CRED: A Chatbot for Credit Card Management

CIS-600 Applied Natural Language Processing

Team Members

1. Venkata Satya Sri Ram Giduthuri (SUID: 972124521)
2. Pranathi Nallala (SUID: 949607909)
3. Kamal Preetam Chittuluri (SUID: 481767149)
4. Kavya Aithagoni (SUID: 550867610)
5. Komal Torlikonda (SUID: 341813954)

Abstract

This project report describes the creation of a chatbot Named Cred Chatbot that uses Natural Language Processing (NLP) to improve customer service for credit card users. This chatbot helps with important tasks like approving credit cards, making bill payments, giving exchange rate information, helping with money transfers, and answering customer questions. The chatbot uses advanced technology to understand and respond to customer inquiries accurately and quickly. Its main features include recognizing the purpose of customer requests, fixing typing errors, and providing responses that fit the situation. The chatbot can also handle voice commands, making it easier for more people to use. This report covers how we built and tested the chatbot and shows that it makes customer service faster and better, which is especially useful in the banking sector.

Introduction

In an era where digital customer service has become a benchmark for the banking industry, particularly in credit card services, the demand for automation and efficiency is paramount. Addressing this, we introduce "Cred Chatbot," an advanced tool powered by Natural Language Processing (NLP). Designed to navigate the complexities of customer interactions from card approvals to billing inquiries, this chatbot aims to streamline customer service by providing timely and precise responses.

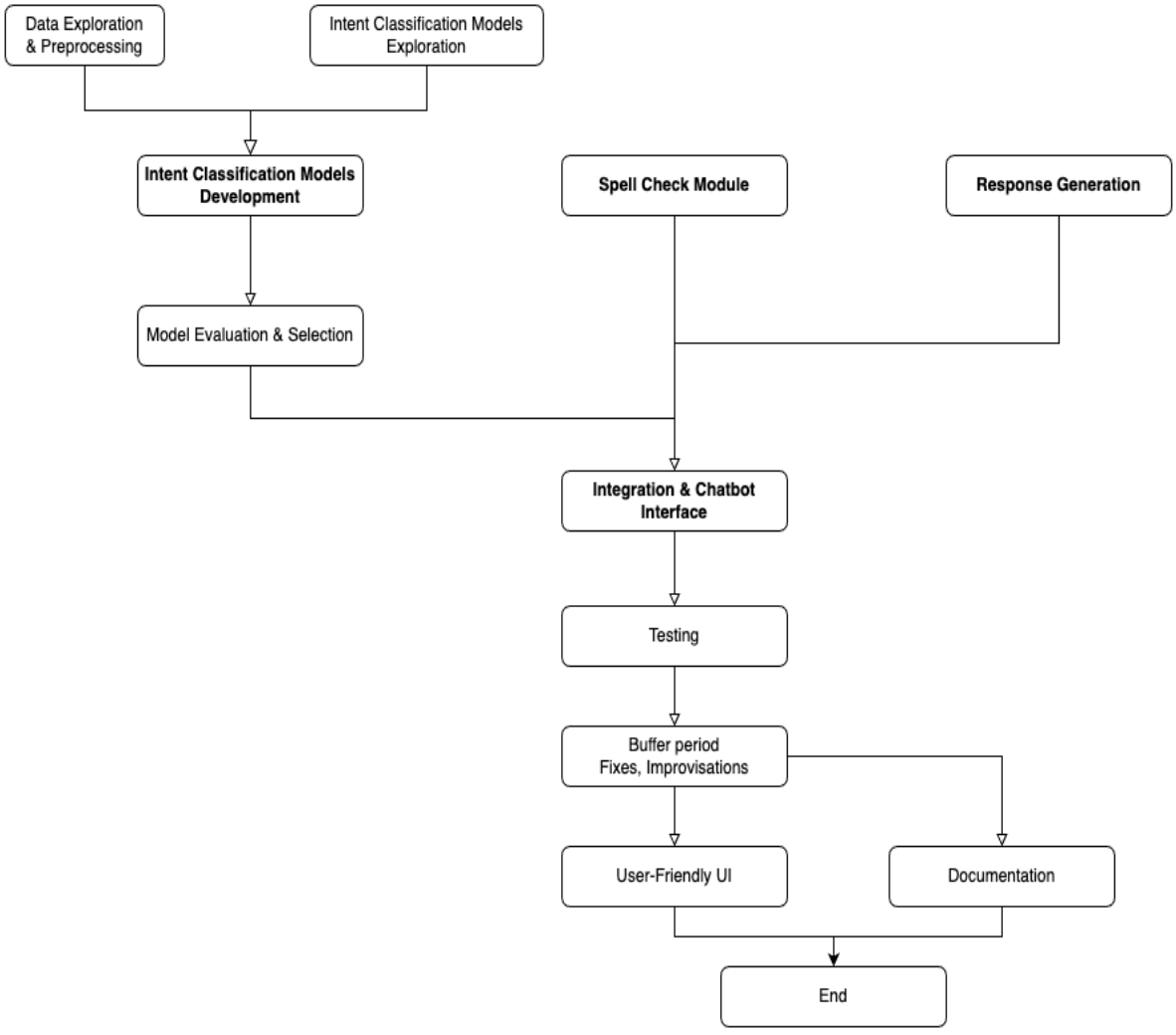
The significance of Cred Chatbot lies in its ability to synthesize the responsiveness of human customer service with the efficiency of automated systems. By integrating user-friendly features such as autocorrect and voice recognition, the chatbot extends service accessibility, reducing the pressure on service agents and enhancing the overall customer experience. This report will explore how Cred Chatbot represents a significant stride in the digital transformation of customer engagement within the credit card sector.

