contains bill numbers so we know which items are coming together usually so we can predict their appearance in a context together and give the higher weight, also can be provided as a hint for the recommendation system.

c. File Limitations:

i. Does not include conversation.
ii. Does include local café menu items.

10.3 Third Dataset:

The third dataset "https://www.kaggle.com/datasets/sripaadsrinivasan/yelp-coffee-reviews?select=ratings_and_sentiments.csv" We are looking at only the Ratting_and_sentiments.csv file:

10.3.1 Ratting_and_sentiments file:

a) The file consists of twenty columns, but we are interested in only two:

i. **Review Text: Data type (String).**
ii. **Num_Rating: Data type (String).**

b) File benefits:

i. This file will be used to enable the chatbot to detect complaints intent when presented with input similar to negative reviews on Yelp.

c) File limitation:

i. The file includes the names of numerous coffee shops. So, if a coffee shop has a lot of bad reviews, the module will associate the name of the coffee shop with bad reviews.
ii. With bad reviews, the file doesn't indicate the category of service (food/drinks/customer service) that lead to this review.

10.4 Fourth Dataset:

a) The file contains intents, stories, actions, lookup table, and entities.

11. Exploratory data analysis (EDA):

For our datasets that are chosen, we're going to investigate the datasets deeper by the following:

- Displaying file contents.
- Visualizing the datasets.
- Exploring labels for each file.
- Exploring indices of each file.
- Seeing information about each file.
- Getting a description of each file.
- Seeing Data types in each file.
- Checking for null values.
- Checking the shape of each file.
- Graphs.

And these features that are used and have been applied by **Google Collaboratory that 'll be attached here:**

Food_Item_to_id.ipynb

URL:https://colab.research.google.com/drive/1qM7xNPJe8ypIWEFXVHHXpNjm44hqysw?usp=sharing

Converstaion.ipynb.

URL:https://colab.research.google.com/drive/1wU2TnrvL6qT3uZwWdXqwoJWKd_Qlm-ah?usp=sharing.

Cafe_Data.ipynb.

URL:https://colab.research.google.com/drive/1pisjnoEgY2IHdPRHQQHI4s7Fj_JiNNO9?usp=sharing.

Rating_EDA.ipynb

URL:https://colab.research.google.com/drive/1G9PIBoh69xpq4f3uCfOw4jZKq54Tllj?usp=sharing.

Fourth Dataset

URL: https://github.com/fouad20000/NLP_Project.

11.1 Food_Item_to_id

Displaying file contents.

	id	times_appeared	food_rating
0	1001.0	9.0	2.0
1	1002.0	116.0	2.0
2	1003.0	13.0	2.0
3	1004.0	41.0	2.0
4	1005.0	68.0	1.0
...	...	...	...
680	NaN	NaN	NaN
681	NaN	NaN	NaN
682	NaN	NaN	NaN
683	NaN	NaN	NaN
684	NaN	NaN	2.0

685 rows × 3 columns

```
#Displaying for the food file
item_to_id
```

	id	name
0	1	Chocolate and Vanila Combo
1	2	Avocado Shake
2	3	Apple pomegranate juice
3	4	Drumstick Milkshake
4	5	Pumpkin Shake
...	...	...
60	61	French Coffee
61	62	Iced Coffee Late
62	63	Irish Coffee
63	64	Latte Macchiato
64	65	Wainans Choco Coffee

65 rows × 2 columns

Exploring labels for each file.

```
[ ] #Food Columns
    food.columns
```

```
Index(['id', 'times_appeared', 'food_rating'], dtype='object')
```

```
[ ] #item_to_id Columns

    item_to_id.columns
```

```
Index(['id', 'name'], dtype='object')
```

Exploring indices of each file.

```
[ ] #Food indices

    food.index
```

```
RangeIndex(start=0, stop=685, step=1)
```

```
[ ] #item_to_id indices

    item_to_id.index
```

```
RangeIndex(start=0, stop=65, step=1)
```

Seeing information about each file.

```
[ ] #food info
    food.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 685 entries, 0 to 684
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             65 non-null     float64
 1   times_appeared 64 non-null     float64
 2   food_rating    67 non-null     float64
dtypes: float64(3)
memory usage: 16.2 KB
```

```
[ ] #item_to_id info
    item_to_id.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      65 non-null     int64
 1   name    65 non-null     object
dtypes: int64(1), object(1)
memory usage: 1.1+ KB
```

Getting a description of each file.

```
[ ] #food describe
    food.describe()
```

	id	times_appeared	food_rating
count	65.000000	64.000000	67.000000
mean	2439.430769	340.906250	1.208955
std	1130.990163	760.886507	0.896750
min	1001.000000	1.000000	0.000000
25%	1017.000000	9.000000	0.000000
50%	2015.000000	66.000000	2.000000
75%	3015.000000	275.500000	2.000000
max	4015.000000	4691.000000	2.000000

```
[ ] #item_to_id describe
    item_to_id.describe()
```

	id
count	65.00000
mean	33.00000
std	18.90767
min	1.00000
25%	17.00000
50%	33.00000
75%	49.00000
max	65.00000

Seeing Data types in each file.

```
[ ] #Food data types
    food.dtypes
```

```
id                float64
times_appeared    float64
food_rating       float64
dtype: object
```

```
#item_to_id data types

item_to_id.dtypes
```

```
id       int64
name    object
dtype: object
```

Checking for null values.

```
[ ] #Food null values
    food.isnull()
```

	id	times_appeared	food_rating
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
680	True	True	True
681	True	True	True
682	True	True	True
683	True	True	True
684	True	True	False

685 rows × 3 columns

```
[ ] #Summing all null values for each column
    food.isnull().sum()
```

```
id                620
times_appeared    621
food_rating       618
dtype: int64
```

```
[ ] #item_to_id null values
    item_to_id.isnull()
```

	id	name
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
60	False	False
61	False	False
62	False	False
63	False	False
64	False	False

65 rows × 2 columns

```
[ ] #Summing all null values for each column
    item_to_id.isnull().sum()
```

```
id      0
name    0
dtype: int64
```

