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.

Project: Phase 3
Literature Review
CS476
Members

1. Fouad Majd Alkadri |218110075
2.Abdullah Rajoub | 218110141
3. Abdulaziz Alowain |219110119
4. Ibrahim Khurfan| 218110082
5. Hammad Ismaeel |219211202

1. Problem and challenges in Chatbot:

Chatbots are not easy to implement, design, or maintain, because of 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?

2. Dataset Specification

In this section, we are going to use 3 datasets gathered from Kaggle.
The first dataset "https://www.kaggle.com/datasets/sonalibhoir/cafe-chatbot-dataset/code?datasetId=693402&select=Item_to_id.csv." that will consist of 3 csv files:

1- items_to_id file:

a. File Description: This file contains all menu items of a coffee shop with their corresponding ID.

b. The file consists of two main columns:

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

c. File benefits:

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

d. File limitation:

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

2- Conversation file:

a. **File Description:** This file contains all conversations that had happened between the café staff and the customers.

b. The file consists of two main columns:

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

c. File benefits:

i. Allows us to gain valuable information on how 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 same meant as: One hot chocolate please.
❖ Allows to identify what is the best answer that is associated with 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?

d. 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.

3- food file:

a. **File Description:** This file is going to be used for recommendation system since it contains which items appears with others and frequency of them.

b. File contains three main columns:

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

c. File Benefits:

i. It will allow us to identify items to customer 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 get reflected in customer questions.

d. File Limitations:

i. It does not include 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 at morning side dishes differs from mid-day or late night.

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

1. Café_data file:

a. File Description: The data set provide you the dataset of a Café Chain for one of its restaurants.

b. 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).

c. File Benefits:

i. It contains date so it can give more weight in predicting items in a certain time frame since it is going to be most frequent in that time.
ii. It contains categories so we know if the item is beverage or 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.
iii. It contains rate so we know which items are the most liked.
iv. It contains bill number 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 recommendation system.

d. File Limitations:

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

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

1. Ratting_and_sentiments file:

a. File Description: This file contains Yelp ratings for a number of coffee shops.

b. 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).

c. 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.

d. 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 clearly indicate the category of service (food/drinks/customer service) that lead to this review.

3. 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 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's used and have been applied by **Google Collaboratory that 'll attached here:**

Cafe_Data.ipynb.
URL:https://colab.research.google.com/drive/1pisjnoEgY2IHdPRHQQHI4s7Fj_JiNNO9?usp=sharing.
Converstaion.ipynb.
URL:https://colab.research.google.com/drive/1wU2TnrvL6qT3uZwWdXqwoJWKd_Qlm-ah?usp=sharing.
Food_Item_to_id.ipynb
URL:https://colab.research.google.com/drive/1qM7xNPJe8ypIWEFXVHHXpNjm44hqysw?usp=sharing.
Rating_EDA.ipynb
URL:https://colab.research.google.com/drive/1G9PIBoh69xpq4f3uCfOw4jZKq54Tllj?usp=sharing.

4. 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 by passing the training data through the pipeline. The first pipeline is called **Pretrained_emdeddings_spacy** and the second is called **Supervised_embeddings**.

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:

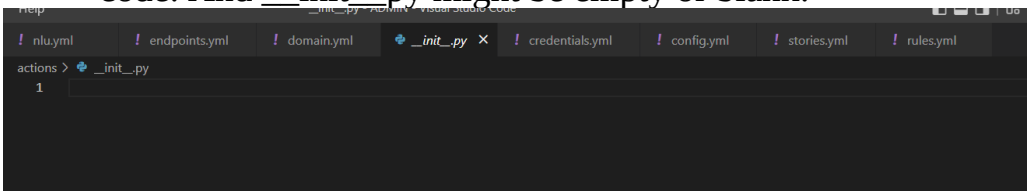
We will test both pipelines on our chatbot and use the pipeline that gives better results. However, since we have domain-specific words and messages with multiple intents, we predict that the **supervised_embeddings** pipeline would be better for our chatbot. We are also open to use any other pipeline we might find in the future.

