***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.

***Project***

**Report**

***CS476***

***Members***

*1****.*** Fouad Majd Alkadri |218110075

2.Abdullah Rajoub | 218110141

3. Abdulaziz Alowain |219110119

4. Ibrahim Khurfan| 218110082

***Chatbot***

**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**

[**Topic Choice:** 2](#_Toc103715954)

[**Goal:** 2](#_Toc103715955)

[**1. The Problem** 6](#_Toc103715956)

[**2. The Dataset available** 7](#_Toc103715957)

[**3. Proposed Approach and Tools to be Used** 7](#_Toc103715958)

[**3.1 Creating The NLU** 8](#_Toc103715959)

[**3.2 Processing pipelines:** 9](#_Toc103715960)

[**4. Scope of the Project** 10](#_Toc103715961)

[**4.1 Features that are included in the project** 10](#_Toc103715962)

[**4.2 Features not included in the project** 10](#_Toc103715963)

[**4.3 Interface Used** 10](#_Toc103715964)

[**5. Work Plan** 11](#_Toc103715965)

[**6. Related Work** 12](#_Toc103715966)

[**6.1 Conclusion** 13](#_Toc103715967)

[**7. Evaluate the Measures and Error Analysis Methods** 14](#_Toc103715968)

[**7.1 Experiment** 14](#_Toc103715969)

[**7.2 Measures** 15](#_Toc103715970)

[**7.3 Error Analysis Methods** 15](#_Toc103715971)

[**8. Group Management** 17](#_Toc103715972)

[**9. Problem and challenges in Chatbot:** 18](#_Toc103715973)

[**10. Dataset Specification** 19](#_Toc103715974)

[**10.1 First Dataset:** 20](#_Toc103715975)

[**10.1.1 items\_to\_id file:** 20](#_Toc103715976)

[**10.1.2 Conversation file:** 20](#_Toc103715977)

[**10.1.3 food file:** 21](#_Toc103715978)

[**10.2 Second Dataset:** 22](#_Toc103715979)

[**10.2.1 Café\_data file:** 22](#_Toc103715980)

[**10.3 Third Dataset:** 23](#_Toc103715981)

[**10.3.1** **Ratting\_and\_sentiments file:** 23](#_Toc103715982)

[**10.4 Fourth Dataset:** 23](#_Toc103715983)

[**11. Exploratory data analysis (EDA):** 24](#_Toc103715984)

[**12. Relevant Models:** 43](#_Toc103715985)

[**13. Elementary experiments:** 46](#_Toc103715986)

[**14. Applied Models and Error Analysis:** 47](#_Toc103715987)

[**15. Techniques used for Reducing the error:** 52](#_Toc103715988)

[**16. Code resources:** 52](#_Toc103715989)

[**17. References:** 52](#_Toc103715990)

# **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](http://blog.rasa.com/level-3-contextual-assistants-beyond-answering-simple-questions/). 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).

Diagram

Description automatically generated with medium confidence

*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**

صورة تحتوي على نص

تم إنشاء الوصف تلقائياًصورة تحتوي على نص

تم إنشاء الوصف تلقائياً

صورة تحتوي على نص

تم إنشاء الوصف تلقائياًصورة تحتوي على نص

تم إنشاء الوصف تلقائياً

صورة تحتوي على نص

تم إنشاء الوصف تلقائياًصورة تحتوي على نص

تم إنشاء الوصف تلقائياً

صورة تحتوي على نص

تم إنشاء الوصف تلقائياًصورة تحتوي على نص

تم إنشاء الوصف تلقائياً

# **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.

صورة تحتوي على نص, داخلي

تم إنشاء الوصف تلقائياً

1. A normal Hi.

Input: “Hi”

Output: The bot greets the user and tells him to pick a service from the list.

صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

1. 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.صورة تحتوي على نص, شاشة عرض, داخلي, لقطة شاشة

تم إنشاء الوصف تلقائياً

1. We picked the 4th service “Subscribe to product updates” and entered a valid email but an unknown domain (@mail).

Input: “[aziz@mail.co](mailto:aziz@mail.com)m”

Output: The bot asks the user to start over.

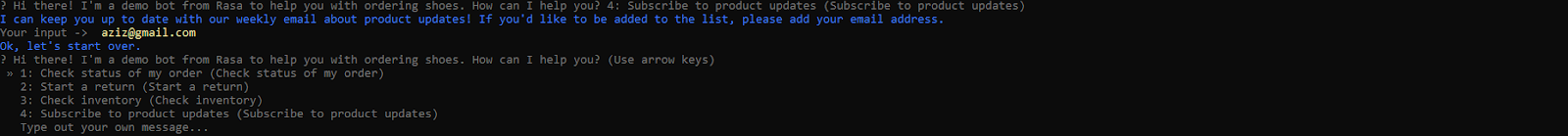
صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

1. We picked the 4th service “Subscribe to product updates” and entered a valid email domain.

Input: “[aziz@gmail.com](mailto:aziz@gmail.com)”

Output: The bot asks the user to start over. This means that there is a bug with the bot.