Motivation

The surge in digital banking has heightened the need for responsive and efficient customer service, especially in the credit card sector. Customers often experience long wait times and unsatisfactory service, impacting their overall satisfaction and the operational efficiency of financial institutions.

Our development of the Cred Chatbot is motivated by the opportunity to improve service delivery through Natural Language Processing (NLP). By automating responses to routine inquiries and transactions, the chatbot reduces wait times and enhances the accuracy of responses. It also frees up human agents to handle more complex issues, optimizing resource allocation and improving service quality. Additionally, the inclusion of voice commands makes the chatbot accessible to a broader range of users, ensuring inclusivity in digital banking advancements.

Architecture



The system architecture of the Cred Chatbot is strategically organized into a sequence of development and processing stage. The architecture can be detailed as follows:

- **Data Exploration & Preprocessing**

At the initial stage, the chatbot's architecture begins with data exploration and preprocessing. This step involves gathering, cleaning, and preparing the data necessary for training the NLP models. Here, raw data is transformed into a format that can be effectively utilized in machine learning.

- **Intent Classification Models Development**

The next layer in the architecture is dedicated to developing intent classification models. This involves training various algorithms to accurately determine user intents from their queries. This critical component enables our chatbot to understand and process customer requests effectively.

- **Model Evaluation & Selection**

Following the development phase, models undergo rigorous evaluation against predefined metrics. The best-performing model is then selected for integration, ensuring that the chatbot responds with the highest possible accuracy.

- **Spell Check Module**

In parallel to intent classification, a spell check module is integrated to correct any input errors. This module enhances the chatbot's understanding by ensuring queries are free from typos and grammatical inaccuracies before they are processed by the intent classifier.

- **Response Generation**

A response generation mechanism works alongside the spell check module. Once the user's intent is identified, this component generates appropriate responses, drawing from a file of predefined intent responses.

- **Integration & Chatbot Interface**

All components converge at the integration stage, where the chatbot's interface is designed. This is where the user interaction front-end is developed, incorporating both the dialogue management and the underlying NLP functionalities into a cohesive user interface.

- **Testing**

After integration, comprehensive testing is conducted to ensure the chatbot operates as intended, effectively understanding, and responding to user inputs.

- **Buffer Period**

  A buffer period allows for refinements based on feedback from testing. This stage is crucial for making iterative improvements that enhance functionality, usability, and reliability.

- **User-Friendly UI**

  The culmination of the development process is a user-friendly UI. The focus here is on creating an intuitive and seamless user experience that facilitates easy navigation and interaction with the chatbot.

- **Documentation**

  Concurrent with the development of the user interface, comprehensive documentation is created. This serves as a reference guide for users and developers, outlining the chatbot's capabilities, usage instructions, and technical details.

Technical Stack

The development of the Cred Chatbot was enabled by a combination of powerful programming languages and specialized libraries that facilitated the incorporation of Natural Language Processing (NLP) and machine learning technologies. This section outlines the key software components that were instrumental in building and optimizing our chatbot's capabilities.

***Programming Languages*:**

- **Python:** Python's versatility and the rich ecosystem of libraries made it the ideal choice for both preliminary data handling and backend logic implementation.

***Core Libraries*:**

- **Pandas (pd):** Essential for data manipulation and reading datasets, used across various modules for training models and organizing response sets.
- **NumPy (np):** Utilized primarily in the intent recognition phase for numerical operations, especially during data preprocessing and model predictions.
- **TensorFlow (tf):** This comprehensive framework supported the development of deep learning models in the intent recognition module, enabling the construction of a bidirectional LSTM network for accurate intent classification.
- **Keras:** Part of TensorFlow, used for building and training neural networks with high-level APIs, simplifying the process of model creation and iteration.
- **Natural Language Toolkit (nltk):** Applied in the autocorrect module to tokenize words and facilitate natural language processing tasks.

- **SpaCy:** Integrated for its powerful NLP capabilities in the autocorrect module, enhancing the context-aware spell-checking system.

- **Textdistance:** Employed for calculating edit distances to suggest the most probable corrections in the autocorrect functionality.

- **SparkNLP:** Used in autocorrect for advanced, context-aware spell checking, although noted for compatibility issues with certain processors.

- **Random:** This module was instrumental in the response generation phase, enabling random selection of responses to enhance the conversational nature of the chatbot.

***Specialized Libraries and Methods*:**

- **contextualSpellCheck**: Added to SpaCy's pipeline to implement contextual spell checking in the autocorrect system.

- **TextBlob**: Used in autocorrect for straightforward spelling correction, ensuring that responses are grammatically correct and contextually appropriate.

