Project Report:

Title: Chatbots

-A.N.V.Hari Charan

Chatbots

Introduction

This project involves the development and analysis of a chatbot system using Rasa, an open-source conversational AI platform, and machine learning techniques for intent classification. The chatbot is designed to understand user intents and provide relevant responses, making it suitable for customer service, personal assistants, and other applications. The project explores both rule-based approaches and supervised learning methods to achieve robust intent recognition and conversational capabilities.

Requirements

Hardware Requirements

A system with at least 8 GB RAM (16 GB recommended for large datasets).

Minimum 10 GB of free storage space.

Software Requirements

Python 3.7 or later.

Libraries:

- Rasa
- Pandas
- NumPy
- Scikit-learn
- NLTK

Jupyter Notebook or Google Colab for development.

1.Chatbot Analysis and Intent Classification

Purpose

To preprocess, train, and evaluate a machine learning model for intent classification based on a dataset of customer service interactions.

Dataset

Source: Customer Service Training Dataset.

Structure: Contains user queries labeled with their corresponding intents.

Steps Involved

Data Loading and Exploration:

The dataset is loaded into a pandas DataFrame and inspected for missing values.

```
[1] import pandas as pd
    import numpy as np
    from sklearn import preprocessing
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.pipeline import Pipeline
    from sklearn.metrics import classification_report
    from sklearn.linear_model import LogisticRegression

    import nltk
    try:
        nltk.data.find('tokenizers/punkt')
    except:
        nltk.download('punkt')

    df = pd.read_csv("/content/Bitext_Sample_Customer_Service_Training_Dataset.csv")
    df.head()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

	flags	utterance	category	intent
0	BM	I have problems with canceling an order	ORDER	cancel_order
1	BIM	how can I find information about canceling ord...	ORDER	cancel_order
2	B	I need help with canceling the last order	ORDER	cancel_order

Preprocessing the Data:

Intent labels are encoded using LabelEncoder for numerical representation.

```
[3] len(df.intent.value_counts())
```

```
27
```

```
label_intent = preprocessing.LabelEncoder()
df['label_num'] = label_intent.fit_transform(df.intent)
df.head()
```

	flags	utterance	category	intent	label_num
0	BM	I have problems with canceling an order	ORDER	cancel_order	0
1	BIM	how can I find information about canceling ord...	ORDER	cancel_order	0
2	B	I need help with canceling the last order	ORDER	cancel_order	0
3	BIP	could you help me cancelling the last order I ...	ORDER	cancel_order	0
4	B	problem with cancelling an order I made	ORDER	cancel_order	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

Text Feature Extraction:

Text data is transformed using TF-IDF vectorization to create numerical features for the model.

```
[4] label_intent.__dict__
```

```
{'classes_': array(['cancel_order', 'change_order', 'change_shipping_address',
       'check_cancellation_fee', 'check_invoice', 'check_payment_methods',
       'check_refund_policy', 'complaint', 'contact_customer_service',
       'contact_human_agent', 'create_account', 'delete_account',
       'delivery_options', 'delivery_period', 'edit_account',
       'get_invoice', 'get_refund', 'newsletter_subscription',
       'payment_issue', 'place_order', 'recover_password',
       'registration_problems', 'review', 'set_up_shipping_address',
       'switch_account', 'track_order', 'track_refund'], dtype=object)}
```

```
X_train, X_test, y_train, y_test = train_test_split(df.utterance, df.label_num, test_size=0.2, random_state=2022, stratify=df.label_num)
print("Shape of X_train: ", X_train.shape)
print("Shape of X_test: ", X_test.shape)

clf_tfid = Pipeline([
    ('vectorizer_tfidf', TfidfVectorizer()),
    ('LogisticRegress', LogisticRegression(C=1.0, penalty='l2', max_iter=5))
])
clf_tfid.fit(X_train, y_train)