1. 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.

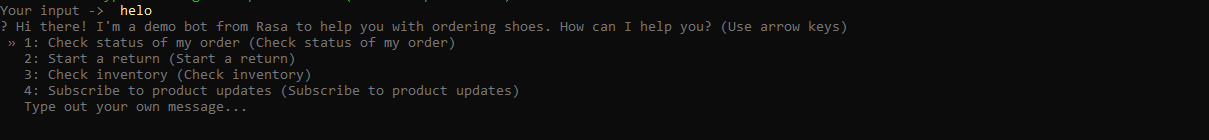
صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

1. 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**

-helloText

Description automatically generated

-hi

-yo

-wassapp

-hya

* 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**

-helloصورة تحتوي على نص

تم إنشاء الوصف تلقائياً

-hi

-yo

-wassapp

-hya

**-Hey**

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

صورة تحتوي على نص

تم إنشاء الوصف تلقائياً

* 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:**
   1. **Item\_to\_id file** contains all menu items of a coffee shop with their corresponding ID.
   2. **Conversation file:** contains all conversations that had happened between the café staff and the customer's **Questions/Answers.**
   3. **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:**
   1. **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:**
4. **Ratting\_and\_sentiments:** This file contains Yelp ratings for several coffee shops.
5. **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:**

1. **The file consists of two main columns:**
   * 1. ID: Data type (int).
     2. Name: Data type (String).
2. **File benefits:**
   * 1. It will make us keep a record of what items are being ordered the most and what items mostly appear together.
3. **File limitation:**
   * 1. It does not include all local café menu items.

### **10.1.2 Conversation file:**

1. **The file consists of two main columns:**
   * 1. Question: Data type (String).
     2. Answer: Data type (String).
2. **File benefits:**
   1. Allows us to gain valuable information on how the conversation is formed in the café and the order of them.
   2. 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?

1. **File Limitations:**
   1. It is not captured from local Café shops.
   2. It will not be able to make us recognize local beverages such as “Shahii Adani or Shaii Karak”.
   3. Limited in only one language which is English.

### **10.1.3 food file:**

1. **The file contains three main columns:**
   1. id: Data type (int).
   2. times\_appeared: Data type (int).
   3. food\_rating: Data type (int).
2. **File Benefits:**
   1. It will allow us to identify items for customers based on their popularity to appear together, which will increase the sale of the coffee.
   2. It will tell us what items are most liked and how that gets reflected in customer questions.
3. **File Limitations:**
   1. 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:**

* + - * 1. **The file contains 9 main columns:**

Date: Data type (date).

Bill Number: Data type (string).

Item Desc: Data type (string).

Quantity: Data type (int).

Rate: Data type (int).

Tax: Data type (float).

Discount: Data type (int).

Total: Data type (float).

Category: Data type (string).

* + - * 1. **File Benefits:**

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.

It contains a rate, so we know which items are the most liked.

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.

* + - * 1. **File Limitations:**

Does not include conversation.

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:

### **Ratting\_and\_sentiments file:**

1. **The file consists of twenty columns, but we are interested in only two:** 
   * 1. **Review Text: Data type (String).**
     2. **Num\_Rating: Data type (String).**
2. **File benefits:**
   * 1. This file will be used to enable the chatbot to detect complaints intent when presented with input similar to negative reviews on Yelp.
3. **File limitation:**
   * 1. 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.
     2. 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](URL:https://colab.research.google.com/drive/1G9PIBoh69xpq4f3uCfOw4jZKq54Tllj?usp=sharing).

**Fourth Dataset**

**URL:** [**https://github.com/fouad20000/NLP\_Project**](https://github.com/fouad20000/NLP_Project)**.**

* 1. **Food\_Item\_to\_id**

**Displaying file contents.**

Table

Description automatically generatedGraphical user interface, application

Description automatically generated

**Exploring labels for each file.**

Graphical user interface, text, application, email

Description automatically generated

**Exploring indices of each file.**

Graphical user interface, text, application, email

Description automatically generated

**Seeing information about each file.**

Text

Description automatically generated

**Getting a description of each file.**

Table

Description automatically generated Graphical user interface, application

Description automatically generated

**Seeing Data types in each file.**

Graphical user interface, text, application

Description automatically generated

**Checking for null values.**

Table

Description automatically generated with low confidence Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence Graphical user interface, text, application