Checking the shape of each file.

```
[ ] #Food Shape
    food.shape
```

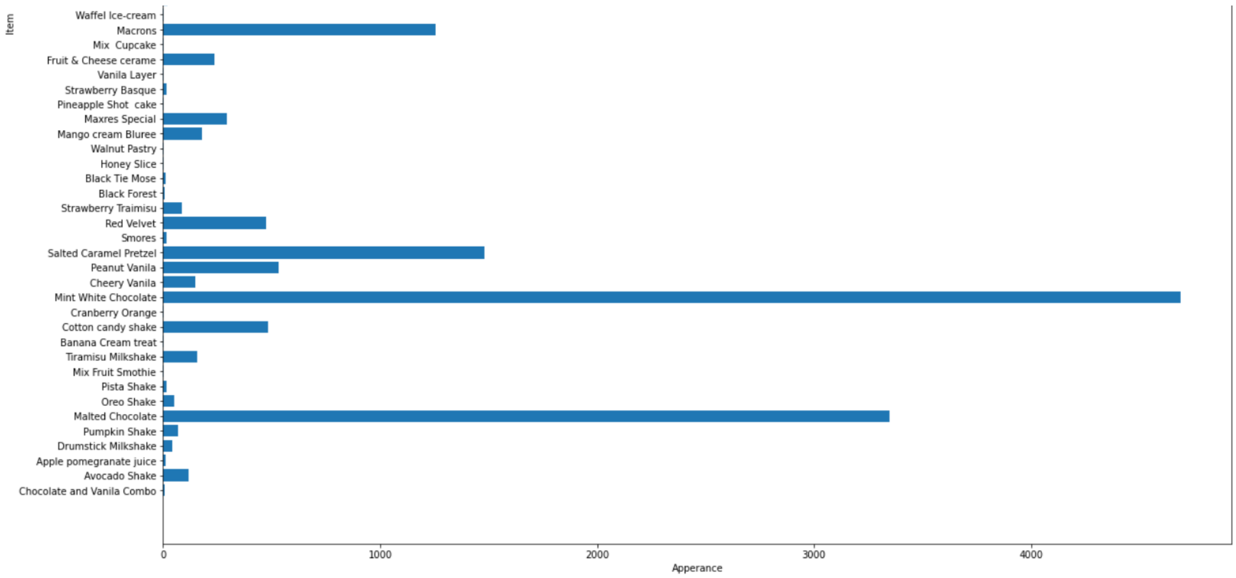
```
(685, 3)
```

```
[ ] #item_to_id Shape
    item_to_id.shape
```

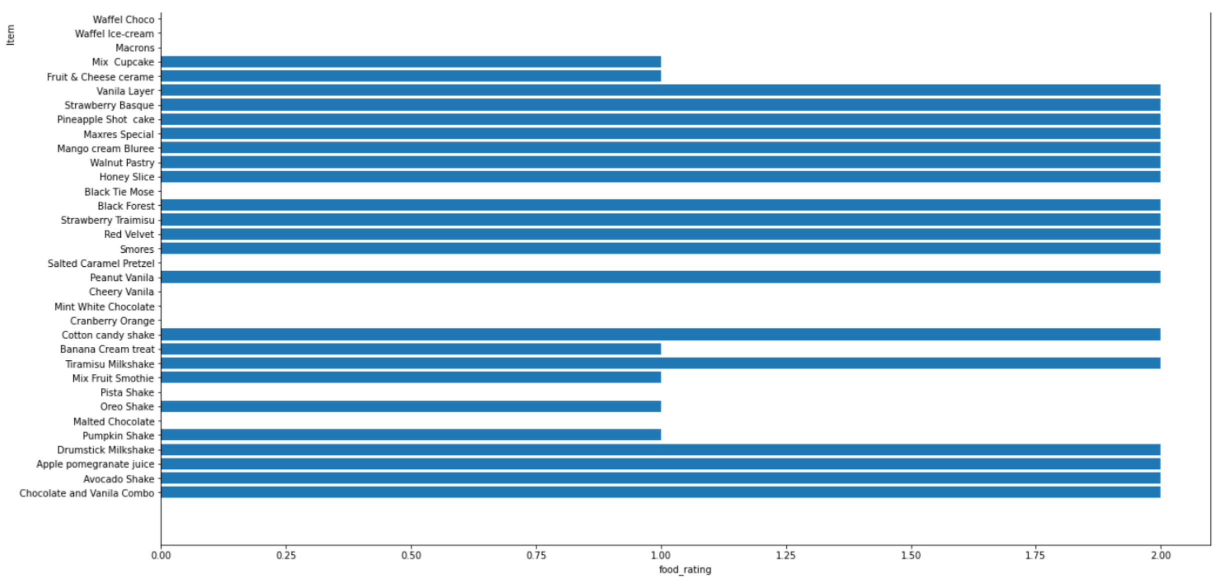
```
(65, 2)
```

Visualizing the datasets.

Seeing which item has been appearing frequently



Seeing each item rating



What we have noticed

- The food file has many NULL values
- The most frequently appearing item is Mint white chocolate
- Data is not normally distributed
- We need to remove NULL values
- If a date was given it would be helpful since we can decide the time and give recommendations based on the time.

11.2 Conversation

Displaying file contents.

	Question	answer
0	hey	Hello! How may I help you.
1	do u have coffee	Yes sir Simple Coffee ,Cappuchino, Americano,...
2	i will take one espresso and 5 americano	Sir thanks for your order. You have ordered 1 ...
3	anything special	We have coffe,pastries,puff pastries and milks...
4	suggest something	We have coffe,pastries,puff pastries and milks...
...	...	...
974	what is price of French Coffee	Its our one of best, you can enjoy it at just ...
975	what is price of Iced Coffee Late	Its our one of best, you can enjoy it at just ...
976	what is price of Latte Macchiato	Its our one of best, you can enjoy it at just ...
977	what is price of Wainans Choco Coffee	Its our one of best, you can enjoy it at just ...
978	book me a table	To book a table you can click on last icon on ...

979 rows × 2 columns

Exploring labels for each file.

```
[ ] #Chat Columns

    chat.columns

    Index(['Question', 'answer'], dtype='object')
```

Exploring indices of each file.

```
[ ] #Chat indices

    chat.index

    RangeIndex(start=0, stop=979, step=1)
```

Seeing information about each file.

```
[ ] #chat info
    chat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 979 entries, 0 to 978
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Question  961 non-null    object
 1   answer    939 non-null    object
dtypes: object(2)
memory usage: 15.4+ KB
```

Getting a description of each file.

```
[ ] #chat describe
    chat.describe()
```

	Question	answer
count	961	939
unique	170	121
top	what is price of Americano	Its our one of best, you can enjoy it at just ...
freq	78	164

Seeing Data types in each file.

```
[ ] #chat data types
    chat.dtypes
```

```
Question    object
answer      object
dtype: object
```

Checking for null values.

```
[ ] #chat null values
    chat.isnull()
```

	Question	answer
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
974	False	False
975	False	False
976	False	False
977	False	False
978	False	False

979 rows × 2 columns

```
[ ] #Summing all null values for each column
    chat.isnull().sum()
```

```
Question    18
answer      40
dtype: int64
```

Checking the shape of each file.

```
[ ] #chat Shape
    chat.shape

(979, 2)
```

What we have noticed

- The file has null values
  - we need to remove null with respect to Question since whenever there is a null Question there will be no answer.
  - for answer null values it could be a result of not recording the answer or it could be a result of a human gesture communication so the customer did not say anything, and the employee approached him, we need to remove that since we are not going to have computer vision in our project so the chatbot will not see customer gesture.

- The most frequently appearing question is the price of americano coffee "what is the price of Americano".
- Data is only for the question and answer for the first time, there is no continuity of the conversation.