y_pred = clf_tfid.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
Shape of X_train:  (6540,)
Shape of X_test:  (1635,)
```

Splitting the Dataset:

The data is split into training and testing sets for model evaluation.

	precision	recall	f1-score	support
0	0.86	0.98	0.92	61
1	0.94	0.80	0.87	61
2	0.97	0.97	0.97	59
3	1.00	0.98	0.99	60
4	1.00	0.97	0.98	65
5	1.00	0.97	0.98	59
6	0.98	0.98	0.98	60
7	1.00	0.95	0.97	60
8	0.98	1.00	0.99	60
9	1.00	0.97	0.98	59
10	0.93	0.83	0.88	60
11	1.00	0.77	0.87	60
12	0.98	1.00	0.99	60
13	1.00	0.98	0.99	60
14	1.00	0.92	0.96	59
15	0.97	1.00	0.98	65
16	1.00	0.97	0.98	59
17	1.00	0.98	0.99	59
18	1.00	1.00	1.00	65
19	0.98	0.97	0.98	61
20	0.98	0.97	0.97	59
21	0.61	1.00	0.76	59
22	1.00	1.00	1.00	63
23	1.00	0.97	0.98	61
24	1.00	0.95	0.97	58
25	0.95	1.00	0.98	61
26	0.98	1.00	0.99	62
accuracy			0.96	1635
macro avg	0.97	0.96	0.96	1635

Evaluating the Model:

The model's performance is evaluated using metrics like precision, recall, and F1-score.

```
from sklearn.feature_extraction.text import CountVectorizer

clf_bow = Pipeline([
    ('vectorizer_tfidf', CountVectorizer()),
    ('LogisticRegress', LogisticRegression(C=1.0, penalty='l2', max_iter=5))
])

clf_bow.fit(X_train, y_train)

y_pred = clf_bow.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        61
           1       0.94      0.79      0.86        61
           2       0.97      1.00      0.98        59
           3       0.97      0.98      0.98        60
           4       1.00      1.00      1.00        65
           5       1.00      0.98      0.99        59
           6       1.00      0.98      0.99        60
           7       0.98      0.98      0.98        60
           8       0.98      1.00      0.99        60
           9       0.98      0.98      0.98        59
          10       0.92      0.92      0.92        60
          11       0.92      0.95      0.93        60
          12       0.98      1.00      0.99        60
          13       1.00      0.98      0.99        60
          14       0.98      0.93      0.96        59
```

Outcome

A trained intent classification model capable of accurately predicting user intents based on text input. Performance metrics can guide further improvement.

```
          23       1.00      0.97      0.98        61
          24       0.97      0.98      0.97        58
          25       0.95      1.00      0.98        61
          26       0.98      1.00      0.99        62

    accuracy                           0.97      1635
   macro avg       0.97      0.97      0.97      1635
weighted avg       0.97      0.97      0.97      1635

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Start coding or generate with AI.
```

2. Setting Up the Chatbot with Rasa

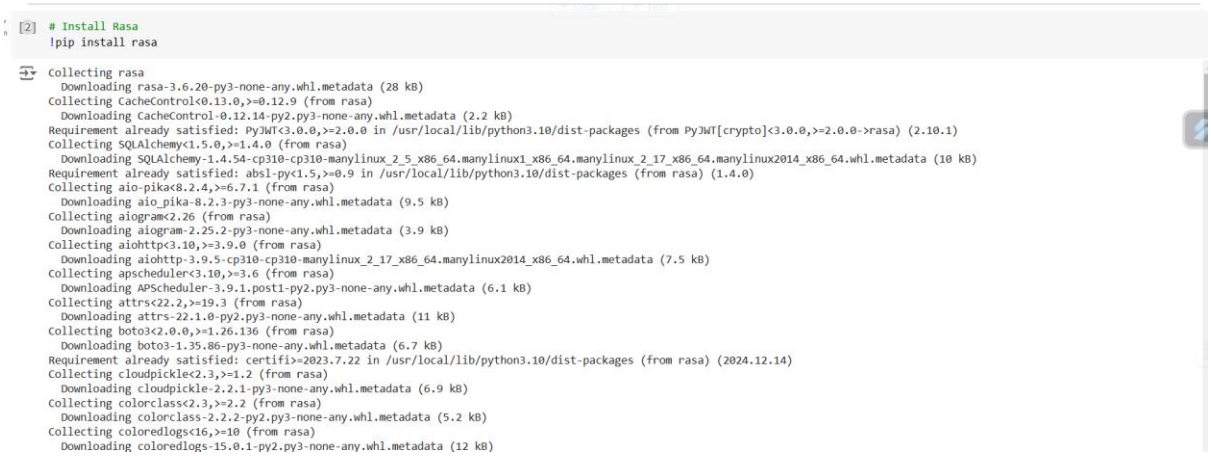
Purpose

To create a conversational agent using Rasa, capable of understanding user intents and responding appropriately.

Steps Involved

Installing Rasa:

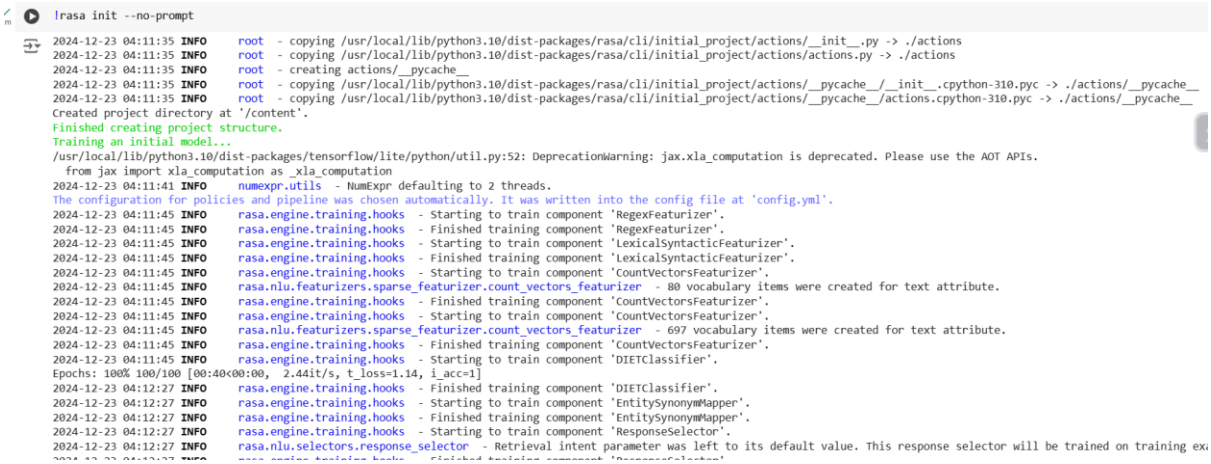
Rasa is installed using the pip package manager.