5. Elementary experiments:

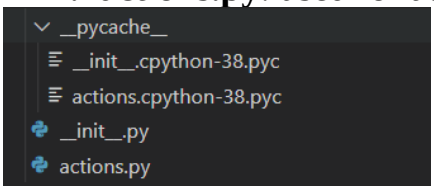
Description:

In this section, we'll discuss some ML/NLP models, features, and differentiate the difference between them in technical approach. And we'll give a brief report about the result by the different models that will be used by using the starter pack that's a pre-made project **that provides the following files:**

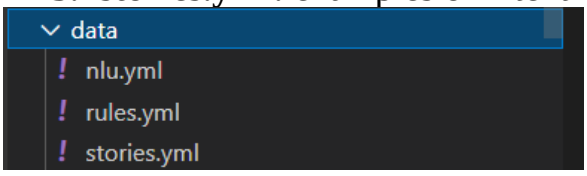
1. **__init__py**: allow the python interpreter to know that a directory contains code. And __init__py might be empty or blank.



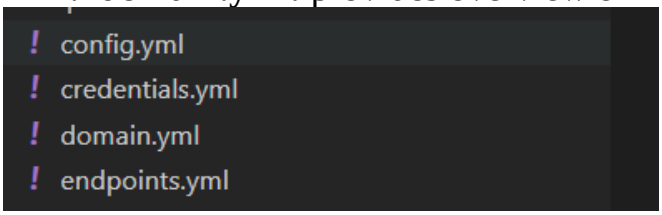
1. **actions.py**: used for adding custom response like functions.



1. **nlu.yml:** examples of intents/entities.
2. **rules.yml**: put for the bot rules for 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.



6. Applied Models and Error Analysis:

So far, we'll perform one 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 3 types of models or processing pipelines that are the following:

- **Pretrained_embedding_spacy.**
- **Supervised_embeddings.**

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

- With Specified Components.

And for the last model we'll use all components for our training in this project.

Report:

In this report we'll provide a detailed follow up steps and show how we'll train the project with these specified models.

1. With specified Components.

```yaml
pipeline:
# # 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
```

Pipeline: "empty".
Components: specified.

Steps:

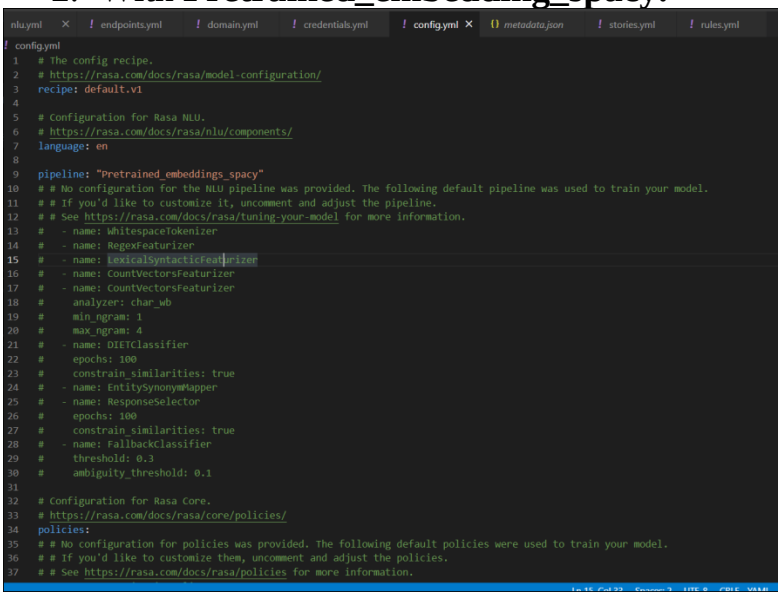
-conda create -n Project_test
-conda activate Project_test
-rasa init then press → yes → yes.
model created.
Note: every time when you train the model it'll create another model and that model will be trained in response of changes you do in the code.
do want to run the project file? then press → no, 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 in the command line interface.
Finally, the confidence percentage shows up and give you expectation of how well the model did.

Results:

As we see here the result was promising. it got a high confidence score which means the chatbot did not get confused for choosing the best intent for the response, but it has also been disadvantaged since it'll take a very long time for computation time for training this chatbot.

Note: make sure that you're using GPU during training the chatbot by checking the TensorFlow if it's working on GPU or not.

