UNIFIED MENTOR
YOUR SKILL, SUCCESS & JOURNEY

| | |
|---|---|
| Project Title | Chatbots |
| Tools | Jupyter Notebook and VS code |
| Technologies | Machine learning |
| Domain | Data Science |
| Project Difficulties level | Advanced |

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

# About Dataset

## Bitext Sample Pre-built Customer Support Dataset for English

### Overview

This dataset contains example utterances and their corresponding intents from the Customer Support domain. The data can be used to train intent recognition models Natural Language Understanding (NLU) platforms.

The dataset covers the "Customer Support" domain and includes 27 intents grouped in 11 categories. These intents have been selected from Bitext's collection of 20 domain-specific datasets (banking, retail, utilities…), keeping the intents that are common across domains. See below for a full list of categories and intents.

## Utterances

The dataset contains over 20,000 utterances, with a varying number of utterances per intent. These utterances have been extracted from a larger dataset of 288,000 utterances (approx. 10,000 per intent), including language register variations such as politeness, colloquial, swearing, indirect style… To select the utterances, we use stratified sampling to generate a dataset with a general user language register profile.

The dataset also reflects commonly ocurring linguistic phenomena of real-life chatbots, such as:

- spelling mistakes
- run-on words
- missing punctuation

## Contents

Each entry in the dataset contains an example utterance from the Customer Support domain, along with its corresponding intent, category and additional linguistic information. Each line contains the following four fields:

- flags: the applicable linguistic flags
- utterance: an example user utterance
- category: the high-level intent category
- intent: the intent corresponding to the user utterance

## Linguistic flags

The dataset contains annotations for linguistic phenomena, which can be used to adapt bot training to different user language profiles. These flags are:
B - Basic syntactic structure
S - Syntactic structure
L - Lexical variation (synonyms)

M - Morphological variation (plurals, tenses…)

I - Interrogative structure

C - Complex/Coordinated syntactic structure

P - Politeness variation

Q - Colloquial variation

W - Offensive language

E - Expanded abbreviations (I'm -> I am, I'd -> I would…)

D - Indirect speech (ask an agent to…)

Z - Noise (spelling, punctuation…)

These phenomena make the training dataset more effective and make bots more accurate and robust.

## Categories and Intents

The intent categories covered by the dataset are:

ACCOUNT

CANCELLATION_FEE

CONTACT

DELIVERY

FEEDBACK

INVOICES

NEWSLETTER

ORDER

PAYMENT

REFUNDS

SHIPPING

The intents covered by the dataset are:

cancel_order

complaint

contact_customer_service

contact_human_agent

create_account

change_order

change_shipping_address

check_cancellation_fee

check_invoices

check_payment_methods
check_refund_policy
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

## Chatbots Machine Learning Project

### Project Overview

The Chatbots Machine Learning project involves developing a conversational agent (chatbot) capable of interacting with users in natural language. This can include answering questions, providing information, performing tasks, or holding a conversation. The project leverages natural language processing (NLP) and machine learning techniques to build and train the chatbot.

### Project Steps

1. **Understanding the Problem**
    - The goal is to build a chatbot that can understand and respond to user queries

## Chatbots Machine Learning Project

**Project Overview**

The Chatbots Machine Learning project involves developing a conversational agent (chatbot) capable of interacting with users in natural language. This can include answering questions, providing information, performing tasks, or holding a conversation. The project leverages natural language processing (NLP) and machine learning techniques to build and train the chatbot.

**Project Steps**

1. **Understanding the Problem**
    - The goal is to build a chatbot that can understand and respond to user queries effectively and efficiently.
    - Define the scope of the chatbot: customer support, personal assistant, FAQ bot, etc.
2. **Dataset Preparation**
    - **Data Sources**: Collect data from chat logs, customer support transcripts, or public datasets such as the Cornell Movie Dialogues Corpus.
    - **Features**: Text of user queries, context information (if available), and corresponding responses.
    - **Labels**: Responses or actions the chatbot should take.
3. **Data Exploration and Preprocessing**
    - Clean the text data by removing special characters, punctuation, and stop words.
    - Tokenize the text and convert it into numerical representations using techniques like TF-IDF, word embeddings (Word2Vec, GloVe), or BERT embeddings.
    - Split the dataset into training, validation, and testing sets.
4. **Model Selection and Training**
    - Choose appropriate NLP models based on the complexity and requirements of the chatbot. Common choices include:
        - Rule-based models
        - Retrieval-based models
        - Generative models (Seq2Seq, Transformer-based models like GPT-3, BERT)

- Train the model on the training data and fine-tune it on the validation data.

5. **Model Evaluation**
   - Evaluate the model using metrics like BLEU score, ROUGE score, perplexity, and user satisfaction ratings.
   - Perform qualitative evaluation by having users interact with the chatbot and provide feedback.
6. **Dialog Management**
   - Implement a dialog management system to handle context and state tracking.
   - Use frameworks like Rasa, Microsoft Bot Framework, or Dialogflow to manage dialog flow and context.
7. **Deployment**
   - Deploy the chatbot using platforms like Flask, Django, or a cloud service like AWS Lambda.
   - Integrate the chatbot with messaging platforms (e.g., Facebook Messenger, Slack, WhatsApp) or websites.
8. **Continuous Improvement**
   - Collect user interactions and feedback to continuously improve the chatbot.
   - Regularly update the model with new data and retrain it to handle new queries and scenarios.
9. **Documentation and Reporting**
   - Document the entire process, including data collection, preprocessing, model training, evaluation, and deployment.
   - Create a final report or presentation summarizing the project, results, and insights.