```
[2] # Install Rasa
    !pip install rasa

Collecting rasa
  Downloading rasa-3.6.20-py3-none-any.whl.metadata (28 kB)
Collecting CacheControl<0.13.0,>=0.12.9 (from rasa)
  Downloading CacheControl-0.12.14-py2.py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: PyJWT<3.0.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from PyJWT[crypto]<3.0.0,>=2.0.0->rasa) (2.10.1)
Collecting SQLAlchemy<1.5.0,>=1.4.0 (from rasa)
  Downloading SQLAlchemy-1.4.54-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (10 kB)
Requirement already satisfied: absl-py<1.5,>=0.9 in /usr/local/lib/python3.10/dist-packages (from rasa) (1.4.0)
Collecting aio-pika<8.2.4,>=6.7.1 (from rasa)
  Downloading aio_pika-8.2.3-py3-none-any.whl.metadata (9.5 kB)
Collecting aiogram<2.26 (from rasa)
  Downloading aiogram-2.25.2-py3-none-any.whl.metadata (3.9 kB)
Collecting aiohttp<3.10,>=3.9.0 (from rasa)
  Downloading aiohttp-3.9.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (7.5 kB)
Collecting apscheduler<3.10,>=3.6 (from rasa)
  Downloading APScheduler-3.9.1.post1-py2.py3-none-any.whl.metadata (6.1 kB)
Collecting attrs<22.2,>=19.3 (from rasa)
  Downloading attrs-22.1.0-py2.py3-none-any.whl.metadata (11 kB)
Collecting boto3<2.0.0,>=1.26.136 (from rasa)
  Downloading boto3-1.35.86-py3-none-any.whl.metadata (6.7 kB)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from rasa) (2024.12.14)
Collecting cloudpickle<2.3,>=1.2 (from rasa)
  Downloading cloudpickle-2.2.1-py3-none-any.whl.metadata (6.9 kB)
Collecting colorclass<2.3,>=2.2 (from rasa)
  Downloading colorclass-2.2.2-py2.py3-none-any.whl.metadata (5.2 kB)
Collecting coloredlogs<16,>=10 (from rasa)
  Downloading coloredlogs-15.0.1-py2.py3-none-any.whl.metadata (12 kB)
```

Initializing the Rasa Project:

A Rasa project is initialized with default settings.