11.3 Cafe_Data

Displaying file contents.

```
[ ] #Displaying for the cafe file
    cafe
```

	date	Bill Number	Item Desc	Quantity	Rate	Tax	Discount	Total	Category
0	2010-04-01 13:15:11	G0470115	QUA MINERAL WATER(1000ML)	1	50.0	11.88	0.0	61.88	BEVERAGE
1	2010-04-01 13:15:11	G0470115	MONSOON MALABAR (AULAIT)	1	100.0	23.75	0.0	123.75	BEVERAGE
2	2010-04-01 13:17:35	G0470116	MASALA CHAI CUTTING	1	40.0	9.50	0.0	49.50	BEVERAGE
3	2010-04-01 13:19:55	G0470117	QUA MINERAL WATER(1000ML)	1	50.0	11.88	0.0	61.88	BEVERAGE
4	2010-04-01 01:20:18	G0470283	MOROCCAN MINT TEA	1	45.0	10.69	0.0	55.69	BEVERAGE
...	...	...	...	...	...	...	...	...	...
145825	2010-05-22 21:43:55	N0028716	ZINZI WHITE (GLS)	2	150.0	78.00	0.0	378.00	LIQUOR
145826	2010-04-27 20:52:11	N0028343	ZINZI WHITE (GLS)	2	150.0	78.00	0.0	378.00	LIQUOR
145827	2010-05-28 01:03:37	N0028835	ZINZI WHITE (GLS)	3	150.0	117.00	0.0	567.00	LIQUOR
145828	2010-04-30 23:44:37	N0028399	ZINZI WHITE (GLS)	1	150.0	39.00	0.0	189.00	LIQUOR
145829	2010-07-09 00:31:51	N0029472	ZINZI WHITE (BTL)	1	700.0	182.00	0.0	882.00	LIQUOR

145830 rows × 9 columns

Exploring labels for each file.

```
#cafe Columns
cafe.columns
```

```
Index(['date', 'Bill Number ', 'Item Desc', 'Quantity', 'Rate', 'Tax',
       'Discount', 'Total', 'Category'],
      dtype='object')
```

Exploring indices of each file.

```
[ ] #cafe indices

    cafe.index
```

```
RangeIndex(start=0, stop=145830, step=1)
```

Seeing information about each file.

```
[ ] #cafe info
    cafe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145830 entries, 0 to 145829
Data columns (total 9 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   date          145830 non-null  datetime64[ns]
 1   Bill Number   145830 non-null  object
 2   Item Desc     145830 non-null  object
 3   Quantity      145830 non-null  int64
 4   Rate          145830 non-null  float64
 5   Tax           145830 non-null  float64
 6   Discount      145830 non-null  float64
 7   Total         145830 non-null  float64
 8   Category      145830 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(1), object(3)
memory usage: 10.0+ MB
```

Getting a description of each file.

```
[ ] #cafe describe
    cafe.describe()
```

	Quantity	Rate	Tax	Discount	Total
count	145830.000000	145830.000000	145830.000000	145830.000000	145830.000000
mean	1.121299	161.782259	48.929061	0.095079	224.959852
std	0.477237	102.244631	40.272851	3.720735	164.960776
min	1.000000	0.010000	0.000000	0.000000	0.010000
25%	1.000000	95.000000	22.560000	0.000000	117.560000
50%	1.000000	125.000000	32.060000	0.000000	167.060000
75%	1.000000	225.000000	72.000000	0.000000	315.000000
max	30.000000	2100.000000	2731.250000	825.000000	14231.250000

Seeing Data types in each file.

```
[ ] #cafe data types
    cafe.dtypes
```

```
date           datetime64[ns]
Bill Number            object
Item Desc              object
Quantity                int64
Rate                  float64
Tax                   float64
Discount              float64
Total                 float64
Category               object
dtype: object
```

Checking for null values.

```
#cafe null values
cafe.isnull()
```

	date	Bill Number	Item Desc	Quantity	Rate	Tax	Discount	Total	Category
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
145825	False	False	False	False	False	False	False	False	False
145826	False	False	False	False	False	False	False	False	False
145827	False	False	False	False	False	False	False	False	False
145828	False	False	False	False	False	False	False	False	False
145829	False	False	False	False	False	False	False	False	False

145830 rows × 9 columns

```
[ ] #Summing all null values for each column
    cafe.isnull().sum()
```

```
date           0
Bill Number    0
Item Desc      0
Quantity       0
Rate           0
Tax            0
Discount       0
Total          0
Category       0
dtype: int64
```

Checking the shape of each file.

```
[ ] #cafe Shape
    cafe.shape
```

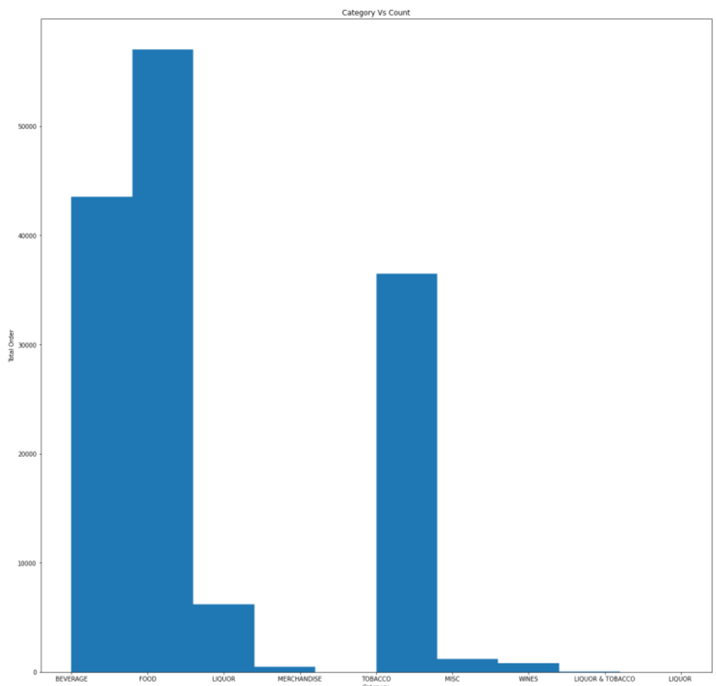
```
(145830, 9)
```

Visualizing the datasets.

Seeing which Category has been appearing frequently

Approximately

Top 1: Food 59k
Top 2: Beverage 43k
Top 3: Tobacco 38k



Seeing Unique values of categories

```
df['Category'].unique()
```

```
array(['BEVERAGE', 'FOOD', 'LIQUOR', 'MERCHANDISE', 'TOBACCO', 'MISC',
       'WINES', 'LIQUOR & TOBACCO', 'LIQUOR '], dtype=object)
```

Removing the Unwanted categories (BEVERAGE)