2. With **Pretrained_embedding_spacy.**

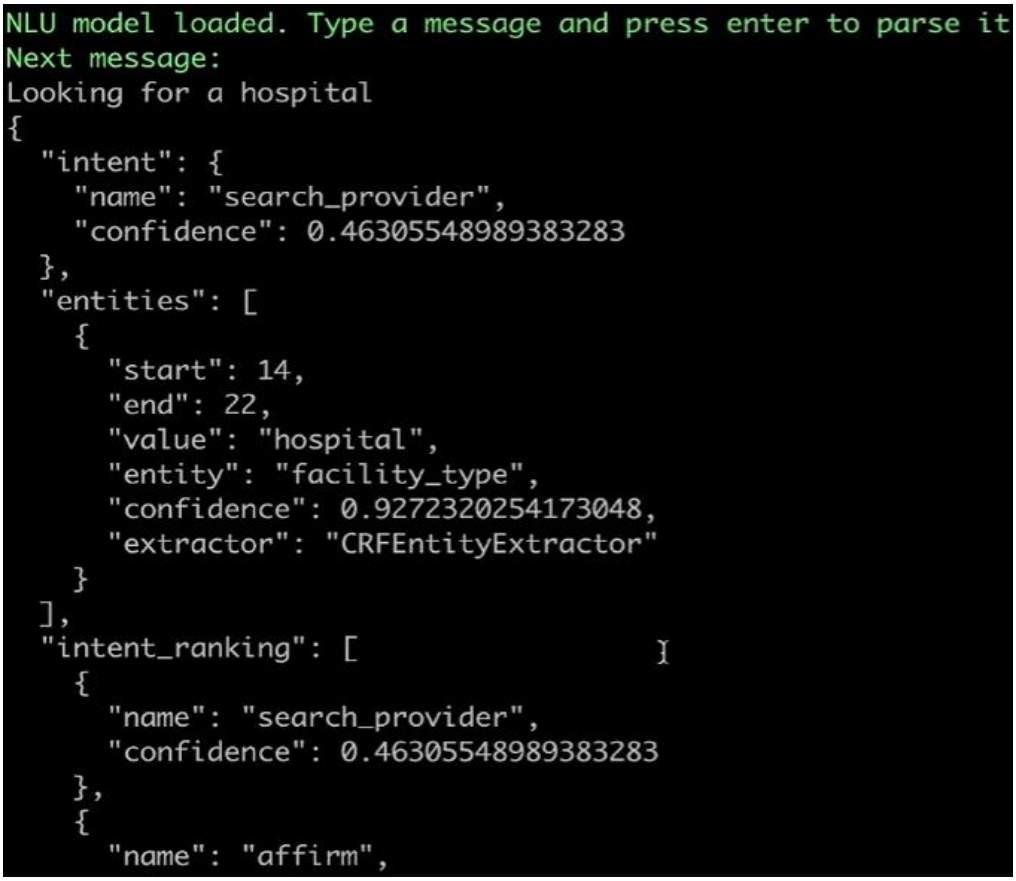


Pipeline: "Pretrained_embedding_spacy".
Components: not specified.

Steps:

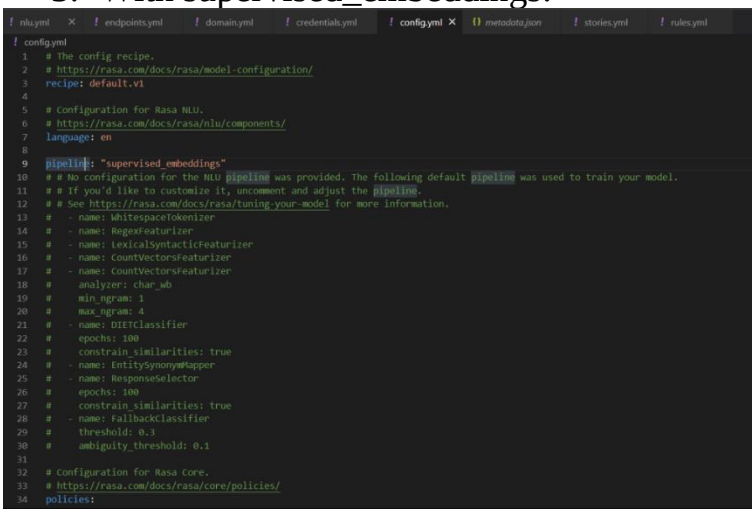
-conda create -n Project_test
-conda activate Project_test
-rasa init then press → yes → yes.
model created.
Note: every time when you train the model it'll create another model and that model will be trained in response of changes you do in the code.
do want to run the project file? then press → no, 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:



As we see here the result was promising. It got a low confidence score which means the chatbot did get confused for choosing the best intent for the response, but it has also the advantage that it'll take a very short amount of computation time for training this chatbot.