```
!rasa init --no-prompt

2024-12-23 04:11:35 INFO     root  - copying /usr/local/lib/python3.10/dist-packages/rasa/cli/initial_project/actions/__init__.py -> ./actions
2024-12-23 04:11:35 INFO     root  - copying /usr/local/lib/python3.10/dist-packages/rasa/cli/initial_project/actions/actions.py -> ./actions
2024-12-23 04:11:35 INFO     root  - creating actions/__pycache__
2024-12-23 04:11:35 INFO     root  - copying /usr/local/lib/python3.10/dist-packages/rasa/cli/initial_project/actions/__pycache__/__init__.cpython-310.pyc -> ./actions/__pycache__
2024-12-23 04:11:35 INFO     root  - copying /usr/local/lib/python3.10/dist-packages/rasa/cli/initial_project/actions/__pycache__/actions.cpython-310.pyc -> ./actions/__pycache__
Created project directory at '/content'.
Finished creating project structure.
Training an initial model...
/usr/local/lib/python3.10/dist-packages/tensorflow/lite/python/util.py:52: DeprecationWarning: jax.xla_computation is deprecated. Please use the AOT APIs.
  from jax import xla_computation as _xla_computation
2024-12-23 04:11:41 INFO     numexpr.utils  - NumExpr defaulting to 2 threads.
The configuration for policies and pipeline was chosen automatically. It was written into the config file at 'config.yml'.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Starting to train component 'RegexFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Finished training component 'RegexFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Starting to train component 'LexicalSyntacticFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Finished training component 'LexicalSyntacticFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Starting to train component 'CountVectorsFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.nlu.featurizers.sparse_featurizer.count_vectors_featurizer  - 80 vocabulary items were created for text attribute.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Finished training component 'CountVectorsFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Starting to train component 'CountVectorsFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.nlu.featurizers.sparse_featurizer.count_vectors_featurizer  - 697 vocabulary items were created for text attribute.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Finished training component 'CountVectorsFeaturizer'.
2024-12-23 04:11:45 INFO     rasa.engine.training.hooks  - Starting to train component 'DIETClassifier'.
Epochs: 100% 100/100 [00:40<00:00,  2.44it/s, t_loss=1.14, i_acc=1]
2024-12-23 04:12:27 INFO     rasa.engine.training.hooks  - Finished training component 'DIETClassifier'.
2024-12-23 04:12:27 INFO     rasa.engine.training.hooks  - Starting to train component 'EntitySynonymMapper'.
2024-12-23 04:12:27 INFO     rasa.engine.training.hooks  - Finished training component 'EntitySynonymMapper'.
2024-12-23 04:12:27 INFO     rasa.engine.training.hooks  - Starting to train component 'ResponseSelector'.
2024-12-23 04:12:27 INFO     rasa.nlu.selectors.response_selector  - Retrieval intent parameter was left to its default value. This response selector will be trained on training ex
```

Defining NLU Data:

The natural language understanding (NLU) data is defined in a YAML format.

```
nlu_data = """
version: "3.0"
nlu:
  - intent: greet
    examples: |
      - hey
      - hello
      - hi
      - good morning
      - good evening

  - intent: bye
    examples: |
      - bye
      - goodbye
      - see you later
      - have a nice day

  - intent: affirm
    examples: |
      - yes
      - indeed
      - of course
      - that sounds good

  - intent: deny
    examples: |
      - no
      - never
```

Defining Domain Data:

The Domain data is defined in a YAML format.

```
domain_data = """
intents:
  - greet
  - bye
  - affirm
  - deny

responses:
  utter_greet:
    - text: "Hello! How can I help you today?"
  utter_bye:
    - text: "Goodbye! Have a nice day!"
  utter_happy:
    - text: "Great to hear!"
  utter_sad:
    - text: "I'm sorry to hear that."

actions:
  - action_greet
  - action_bye
  - action_happy
  - action_sad


"""
with open("domain.yml", "w") as f:
    f.write(domain_data)
```

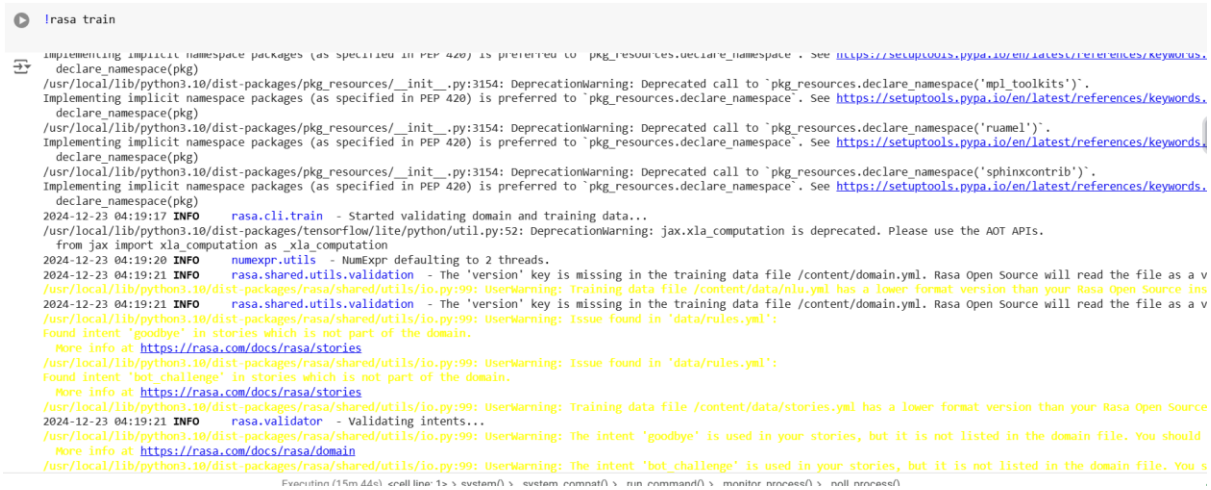
Defining Stories Data:

The Stories data is defined in a YAML format.