- **PySpark**: Facilitated large-scale data processing in the autocorrect system when working with SparkNLP.

***Voice and Text Processing*:**

- **SpeechRecognition:** This library was integral for converting spoken words into text, allowing our chatbot to process and respond to voice queries.

- **gTTS (Google Text-to-Speech):** gTTS enabled the chatbot to articulate responses aloud, offering an interactive experience for users who prefer or require auditory communication.

***User Interface Development*:**

- **React:** To build a dynamic and responsive user interface, we utilized React, a JavaScript library for creating user interfaces that can efficiently update and render components based on interaction data.

Data

Data collection

The Cred Chatbot was trained and tested using data from many platforms to provide a complete portrayal of credit card customer queries and interactions. The main techniques for collecting information were the following:

Kaggle Datasets: We used datasets that are freely available from Kaggle, a popular venue for machine learning and data science competitions. Customer Service interactions, customer inquiries, and previous credit card transaction records are all included in these datasets.

GitHub Repositories: Supplementary datasets related to credit card services have been obtained from GitHub repositories which offer open-source data. Additional datasets including user questions and responses have been made available by these repositories.

Crowdsourced Data: We gathered data using crowdsourcing platforms where people submitted credit card-related questions and interactions to enhance the training data with real-world events and various user intents.

Data Preprocessing

Intent classification model

Text Cleaning: It's possible that special characters, HTML tags, and non-alphanumeric characters were eliminated during preprocessing of the text data (train_data and test_data). The text is sure to be standardized and ready for more work after this cleaning.

Tokenization: The text data is tokenized using the Tokenizer from tensorflow.keras.preprocessing.text. Tokenization is a method of separating sentences into distinct words or tokens, which are required for further stages like embedding.

Padding Sequences: To make sure consistent length (max_length), token sequences (train_sequences and test_sequences) have been padded or reduced after tokenization. To produce the fixed-size input tensors required by the LSTM model, this step is needed.

Label Encoding: pd.Categorical(df_train['category']).codes are used to encode the target labels (train_labels and test_labels). In order to train the algorithm, category labels must be converted into numerical format in this stage

Auto-correct

Text Cleaning: One of the characteristics of all autocorrect techniques is text cleaning.
the goal of cleaning and preprocessing text data in order to ensure consistency and remove noise.

Model Initialization/Training: Applied autocorrect techniques (SparkNLPMethod) for model initialization and training with the intention of initializing or training particular NLP pipelines or models for spell check.

Tokenization: Used for all autocorrect techniques (Edit Distance Method, Textblob Method, Spacy ContextualSpellCheck Method, and SparkNLP Method). To remove extra processing by tokenizing incoming messages text into individual words or tokens.

Vocabulary Creation: Implemented in a large number of autocorrect techniques (Textblob, EditDistance, and Spacy ContextualSpellCheck) on the basis of using the tokenized words as a starting point, build a vocabulary gathering for grammar corrections.

Dialogue Response Generation

Loading Intent Responses: Predefined response mappings can be imported into dictionaries inside the ResponseGeneration class by importing them from external modules.

Random Response Generation: Based on the given input question or intent, will generate a random response.

Input Validation and Processing: It wil make sure the input question is handled correctly so that a response may be generated. Checks if the input_question matches to any already established response mappings by verifying it. To make it easier to compare the input_question with response mappings, normalize it (for example, by making it lowercase).

Response Retrieval and Handling: Depending on the input query or intent,get and handle the responses. Obtains responses from the responses dictionary in line with the input question, use dictionary lookups (get()). If there are more than one response set available, pick one at random (intent_responses_1 or intent_responses_2).