3. With supervised_embeddings.

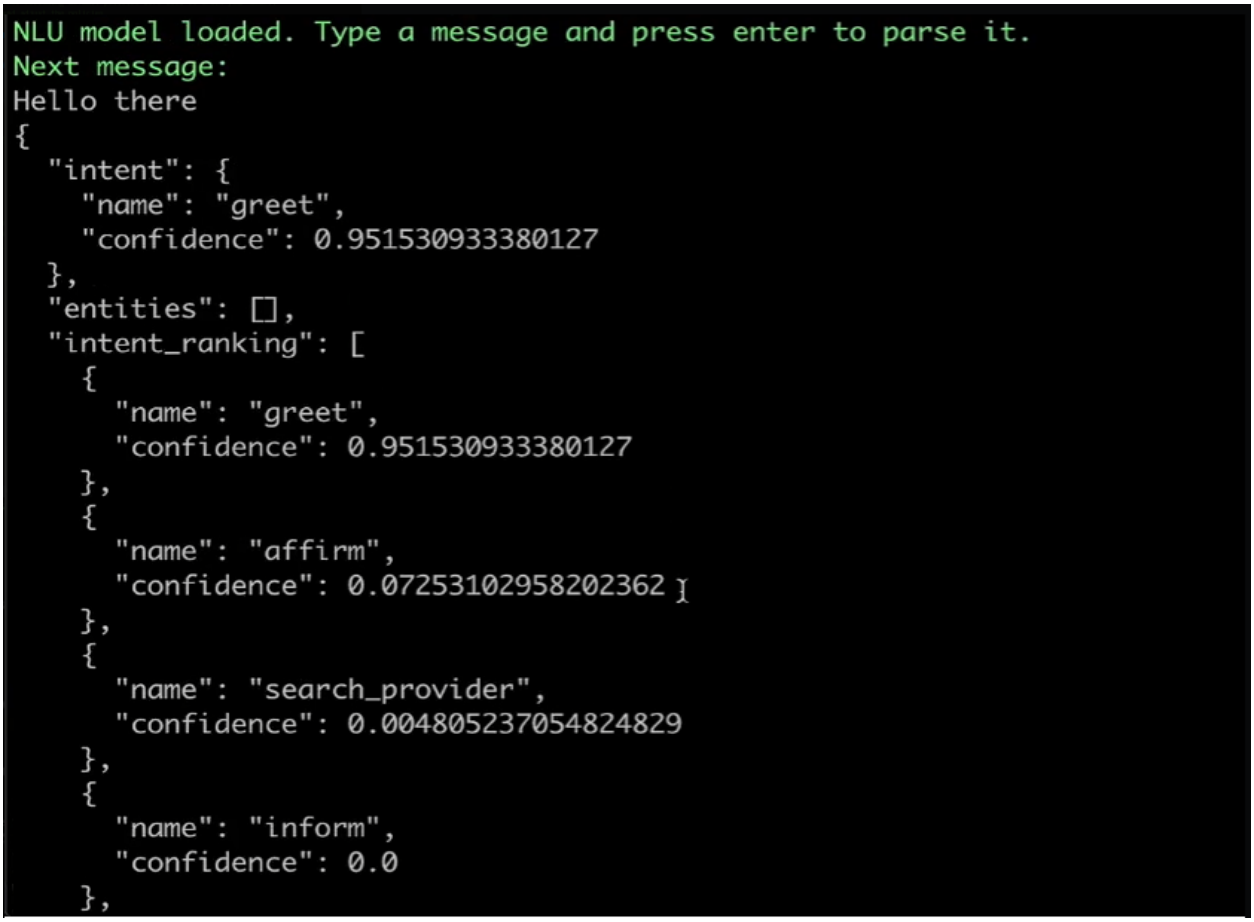


Pipeline: "supervised_embeddings".
Components: not specified.

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:



As we see here the result was promising. It got a fair confidence score which means the chatbot knows how to choose the intent for the response, but it has also the advantage that it handles multiple intents, unlike the last model that got a confused or bad result.

In Summary:

the first model was more efficient than other models that have used, but it takes more time than usual.

7. Techniques used for Reducing the error:

- Making 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).**

8. Code resources:

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

9. 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.