Grouping items based on the Bill Number

One Bill number example

	date	Bill Number	Item Desc	Quantity	Rate	Tax	Discount	Total	Category
20379	2010-12-25 14:47:32	G0516596	ASSORTED TAPAS(2)VEG	2	275.0	130.63	0.0	680.63	FOOD
39318	2010-12-25 14:47:32	G0516596	CHOCOLATE FONDUE	3	325.0	231.56	0.0	1206.56	FOOD
102401	2010-12-25 14:47:32	G0516596	POLLO CON AIOLI	4	180.0	171.00	0.0	891.00	FOOD
113571	2010-12-25 14:47:32	G0516596	RED BULL 2+1	1	250.0	59.38	0.0	309.38	BEVERAGE
118153	2010-12-25 14:47:32	G0516596	CAPPUCCINO	1	60.0	14.25	0.0	74.25	BEVERAGE
118154	2010-12-25 14:47:32	G0516596	MASALA CHAI CUTTING	3	40.0	28.50	0.0	148.50	BEVERAGE
118155	2010-12-25 14:47:32	G0516596	STRAWBERRY ICED TEA	2	85.0	40.38	0.0	210.38	BEVERAGE
118156	2010-12-25 14:47:32	G0516596	LEMON ICED TEA	26	85.0	524.88	0.0	2734.88	BEVERAGE
118295	2010-12-25 14:47:32	G0516596	POUTINE WITH FRIES	1	125.0	29.69	0.0	154.69	FOOD
118296	2010-12-25 14:47:32	G0516596	STRAWBERRY CHEESECAKE SHAKE	1	135.0	32.06	0.0	167.06	FOOD
118297	2010-12-25 14:47:32	G0516596	GREAT LAKES SHAKE	3	110.0	78.38	0.0	408.38	FOOD

Seeing the most ordered item in each bill

Bill Number	Item Desc	
G0516596	LEMON ICED TEA	26
G0527570	LEMON ICED TEA	12
G0481424	ORANGE ARRABIATA	10
G0508496	GREAT LAKES SHAKE	9
G0485280	CAPPUCCINO	9
		..
G0492623	MAGGI NDLCREAM/ CHEE/GARLIC	1
	MAGGI NDL ARRABIATA	1
	LINDT HOT CHOCOLATE	1
	CHAI LATTE	1
G0502397	STRAWBERRY CHEESECAKE SHAKE	1

Seeing the most item sold

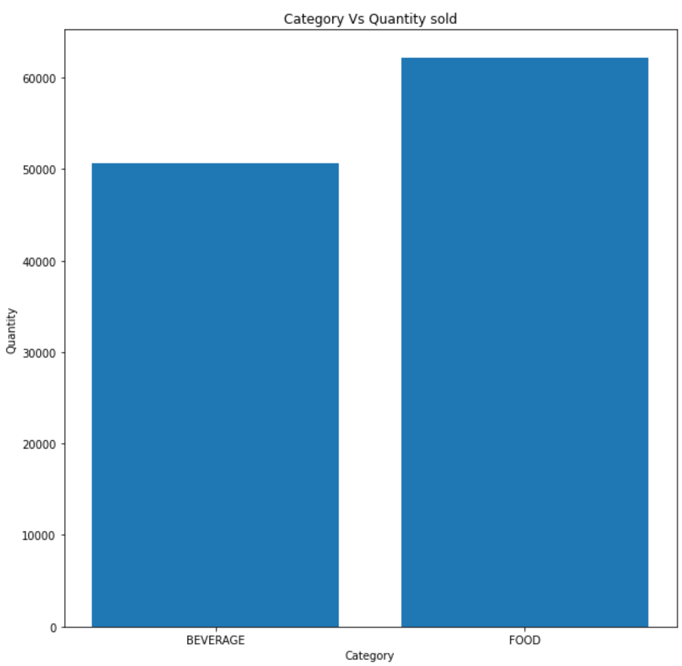
Item Desc	
CAPPUCCINO	7144
GREAT LAKES SHAKE	5914
POUTINE WITH FRIES	3741
QUA MINERAL WATER(1000ML)	3633
JR.CHL AVALANCHE	3446
	...
PEACH BULL	1
CAPONATA	1
DECAFFINATE COFFEE FRAPPE	1
MOCAFE HOT CHOCOLATE(SF)	1
2 AXE TWIST	1

Name: Quantity, Length: 327, dtype: int64

Seeing the item in which has more discount

```
Item Desc
CHEESE FONDUE                              396.0
LEMON INFUSED CHAR GRILLED VEG             270.0
ASSORTED TAPAS(2 )VEG                      247.5
ORANGE ARRABIATA                           225.0
PHILLYCREAM CHEESE &CHILLY PAN             220.5
                                            ...
GOOEY CHOCOLATE FUDGE                        0.0
GREAT LAKES CREAM                            0.0
GREAT LAKES FLOATS W CHOC                    0.0
GREAT LAKES FRAPPE                           0.0
YIN N YANG FONDUE                            0.0
Name: Discount, Length: 327, dtype: float64
```

Seeing Which Category Selling more



Counting the number of unique items in each bill Making them a series in Item Description

Bill Number	Item Desc
G0470109	1
G0470111	4
G0470112	1
G0470114	3
G0470115	3
...	...
G0533898	5
N0027941	1
N0028811	1
N0028973	1
N0034022	1

47365 rows × 1 columns

Seeing each bill and total item ordered

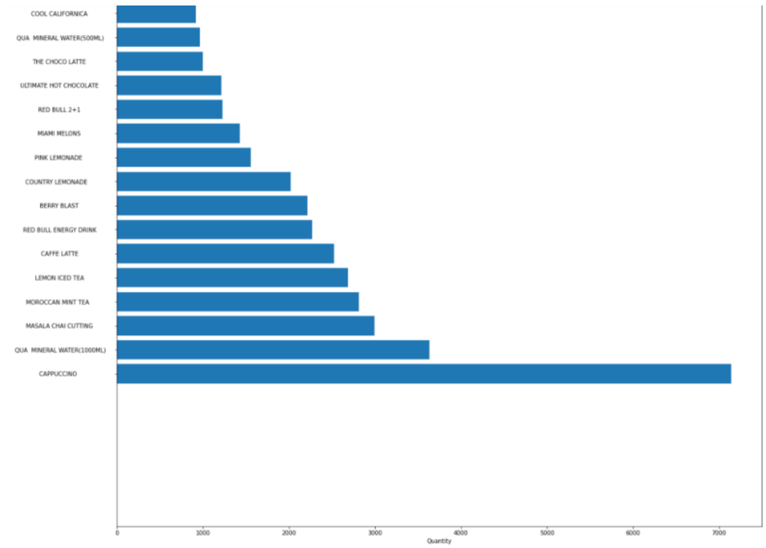
Bill Number	Quantity
G0470109	1
G0470111	4
G0470112	1
G0470114	5
G0470115	3
...	...
G0533898	5
N0027941	1
N0028811	1
N0028973	1
N0034022	2

47365 rows × 1 columns

Seeing which item from BEVERAGE is selling more

Approximately

Top 1: Cappuccino 7k
Top 2: Water 3.5k
Top 3: Masala Chai Cutting 3.1k



Getting the bill with the most ordered items

```
58
Index(['G0518491'], dtype='object', name='Bill Number ')
```