Voice Query Integration

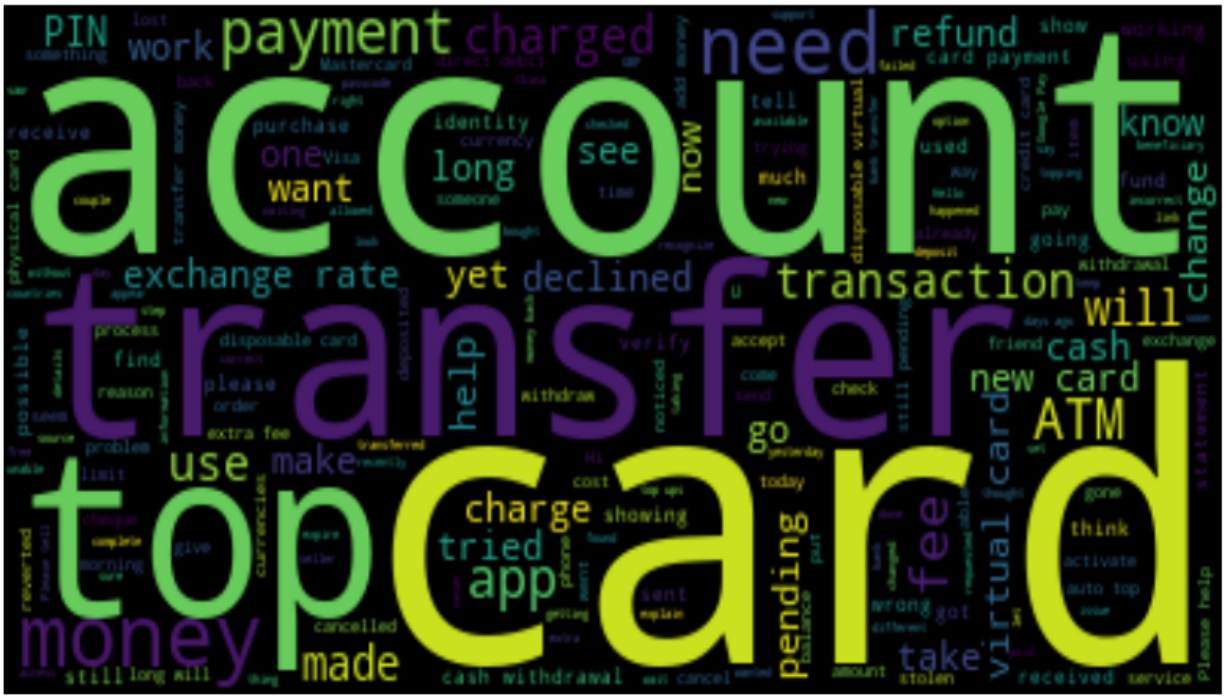
Noise Reduction and Adjustment: The recognizer's energy threshold changes with the adjust_for_ambient_noise() method in response to the noise level in the environment. This helps in decreasing the amount of noise from other sources in the audio input.

Audio Capture and Recognition: A microphone is used to record what is spoken (sr.Microphone()). For recording the audio input, use the listen() method. After that, Google's speech recognition service (r.recognize_google(audio2)) gets the recorded audio and uses it to convert the speech to text.

Text Normalization: For making consistency and processing easy, the recognized text (MyText) is converted to lowercase (MyText.lower()).

Audio Synthesis: gTTS (Google Text-to-Speech) is used to turn the response text (response_text) into speech.tts.save(audio_file) is used to save the synthesized audio as a WAV file.
Exploratory Data Analysis:

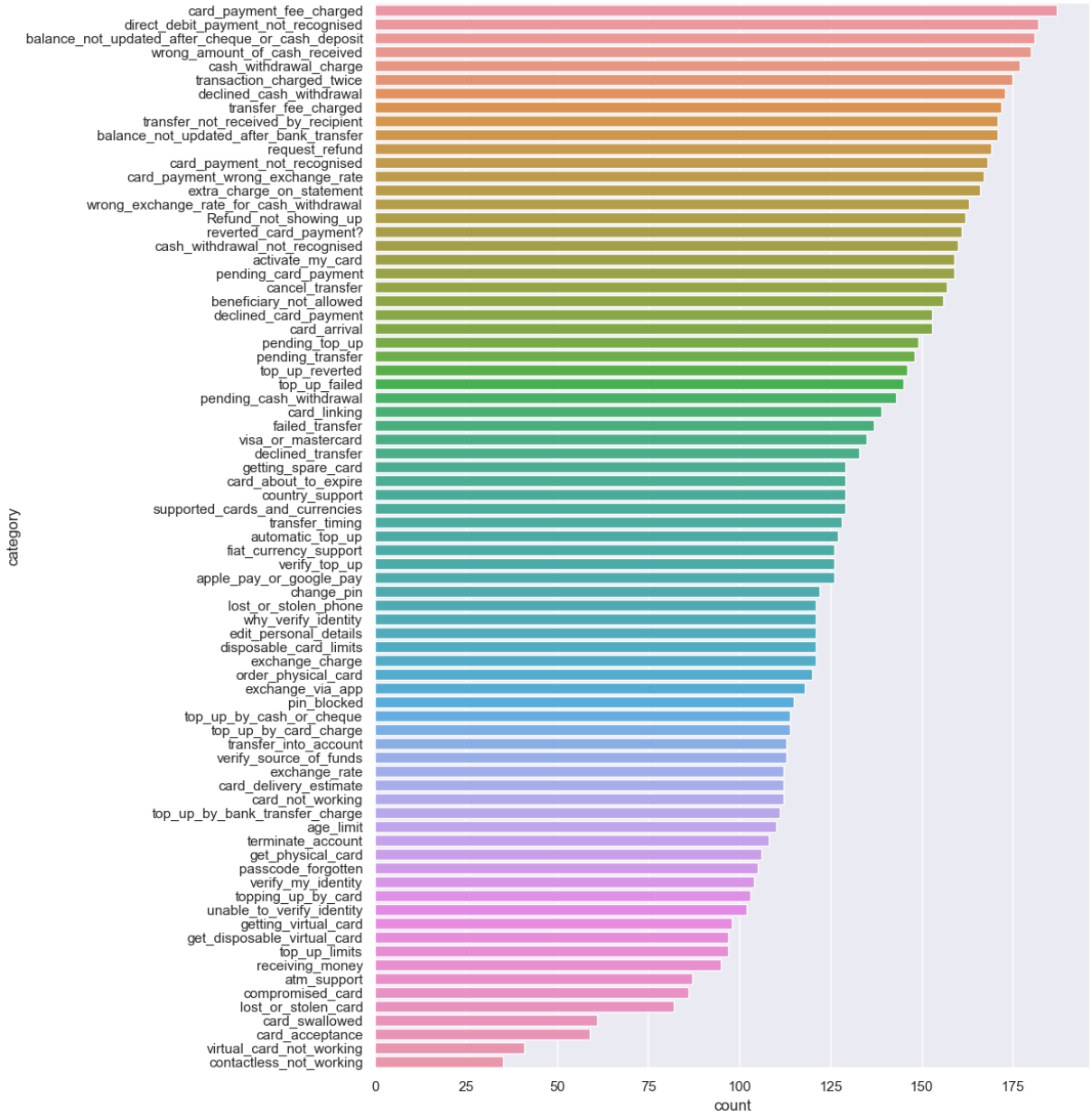
We utilized the wordcloud library to plot the top 200 frequent words, without considering the stopwords. This gives us an insight about the words that are being used by users in the data.