**Sample Code**

Here's a basic example using Python and the Rasa framework to build a simple chatbot:

```
# Install Rasa
!pip install rasa
```

```
# Create a new Rasa project
!rasa init --no-prompt

# Define the NLU model
nlu.md:
"""
## intent:greet
- hey
- hello
- hi
- good morning
- good evening

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

## intent:affirm
- yes
- indeed
- of course
- that sounds good

## intent:deny
- no
- never
- I don't think so
"""
```

```
# Define the stories
stories.md:
"""
## happy path
* greet
  - utter_greet
* affirm
  - utter_happy

## sad path
* greet
  - utter_greet
* deny
  - utter_sad
"""

# Define the domain
domain.yml:
"""
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:
"""

# Train the model
!rasa train

# Run the chatbot
!rasa shell
```

This code demonstrates creating a simple chatbot using the Rasa framework, defining intents, responses, and stories, and training the model.

### Additional Tips

- Use pre-trained language models like BERT, GPT-3, or Transformer-based models for more advanced chatbots.
- Implement fallback mechanisms to handle out-of-scope queries gracefully.
- Incorporate sentiment analysis to understand user emotions and tailor responses accordingly.
- Regularly monitor and update the chatbot to ensure it remains accurate and relevant.

## Sample Project Report

In [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')
```

In [2]:

```
df = pd.read_csv("/kaggle/input/training-dataset-for-chatbotsvirtual-assistants/20000-Utteranc
es-Training-dataset-for-chatbots-virtual-assistant-Bitext-sample/20000-Utterances-Training-dat
aset-for-chatbots-virtual-assistant-Bitext-sample/20000-Utterances-Training-dataset-for-chatbo
ts-virtual-assistant-Bitext-sample.csv")
df.head()
```

Out[2]:

| | flags | utterance | category | intent |
|---|---|---|---|---|
| 0 | BILC | I don't have an online account, what do I have... | ACCOUNT | create_account |
| 1 | BILQZ | can you tell me if i can regisger two accounts... | ACCOUNT | create_account |
| 2 | BPLC | I have no online account, open one, please | ACCOUNT | create_account |
| 3 | BIPLD | could you ask an agent how to open an account,... | ACCOUNT | create_account |
| 4 | BLQC | i want an online account, create one | ACCOUNT | create_account |

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

Out[3]:
27

## Without preprocessing

In [4]:
```
label_intent = preprocessing.LabelEncoder()
df['label_num'] = label_intent.fit_transform(df.intent)
df.head()
```

| | flags | utterance | category | intent | label_num |
|---|---|---|---|---|---|
| 0 | BILC | I don't have an online account, what do I have... | ACCOUNT | create_account | 10 |
| 1 | BILQZ | can you tell me if i can regisger two accounts... | ACCOUNT | create_account | 10 |
| 2 | BPLC | I have no online account, open one, please | ACCOUNT | create_account | 10 |
| 3 | BIPLD | could you ask an agent how to open an account,... | ACCOUNT | create_account | 10 |
| 4 | BLQC | i want an online account, create one | ACCOUNT | create_account | 10 |

ı [5]:
```
label_intent.__dict__
```

ut[5]:
```
{'classes_': array(['cancel_order', 'change_order', 'change_shipping_address',
       'check_cancellation_fee', 'check_invoices',
       '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)}
```

In [6]:
```
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)
```

```
Shape of X_train:  (17227,)
Shape of X_test:  (4307,)
```

In [7]:
```
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))
```

```
              precision    recall  f1-score   support
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 7 |
| 1 | 0.91 | 0.78 | 0.84 | 185 |
| 2 | 0.00 | 0.00 | 0.00 | 22 |
| 3 | 0.00 | 0.00 | 0.00 | 72 |
| 4 | 0.85 | 0.69 | 0.76 | 203 |
| 5 | 0.00 | 0.00 | 0.00 | 54 |
| 6 | 0.00 | 0.00 | 0.00 | 96 |
| 7 | 1.00 | 0.86 | 0.92 | 149 |
| 8 | 0.80 | 1.00 | 0.89 | 411 |
| 9 | 1.00 | 1.00 | 1.00 | 205 |
| 10 | 0.80 | 1.00 | 0.89 | 424 |
| 11 | 1.00 | 0.89 | 0.94 | 183 |
| 12 | 0.00 | 0.00 | 0.00 | 72 |
| 13 | 0.00 | 0.00 | 0.00 | 28 |
| 14 | 0.00 | 0.00 | 0.00 | 27 |
| 15 | 0.53 | 1.00 | 0.70 | 286 |
| 16 | 0.70 | 0.98 | 0.81 | 230 |
| 17 | 0.00 | 0.00 | 0.00 | 47 |
| 18 | 0.98 | 0.99 | 0.98 | 873 |
| 19 | 0.00 | 0.00 | 0.00 | 14 |
| 20 | 1.00 | 0.98 | 0.99 | 197 |
| 21 | 0.00 | 0.00 | 0.00 | 26 |

```
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 | 0.00 | 0.00 | 0.00 | 7 |
| 1 | 0.80 | 0.98 | 0.88 | 185 |
| 2 | 0.00 | 0.00 | 0.00 | 22 |
| 3 | 1.00 | 0.74 | 0.85 | 72 |
| 4 | 0.84 | 0.98 | 0.91 | 203 |
| 5 | 0.00 | 0.00 | 0.00 | 54 |