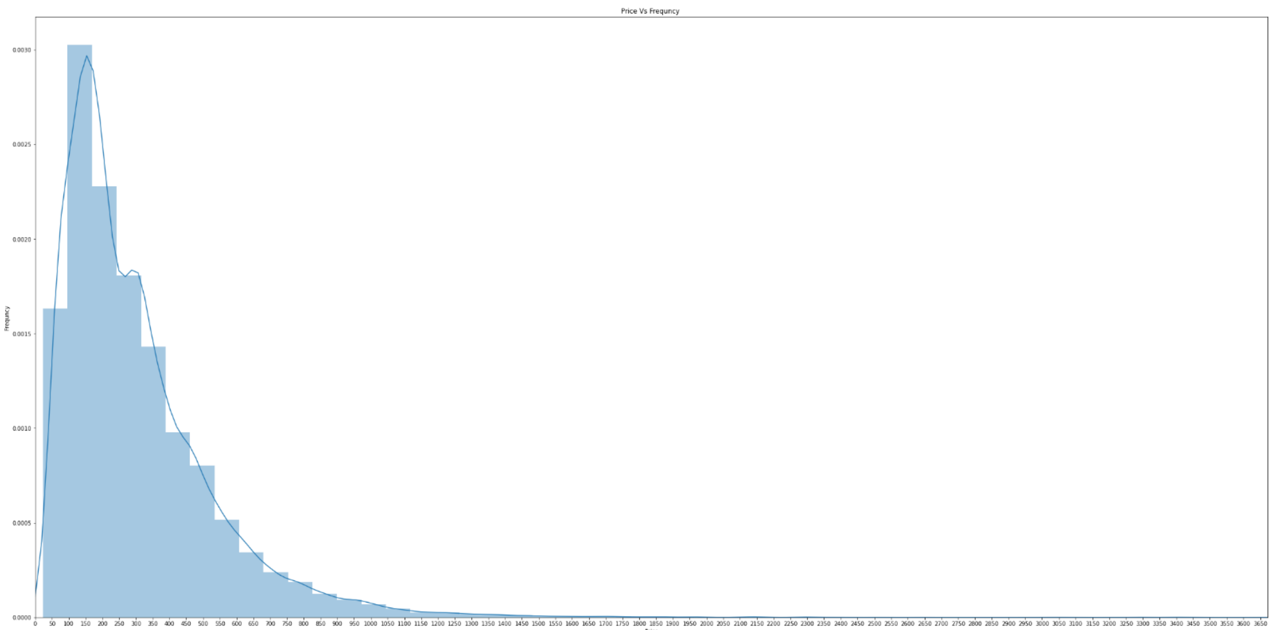
Bill with the highest number of items ordered

	date	Bill Number	Item Desc	Quantity	Rate	Tax	Discount	Total	Category
20061	2011-01-02 23:27:37	G0518491	ASSORTED TAPAS(2)VEG	3	275.0	137.16	247.5	962.16	FOOD
34677	2011-01-02 23:27:37	G0518491	CHEESE FONDUE	4	330.0	219.45	396.0	1539.45	FOOD
43070	2011-01-02 23:27:37	G0518491	COOL CALIFORNICA	1	85.0	14.13	25.5	99.13	BEVERAGE
43071	2011-01-02 23:27:37	G0518491	CURRANT COOLER	1	95.0	15.79	28.5	110.79	BEVERAGE
43072	2011-01-02 23:27:37	G0518491	BERRY BLAST	2	95.0	31.59	57.0	221.59	BEVERAGE
43073	2011-01-02 23:27:37	G0518491	MIAMI MELONS	3	85.0	42.39	76.5	297.39	BEVERAGE
43074	2011-01-02 23:27:37	G0518491	LEMON ICED TEA	6	85.0	84.79	153.0	594.79	BEVERAGE
43075	2011-01-02 23:27:37	G0518491	PINK LEMONADE	6	85.0	84.79	153.0	594.79	BEVERAGE
43230	2011-01-02 23:27:37	G0518491	HERBED CHICKEN PIE	1	135.0	22.44	40.5	157.44	FOOD
43231	2011-01-02 23:27:37	G0518491	SATAY CHICKEN PANINI	3	115.0	57.36	103.5	402.36	FOOD
43232	2011-01-02 23:27:37	G0518491	B.M.T. PANINI	5	105.0	87.28	157.5	612.28	FOOD
43233	2011-01-02 23:27:37	G0518491	GARDEN FRESH PANINI	5	105.0	87.28	157.5	612.28	FOOD
43234	2011-01-02 23:27:37	G0518491	PHILLYCREAM CHEESE &CHILLY PAN	7	105.0	122.19	220.5	857.19	FOOD
75475	2011-01-02 23:27:37	G0518491	LEMON INFUSED CHAR GRILLED VEG	6	150.0	149.63	270.0	1049.63	FOOD
92655	2011-01-02 23:27:37	G0518491	ORANGE ARRABIATA	5	150.0	124.69	225.0	874.69	FOOD

Total price of each Bill

```
Bill Number
G0470109     117.56
G0470111     495.00
G0470112      61.88
G0470114     342.13
G0470115     346.51
              ...
G0533898     748.69
N0027941    2142.00
N0028811    2142.00
N0028973    2142.00
N0034022     975.00
Length: 47365, dtype: float64
```

Seeing distribution of the total price



What we have noticed

- cafe file has No NULL values.
- Most frequently the price is between 150 to 200.
- Data is not normally distributed.
- CAPPUCCINO is the most sold item from both Categories.
- For Food: GREAT LAKES SHAKE is the most ordered item, but for BEVERAGE CAPPUCCINO is the most ordered.
- For discount CHEESE FOUNDE is the most item having a discount.
- For Food: CHEESE FOUNDE is the most item having a discount, but for BEVERAGE RED BILL 2+1 is the most item having a discount on.
- Bill (G0518491) has the most ordered items with 58 in total.