We also conducted an analysis to study the most common intents by the users. We made a plot for all the intents available.

Key Findings:

- The most common words are account, card, etc.
- By this, we can infer that most queries might be regarding the users cards, accounts, transfers, etc.
- The least common words include show, change, identify. This can mean that users might not be asking much about minor details such as show my balance, etc.



Key Findings:

- The most common type of inquiry is “card_payment_fee_charged”.
- By looking at the most common intents, we can understand that users ask most often regarding the hidden or extra charges on their credit card.
- The least common queries include “contactless_not_working”, “virtual_not_working”, implying most of the users have no problems with virtual and NFC payments.

Additional Observations:

The graph indicates a notable decline after the top three clarifications, even though it doesn't give exact frequencies. It looks like less people are asking questions about other problems, mobile app logins, and account features.

Approach: Models & Methodologies

The Cred Chatbot employs a variety of models and methodologies to automate and enhance interactions within the credit card service sector. This section outlines the key models and methodologies implemented across different aspects of the chatbot, from natural language understanding to response generation.

1. Autocorrect Models:

The autocorrect feature of the Cred Chatbot was meticulously crafted using various models, each tested for its effectiveness in correcting spelling errors and enhancing text input quality.

- **Edit Distance Method**
  Initially, the edit distance algorithm was utilized to identify potential corrections by measuring the number of edits required to transform one word into another. Despite its utility, it was not chosen as the final solution.

- **Contextual Spell Check**
  Using SpaCy integrated with **contextualSpellCheck**, this method provided context-aware corrections. Although powerful, it did not completely align with the project's requirements.

- **SparkNLP Spell Checking**
  Through SparkNLP, we explored a sophisticated spell-checking method that capitalized on contextual understanding at the sentence level. However, hardware compatibility issues prevented it from being the selected method.

- **TextBlob**
  TextBlob's correction mechanism is based on a probabilistic model that considers word frequency and contextual relevance, making it a valuable tool for handling common spelling mistakes in user queries. It is particularly user-friendly, making it a popular choice for projects requiring natural language processing capabilities.

After rigorous testing of the methods, **TextBlob** emerged as the preferred choice. It proved to be exceptionally well-suited for our needs due to its simplicity and the effectiveness of its probabilistic models for error correction.

2. Intent Recognition Model:

To ensure the Cred Chatbot can interpret and respond to user inquiries with high accuracy, various intent recognition models were explored and rigorously evaluated. The following is an overview of each model and the methodologies applied:

- **Long Short-Term Memory (LSTM):**
  LSTM networks are a type of recurrent neural network (RNN) capable of learning order dependence in sequence prediction problems. This model is particularly useful for processing and making predictions based on time series data or any data where the sequence is essential. Its memory cells can maintain information in memory for long periods, which is the key advantage in handling sequential data like text.

- **eXtreme Gradient Boosting (XGBoost) Classifier:**
  XGBoost is a decision-tree-based ensemble machine learning algorithm that uses a gradient boosting framework. Renowned for its performance and speed, XGBoost is a frequent winner in machine learning competitions due to its ability to handle a wide variety of data types and its robustness to outliers and non-linearities.

- **Random Forest Classifier:**
  Random Forest is an ensemble of decision trees, typically trained with the "bagging" method. This approach combines the predictions of several base estimators to improve generalizability and robustness over a single estimator. It works well with large datasets and can handle thousands of input variables without variable deletion.

- **Bidirectional LSTM:**
  This variant of LSTM has the capability to process data sequences in both forward and backward directions, providing it with a broader context for making predictions. This model is especially effective in understanding the context of language, which can be crucial for accurately determining the in**r**tent behind user inputs.

Model Optimization: Features like **EarlyStopping** and **ReduceLROnPlateau** callbacks are integrated to optimize training, preventing overfitting and enhancing the learning rate adjustments during model training.

3. **Response Generation Methodology:**
   - **Random Response Selection:** The chatbot uses a simple yet effective method to diversify its responses. Based on the identified intent, it randomly selects from predefined sets of responses, ensuring that the dialogue remains dynamic and less predictable.

   - **Modular Response Handling:** Responses are stored in separate modules and are fetched dynamically depending on the conversation context, allowing easy updates and scalability of response content.