Description automatically generated

**Checking the shape of each file.**

Graphical user interface, text, application, chat or text message

Description automatically generated

**Visualizing the datasets.**

Seeing which item has been appearing frequently

**Chart

Description automatically generated**

Seeing each item rating

**Chart, bar chart

Description automatically generated**

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.
  1. **Conversation**

**Displaying file contents.**

Text

Description automatically generated

**Exploring labels for each file.**

Graphical user interface, text, application

Description automatically generated

**Exploring indices of each file.**

Graphical user interface, text, application

Description automatically generated

**Seeing information about each file.**

Text

Description automatically generated

**Getting a description of each file.**

Graphical user interface, application

Description automatically generated

**Seeing Data types in each file.**

Text

Description automatically generated

**Checking for null values.**

Graphical user interface, table

Description automatically generatedGraphical user interface, text, application

Description automatically generated

**Checking the shape of each file.**

Text

Description automatically generated with low confidence

**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.**

Table

Description automatically generated

**Exploring labels for each file.**

Graphical user interface, text

Description automatically generated

**Exploring indices of each file.**

Graphical user interface, text, application

Description automatically generated

**Seeing information about each file.**

Table

Description automatically generated

**Getting a description of each file.**

Table

Description automatically generated

**Seeing Data types in each file.**

Table

Description automatically generated

Table

Description automatically generated**Checking for null values.**

Table

Description automatically generated

**Checking the shape of each file.**

Text

Description automatically generated with medium confidence

**Visualizing the datasets.Chart, histogram

Description automatically generated**

Top 1: Food 59k

Top 2: Beverage 43k

Top 3: Tobacco 38k

**Approximately**

Seeing which Category has been appearing frequently

**Seeing Unique values of categories**

Text

Description automatically generated

**Removing the Unwanted categories (**BEVERAGE**)**

**Grouping items based on the Bill Number**

**One Bill number example**

Graphical user interface

Description automatically generated

**Seeing the most ordered item in each bill**

A picture containing text

Description automatically generated

**Seeing the most item sold**

Text

Description automatically generated

**Seeing the item in which has more discount**

A picture containing text

Description automatically generated

Seeing Which Category Selling more

**Logo, icon

Description automatically generated**

**Counting the number of unique items in each bill**

**Making them a series in Item Description**

Graphical user interface

Description automatically generated

**Seeing each bill and total item ordered**

Graphical user interface, application

Description automatically generated

Seeing which item from BEVERAGE is selling more

**Chart, bar chart

Description automatically generated**

**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**



**Bill with the highest number of items ordered**

Table

Description automatically generated

**Total price of each Bill**

Table

Description automatically generated

**Seeing distribution of the total price**

**Shape, square

Description automatically generatedWhat 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.

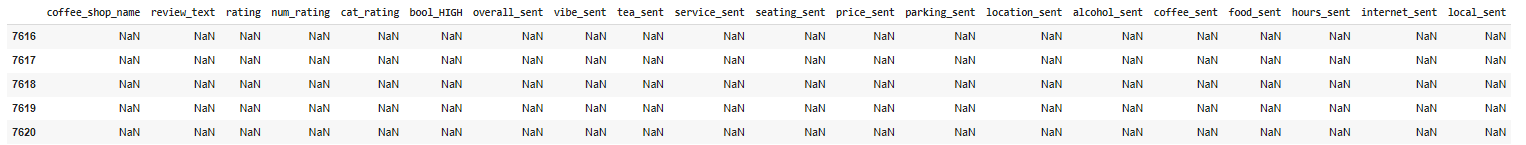
**11.3 Rating\_EDA**

**Looking at the beginning of the dataset**

A picture containing background pattern

Description automatically generated

**checking where our dataset ends at.**



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

Text

Description automatically generated with low confidence

**Description of Dataset.**

Graphical user interface

Description automatically generated

**Checking how many unique values we have for each column.**

Table

Description automatically generated with medium confidence

**Dropping some unwanted data categories.**

Text

Description automatically generated with medium confidence

**Checking if deleted unwanted data is working.**

Graphical user interface, application

Description automatically generated

**Checking for null/empty values in our dataset.**

Text

Description automatically generated with medium confidence

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

Chart, bar chart

Description automatically generated

**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.

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

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.

Graphical user interface, text

Description automatically generated with medium confidence

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.

**Graphical user interface, text, application

Description automatically generated**

# **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 theconfidence 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.

**Text

Description automatically generatedText

Description automatically generatedCustom Pipeline: Pretrained\_embedding\_spacy.**

**Report:**

1. With **Pretrained\_embedding\_spacy.**

Text

Description automatically generated

**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 | 0.9999746084213257 | 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.**

**Text

Description automatically generated**

**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 | 0.999812304973049736023 | 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 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:
  + Icon

    Description automatically generatedSimple 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.