11.3 Rating_EDA

Looking at the beginning of the dataset

	coffee_shop_name	review_text	rating	num_rating	cat_rating	bool_HIGH	overall_sent	vibe_sent	tea_sent	service_sent	seating_sent	price_sent	parking_sent	location_sent	alcohol_sent	coffee_sent	food_sent	hours_sent	internet_sent	local_sent
0	The Factory - Cafe With a Soul	11/25/2016 1 check-in Love love loved the vibe! Every corner of the coffee shop had its own style, and there were swings!!! I ordered the matcha coffee, and it was muy fantastico! Ordering and getting my drink were pretty streamlined. I ordered on an iPad, which included all beverage selections that ranged from coffee to alcohol, desired level of sweetness, and a checkout system. I got my coffee within minutes! I was hoping for a typical heart or feather on my coffee, but found myself listing out all the possibilities of what the vibe may be. Any ideas?	5.0 star rating	5.0	HIGH	1.0	4.0	3	0.0	0.0	0.0	0.0	0	0.0	1.0	3	0	0.0	0.0	0.0
1	The Factory - Cafe With a Soul	12/2/2016 Listed in Date Night: Austin, vibe in Austin BEAUTIFUL!!!! Love the vibe! Instagram-worthy!!! Definitely $$$, so be prepared. This is gonna cost you a pretty penny :) food food was just decent...nothing to rave about. But, will probably be back just to be somewhere unique and nice.	4.0 star rating	4.0	HIGH	1.0	3.0	3	0.0	0.0	0.0	0.0	0	0.0	0.0	0	2	0.0	0.0	0.0
2	The Factory - Cafe With a Soul	11/30/2016 1 check-in Listed in food seating I loved the eclectic and homey plush vibe and who doesn't want to swing and drink their coffee? I would categorize this as a pricier coffee place but, to me it was worth it. After Thanksgiving nobody wants to make food so we headed out in search of food and foods. There is ample parking in the private lot and ordering is done through I-Pads. Pick a seating and they will call your name when your items are ready. Be patient because some of the coffee drinks take longer than others. I had the cold coffee coffee, food and foods and we shared a pop+art! The po+art was strawberry; light and fluffy on the outside and gooey sweet filling on the inside. The food and foods wasn't the standard so expect different, complex and interesting flavors from the accompanying slaw and sauce. I thoroughly enjoyed it but, the food was a bit bland and maybe could have used a bit of honey to spruce it up. The coffee here is spot on and I hear that their hot chocolate is the bomb but, I will have to wait for the weather to cool off to try that. A great place to be vibe and chat with your friends, a date or read a book. food and foods Pop tart Coffee See all photos from parkingmela S. for The Factory - Cafe With a Soul	4.0 star rating	4.0	HIGH	1.0	2.0	2	0.0	0.0	3.0	0.0	0	0.0	0.0	-1	2	0.0	0.0	0.0
3	The Factory - Cafe With a Soul	11/25/2016 Very cool vibe! Good drinks Nice seating However... Just about everything is bad price. $1.50 extra for 3 ounces of Almond Milk in a coffee. No WiFi vibe is a bit loud and the mix is odd. Pleasant French oldies followed by loud techno. Several seatings were dirty when we got there. Service is average. It feels like a Los Angeles coffee shot that is out of place.	2.0 star rating	2.0	LOW	0.0	1.0	0	0.0	0.0	-1.0	-1.0	0	0.0	0.0	0	0	0.0	0.0	0.0
4	The Factory - Cafe With a Soul	12/3/2016 1 check-in They are location within the Northcross mall shopping center facing east toward Burnet Rd with plenty of parking. I loved their computerized user friendly ordering system, it made it easy me to pick & choose all the items I wanted to try. I ordered pop tart, food & foods, strawberry foods (photos uploaded). This place is ideal for ordering a few dishes to share with your group. seatinging is limited since this place is fairly new with lots of visitors. Arrive early and be prepared to wait a bit if you with a large grop. Store front facing Burnet Rd See all photos from Michelle A. for The Factory - Cafe With a Soul	4.0 star rating	4.0	HIGH	1.0	2.0	0	0.0	0.0	0.0	0.0	3	0.0	0.0	0	0	0.0	0.0	0.0

checking where our dataset ends at.

	coffee_shop_name	review_text	rating	num_rating	cat_rating	bool_HIGH	overall_sent	vibe_sent	tea_sent	service_sent	seating_sent	price_sent	parking_sent	location_sent	alcohol_sent	coffee_sent	food_sent	hours_sent	internet_sent	local_sent
7616	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7617	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7618	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7619	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7620	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

checking the shape of the data: how many columns and rows.

```
[ ]  df.shape
```

```
(7621, 20)
```

Description of Dataset.

	num_rating	bool_HIGH	overall_sent	tea_sent	service_sent	seating_sent	price_sent	location_sent	alcohol_sent	hours_sent	internet_sent	local_sent
count	7616.000000	7616.000000	7616.000000	7616.000000	7616.000000	7616.000000	7616.000000	7616.000000	7616.000000	7615.000000	7616.000000	7616.000000
mean	4.169118	0.806197	1.107537	0.047006	0.325105	0.124869	0.015362	0.074711	0.042936	0.031779	0.025210	0.035583
std	1.065311	0.395302	1.177984	0.330775	0.827549	0.521658	0.381999	0.395392	0.298598	0.274642	0.277679	0.271992
min	1.000000	0.000000	-4.000000	-3.000000	-4.000000	-3.000000	-3.000000	-4.000000	-3.000000	-3.000000	-3.000000	-1.000000
25%	4.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	5.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	5.000000	1.000000	4.000000	4.000000	4.000000	4.000000	3.000000	4.000000	3.000000	3.000000	3.000000	4.000000

Checking how many unique values we have for each column.

```
df.nunique()
```

```
coffee_shop_name      78
review_text         6915
rating                 5
num_rating             5
cat_rating             2
bool_HIGH              2
overall_sent           9
vibe_sent              9
tea_sent               8
service_sent           9
seating_sent           8
price_sent             7
parking_sent           9
location_sent          9
alcohol_sent           7
coffee_sent            9
food_sent              9
hours_sent             7
internet_sent          7
local_sent             6
dtype: int64
```

Dropping some unwanted data categories.

```
newDf = df.drop(['coffee_shop_name','rating', 'cat_rating',
       'bool_HIGH', 'overall_sent', 'vibe_sent', 'tea_sent', 'service_sent',
       'seating_sent', 'price_sent', 'parking_sent', 'location_sent',
       'alcohol_sent', 'coffee_sent', 'food_sent', 'hours_sent',
       'internet_sent', 'local_sent'], axis=1)
```

Checking if deleted unwanted data is working.

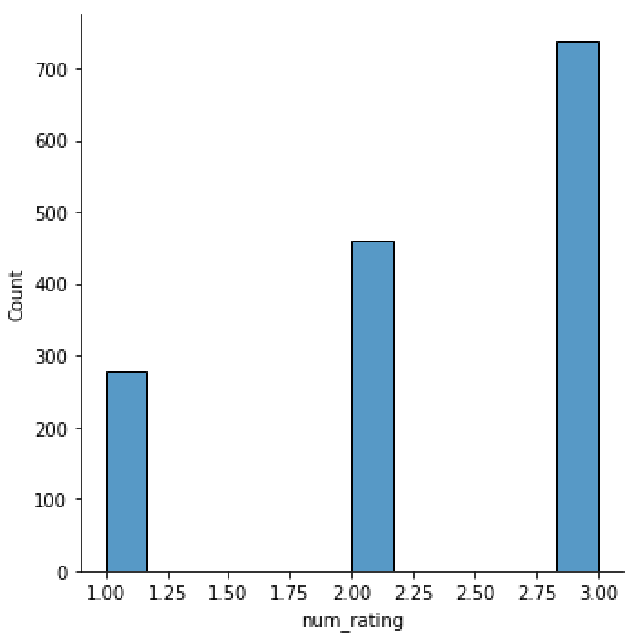
	num_rating
count	1476.000000
mean	2.311653
std	0.769168
min	1.000000
25%	2.000000
50%	2.500000
75%	3.000000
max	3.000000

Checking for null/empty values in our dataset.

```
[ ]  df.isnull().sum()

     review_text    0
     num_rating     0
     dtype: int64
```

A histogram distribution of rating on coffee shops in this dataset.



11.4 Fourth Dataset

For each category we'll represent how many training data we've as follows:

Approximately:

Intent:	**|**	**25 intents.**
Entity:	**|**	**11 Entities**
Actions:	**|**	**20 action.**
Story:	**|**	**21 stories**
Lookup:	**|**	**4 lookups.**
Synonym:	**|**	**3 synonyms.**
Rule:	**|**	**10 rules.**
Response:	**|**	**19 responses**

12. Relevant Models:

In section 6 of the proposal, we tested Rasa's retail starter pack chatbot. In this section, we will discuss two preconfigured pipelines, the components used in each, and the pros and cons of each pipeline. A pipeline consists of a sequence of components that are used to train the model bypassing the training data through the pipeline. The first pipeline is called **Pretrained_emdeddings_spacy** and the second is called **Supervised_embeddings**.