4. **Data Handling and Model Training:**
   - **Data Sampling and Balancing:** The training process involves random sampling of data to ensure that the model is exposed to a balanced mix of examples, which helps in generalizing better across different types of user queries.

   - **Categorical Encoding:** Utilizes Pandas' **Categorical** type for encoding labels, which is essential for processing by the neural network.

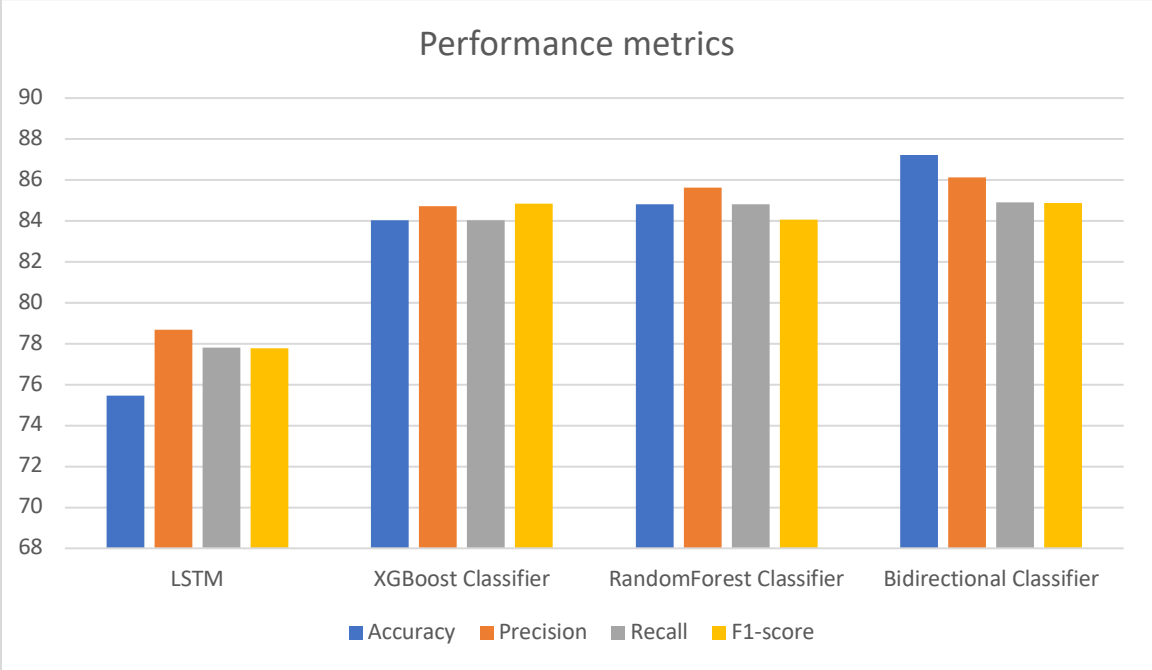
Evaluation Results & Analysis

In the development of the Cred Chatbot's intent recognition capabilities, we explored a variety of models to ensure that our system could accurately interpret and classify user queries. Here we detail the models employed, their performance metrics, and the methodologies underpinning their integration into our system.

- **Long Short-Term Memory (LSTM)**
  The LSTM model, known for its ability to capture long-term dependencies, achieved an accuracy of 75.47%. Its precision, recall, and F1-score were 78.69%, 77.82%, and 77.77% respectively, laying a solid foundation for understanding sequential data.