```
stories_data = """
version: "3.0"
stories:
  - story: greet user
    steps:
      - intent: greet
      - action: utter_greet

  - story: say goodbye
    steps:
      - intent: bye
      - action: utter_bye

  - story: affirm something
    steps:
      - intent: affirm
      - action: utter_affirm

  - story: deny something
    steps:
      - intent: deny
      - action: utter_deny
"""
with open("data/stories.yml", "w") as f:
    f.write(stories_data)
```

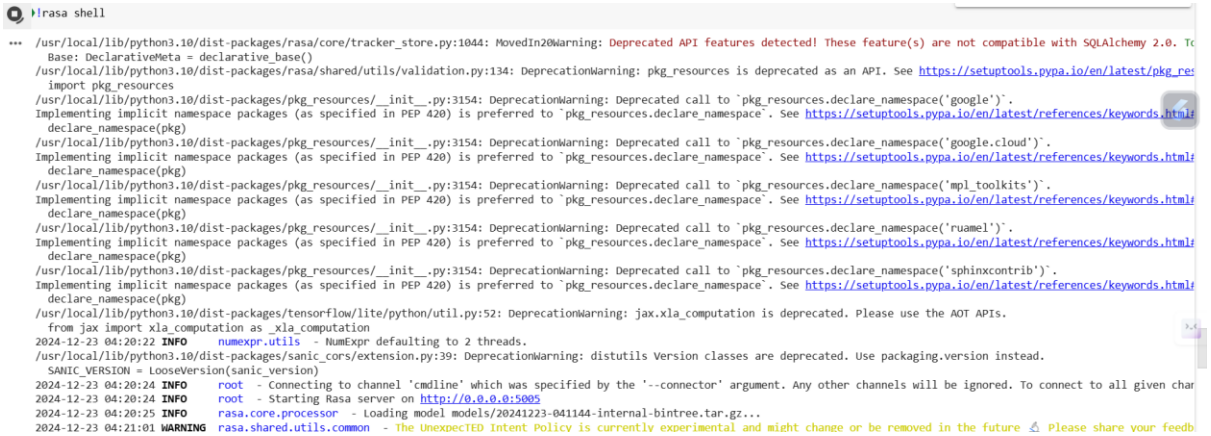
Training the Model:

The model is trained using the command:



Running the Chatbot:

To test the chatbot locally:



Outcome

A Rasa-based chatbot is successfully initialized, trained, and ready for further customization with more intents and stories.

```
WARNING: your terminal doesn't support cursor position requests (CPR).

Hey! How are you?
WARNING: your terminal doesn't support cursor position requests (CPR).

Hey! How are you?
WARNING: your terminal doesn't support cursor position requests (CPR).

Bye
WARNING: your terminal doesn't support cursor position requests (CPR).

I am a bot, powered by Rasa.
WARNING: your terminal doesn't support cursor position requests (CPR).

Bye
WARNING: your terminal doesn't support cursor position requests (CPR).

Hey! How are you?
WARNING: your terminal doesn't support cursor position requests (CPR).

Great, carry on!
WARNING: your terminal doesn't support cursor position requests (CPR).
```

Conclusion

The project combines rule-based and machine learning approaches to chatbot development. The Rasa-based setup provides flexibility for conversational design, while the machine learning model enhances the system's ability to classify intents with high accuracy. Further work can include:

Expanding the NLU training data.

Integrating the ML model into the Rasa pipeline.

Deploying the chatbot for real-world usage.