Note: **Supervised_embeddings** pipeline will be not applicable for our project, as it requires big datasets. And our project vocabulary isn't domain-specific.

Pretrained_emdeddings_spacy:

Brief Overview:

This pipeline uses the **spaCy** library which allows words to be represented as word embeddings. Word embeddings are vector representations of words. For example, we can have a word embedding model with two dimensions, the first is masculinity and the second is royalty. In this model words like a king would score high on masculinity and royalty and words like a queen would score high on royalty and low on masculinity. In reality, though, word embedding models have hundreds of dimensions. Word embedding models allow us to capture the similarity of words by making their vectors similar.

Components:

1. **SpacyNLP:** This component is used to load the Spacy language model which is used for word embeddings. This means that this component must be placed at the beginning.

2. **SpacyTokenezier:** This component is a tokenizer that splits the text into smaller chunks called tokens. This component should be one of the first since it prepares the text for subsequent components. The spacy tokenizer splits the text into words and punctuation according to predefined rules.

3. **SpacyEntityExtractor**: this component is used to extract entities from the user input. For example, if the user says, "The Dunkin store at Abu Bakr Road gave me a cappuccino instead of an americano." then the model should identify "Dunkin store" as a coffee shop, "Abu Bakr Road" as a street, "cappuccino" and "americano" as a drink.

4. **SpacyFeaturizer (Featurizer) and SklearnIntentClassifier (Intent Classifier):** These two components are used together to classify intents. The featurizer is used to convert tokens to spaCy word vectors and the intent classifier takes that vector to train a model called support vector machine (SVM). The SVM is a model used to predict the intent of the user based on text features. The output of the model is the top-ranked intent and an array of other possible intents.

Pros and Cons:

Pros	**Cons**
Less training data is needed for better model performance. This is because the pre-trained Spacy word embeddings provide the meaning of words.	Good word embeddings are not available for all languages.
The training is faster since the training is not done from scratch (**Spacy's** pre-trained word embeddings are the foundation).	Domain-Specific words are not captured by the word embeddings, since the word embeddings are based on generic data sets. For example, in Dunkin Doughnuts the word "charged" means extra coffee which could be interpreted as charged with an electric charge by spaCy.

Supervised_embeddings:

Brief Overview:

This pipeline trains the model from scratch using the training data, unlike **Pretrained_embeddings_spacy** which uses the training data along with the pre-trained spacy word embeddings.

Components:

1. **WhitespaceTokenizer:** this component is a tokenizer that uses white spaces as delimiters to separate tokens. In our case, we would not use the tokenizer Jieba which is for specific languages such as Chinese.

2. **CRFEntityExtractor and DucklingHttpExtractor**: Both of these components are used to extract entities. DucklingHttpExtractor is a specialized component used to extract specific entities, such as dates, numbers, and distances.

3. **Regex_featurizer:** this component can be added before CFREntityExtractor to assist with entity extraction if you're using regular expressions. For example, 10-digit phone numbers.

4. **CountVectorsFeaturizer (Featurizer) and EmbeddingIntentClassifier (Intent Classifier):** the CountVectorsFeaturizer creates a bag-of-words with the number of times a word appears in a text. The countvectorsFeaturizer disregards the order of words. this bag-of-words is used as input for EmbeddingIntentClassifer to predict the intent of the user's input.

Pros and Cons:

Pros	Cons
Since the model is trained by your training data only, domain-specific words and messages can be dealt with.	More training data is needed for this pipeline compared to pre-trained embeddings pipeline.
It is easier to build chatbots for any language that can be tokenized.	
This pipeline is better at handling messages with multiple intents.	

Which pipeline are we going to use:

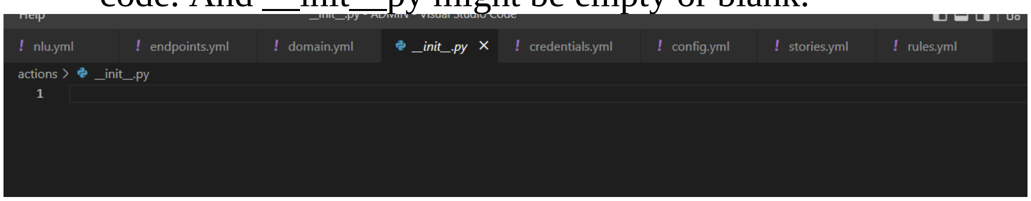
We will test 2 pipelines on our chatbot and use the pipeline that gives better results. We'll try to use the **Pretrained_embedding_spacy** pipeline which would be better for our chatbot. And we are also open to use any other pipeline we might find in the future.

13. Elementary experiments:

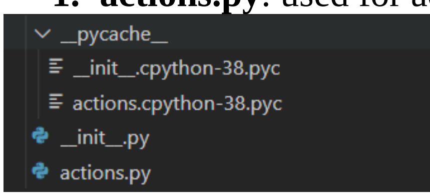
Description:

In this section, we'll discuss some ML/NLP models, features, and differentiate the difference between them in a technical approach. And we'll give a brief report about the result by the different models that will be used in our café chatbot that's **that gave the following files:**

1. **__init__py**: allow the python interpreter to know that a directory contains code. And __init__py might be empty or blank.



1. **actions.py**: used for adding custom response-like functions.



1. **nlu.yml:** examples of intents/entities.
2. **rules.yml**: put for the bot rules for the response.
3. **stories.yml:** examples of intent in sequence.

1. **config.yml:** contains pipelines/policy
2. **credentials.yml:** establishes rasa server and that contains credentials for voice and chat.
3. **endpoints.yml**.
4. **domain.yml:** provides overview of intent+ response+ custom actions.

```
! config.yml
! credentials.yml
! domain.yml
! endpoints.yml
```

14. Applied Models and Error Analysis:

So far, we'll perform 2 of the techniques that we've mentioned in section 7 using the command **(rasa shell nlu)** to see the confidence score. And we'll use 2 types of models or processing pipelines that are the following:

- **Pretrained_embedding_spacy.**
- **Custom pipeline.**

Both of them are restricted by components that each one of them is responsible for a different task.