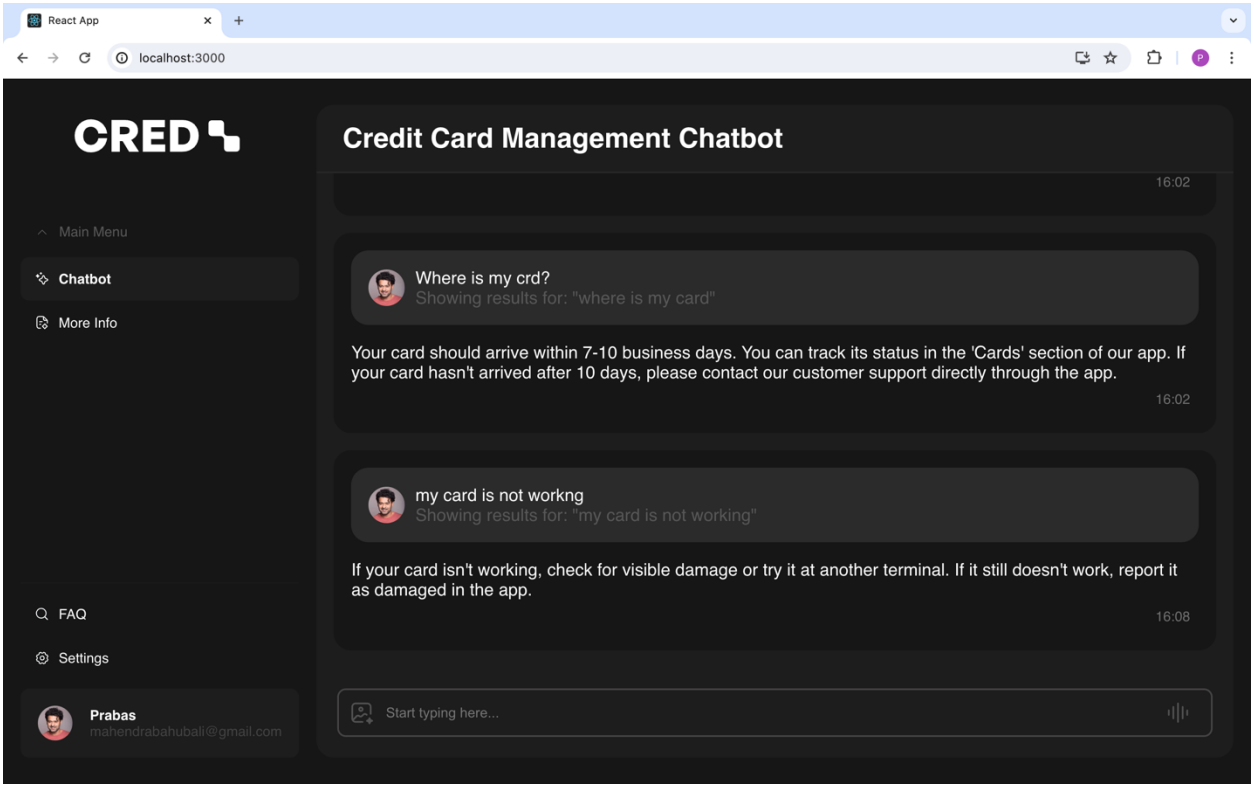
- **eXtreme Gradient Boosting (XGBoost) Classifier**
  This model is renowned for handling a variety of data types and was a strong performer in our tests, with an accuracy of 84.03%, precision of 84.71%, and an F1-score of 84.85%, highlighting its effectiveness in our predictive tasks.

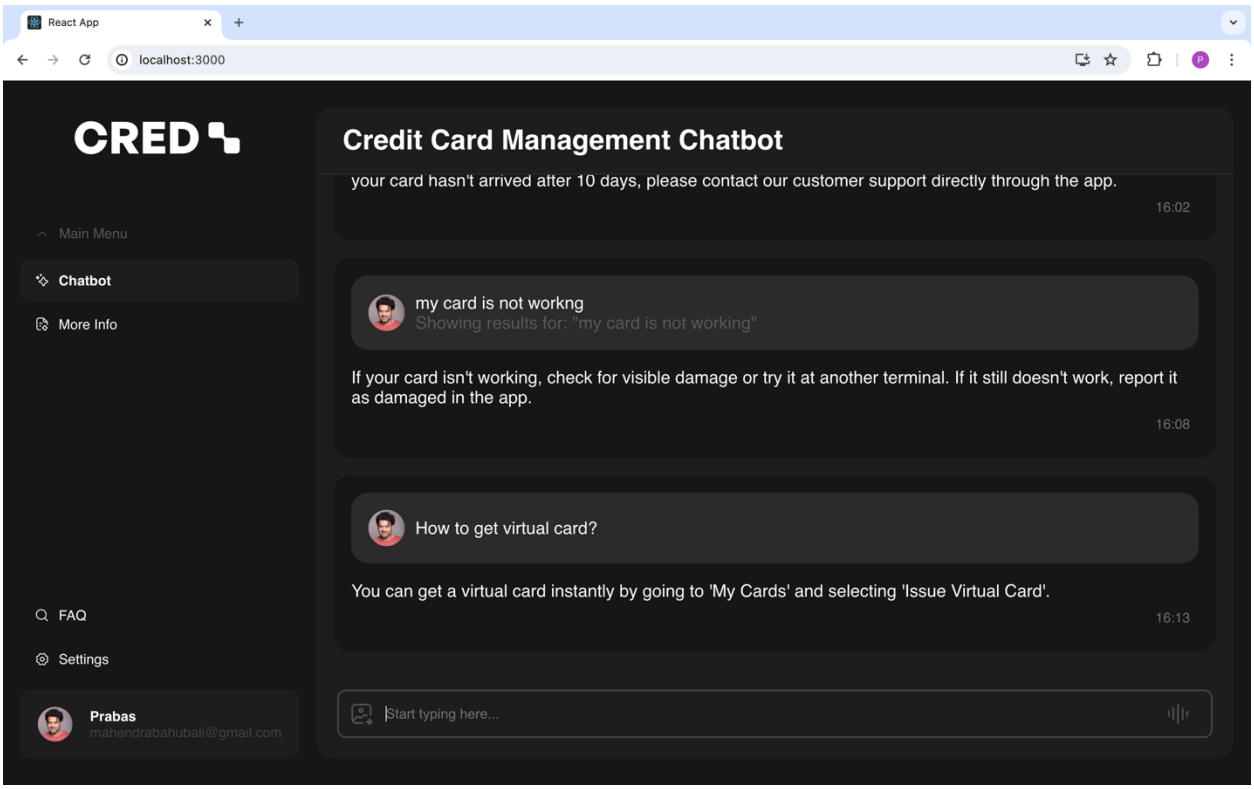
- **Random Forest Classifier**
  Leveraging the power of ensemble learning, this model yielded the highest precision among the classifiers at 85.63%, with an overall accuracy of 84.81% and an F1-score of 84.06%, demonstrating its robustness in classification tasks.
- **Bidirectional LSTM:** By processing data in both directions, this model topped our accuracy metric at 87.23%. It also demonstrated strong precision and recall with 86.12% and 84.92%, respectively, and an F1-score of 84.86%, making it the best performer in our suite.