Custom Pipeline:

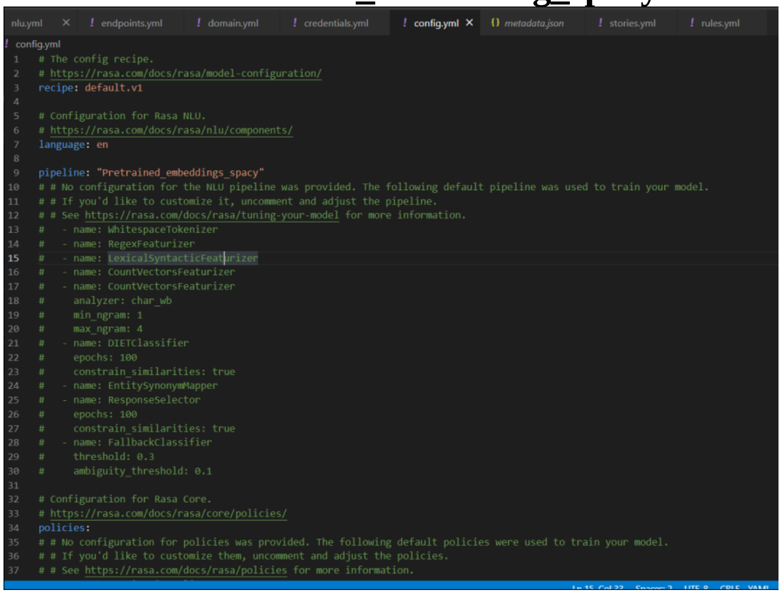
```yaml
language: en
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
  - name: EntitySynonymMapper
  - name: ResponseSelector
  - name: RegexEntityExtractor
    # text will be processed with case insensitive as default
    case_sensitive: False
    # use lookup tables to extract entities
    use_lookup_tables: True
    # use match word boundaries for lookup table
    "use_word_boundaries": True
```

Pretrained_embedding_spacy.

```yaml
language: en
pipeline:
  - name: SpacyNLP
  - name: SpacyTokenizer
  - name: SpacyFeaturizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
```

Report:

1. With **Pretrained_embedding_spacy.**



```
nlu.yml  endpoints.yml  domain.yml  credentials.yml  config.yml  metadata.json  stories.yml  rules.yml
config.yml
# The config recipe.
# https://rasa.com/docs/rasa/model-configuration/
recipe: default.v1

# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en

pipeline: "Pretrained_embeddings_spacy"
# # No configuration for the NLU pipeline was provided. The following default pipeline was used to train your model.
# # If you'd like to customize it, uncomment and adjust the pipeline.
# # See https://rasa.com/docs/rasa/tuning-your-model for more information.
#   - name: WhitespaceTokenizer
#   - name: RegexFeaturizer
#   - name: LexicalSyntacticFeaturizer
#   - name: CountVectorsFeaturizer
#   - name: CountVectorsFeaturizer
#     analyzer: char_wb
#     min_ngram: 1
#     max_ngram: 4
#   - name: DIETClassifier
#     epochs: 100
#     constrain_similarities: true
#   - name: EntitySynonymMapper
#   - name: ResponseSelector
#     epochs: 100
#     constrain_similarities: true
#   - name: FallbackClassifier
#     threshold: 0.3
#     ambiguity_threshold: 0.1

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
# # No configuration for policies was provided. The following default policies were used to train your model.
# # If you'd like to customize them, uncomment and adjust the policies.
# # See https://rasa.com/docs/rasa/policies for more information.
```

Pipeline: "Pretrained_embedding_spacy".
Components: not specified.

Steps:

-conda create -n Project_test
-conda activate Project_test
-rasa init then press → yes → yes.
model created.
Note: every time when you train the model it'll create another model and that model will be trained in response of changes you do in the code.
do want to run the project file? then press → no, since we're interested in something else.
After that, you might do some changes in the code → finish.
-rasa train → new model created.
-rasa shell nlu → it'll show the confidence of the message or the word you've written.
Finally, the confidence percentage shows up and give you expectation of how well the model did.

Results:

Example: I want to order a black coffee			
Percentage of Confidence	0.9994513392448425	0.9996902942657471	0.7742573618888855
Predicted	Intent: order_drinks	Entity: drinks	Entity: number

Example: I want to order a cookie			
Percentage of Confidence	0.9997460084213257	0.999812304973049736023	0.9995111227035522
Predicted	Intent: order_bakery	Entity: bakery	Entity: number

As we see here the result was promising. It got a high confidence score which means that the chatbot did a good job of choosing the best intent for the response.

Report:

1. With **Custom Pipeline.**

```yaml
language: en
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
  - name: EntitySynonymMapper
  - name: ResponseSelector
  - name: RegexEntityExtractor
    # text will be processed with case insensitive as default
    case_sensitive: False
    # use lookup tables to extract entities
    use_lookup_tables: True
    # use match word boundaries for lookup table
    "use_word_boundaries": True
```

Pipeline: "Custom Pipeline".
Components: specified.

Steps:

-conda create -n Project_test
-conda activate Project_test
-rasa init then press → yes → yes.
model created.
Note: every time when you train the model it'll create another model and that model will be trained in response of changes you do in the code.
do want to run the project file? then press → no, since we're interested in something else.
After that, you might do some changes in the code → finish.
-rasa train → new model created.
-rasa shell nlu → it'll show the confidence of the message or the word you've written.
Finally, the confidence percentage shows up and give you expectation of how well the model did.

Results:

Example: I want to order a black coffee			
Percentage of Confidence	0.9994513392448425	0.9996902942657471	0.7742573618888855
Predicted	Intent: order_drinks	Entity: drinks	Entity: number

Example: I want to order a cookie			
Percentage of Confidence	0.9999746084213257	0.9998123049730497 36023	0.9995111227035522
Predicted	Intent: order_bakery	Entity: bakery	Entity: number

As we see here the result was promising. It got a high confidence score which means that the chatbot did good job for choosing the best intent for the response.

In Summary:

Both models were efficient, and we might be able to use both of them. And the reason why we don’t train with **Supervised embeddings** since our chatbot will be using a small amount of data and our training will be fast.

15. Techniques used for Reducing the error:

- Make sure that the order of the component is right since the model gets affected by changing the order.
- Make sure that the intent in the training data file of classes is balanced.
- Choosing the custom pipelines to achieve the best result with less error.
- Use a visualization tool to make sure that we understand the project and class well by using this specified tool: https://github.com/RasaHQ/rasalit. That'll provide us with the following:
  - Simple text clustering.
  - GridResults Summary.
  - NLU model playground **(Very Useful).**

16. Code resources:

GitHub link: https://github.com/fouad20000/NLP_Project.git.

17. References:

1. https://bhashkarkunal.medium.com/conversational-ai-chatbot-using-rasa-nlu-rasa-core.how-dialogue-handling-with-rasa-core-can-use-331e7024f733.
2. https://analyticsindiamag.com/10-nlp-open-source-datasets-to-start-your-first-nlp-project/.
3. https://rasa.com/docs/rasa/nlu-training-data/.
4. https://analyticsindiamag.com/10-nlp-open-source-datasets-to-start-your-first-nlpproject/.
5. https://www.geeksforgeeks.org/chatbots-using-python-and-rasa/.
6. https://www.irjet.net/archives/V8/i6/IRJET-V8I6683.pdf.
7. https://www.researchgate.net/publication/234805405_Different_measurements_metrics_to_evaluate_a_chatbot_system.
8. https://arxiv.org/pdf/2009.12341.pdf.
9. https://www.researchgate.net/publication/220046725_Chatbots_Are_they_Really_Useful.
10. THE RASA MASTERCLASS HANDBOOK: A COMPANION GUIDE TO THE RASA MASTERCLASS VIDEO SERIES.