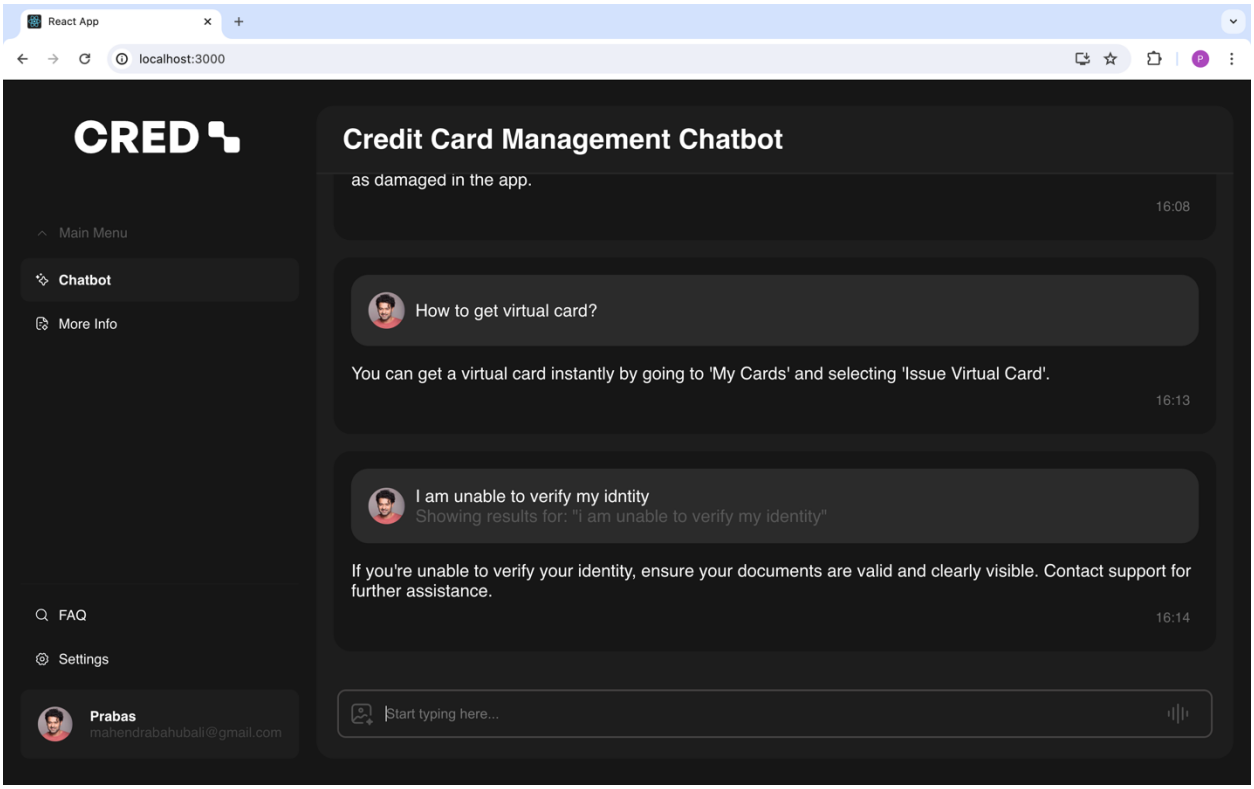


The **Bidirectional LSTM** was finalized for its superior ability to understand context by analyzing sequences in both past and future directions, resulting in more accurate intent recognition. It demonstrated higher accuracy and better performance on precision, recall, and F1-score metrics compared to other models. This model's proficiency in handling varied linguistic expressions made it especially suitable for the complex interactions within the chatbot. Ultimately, its alignment with the project's goal of delivering a nuanced and responsive user experience solidified its selection.

Chatbot In Conversation: User Interface Showcase




React App
localhost:3000
CRED
Credit Card Management Chatbot
your card hasn't arrived after 10 days, please contact our customer support directly through the app.
16:02
Main Menu
Chatbot
More Info
my card is not workng
Showing results for: "my card is not working"
If your card isn't working, check for visible damage or try it at another terminal. If it still doesn't work, report it as damaged in the app.
16:08
How to get virtual card?
You can get a virtual card instantly by going to 'My Cards' and selecting 'Issue Virtual Card'.
16:13
FAQ
Settings
Prabas
mahendrabahubali@gmail.com
Start typing here...


React App
localhost:3000
CRED
Credit Card Management Chatbot
as damaged in the app.
16:08
Main Menu
Chatbot
More Info
How to get virtual card?
You can get a virtual card instantly by going to 'My Cards' and selecting 'Issue Virtual Card'.
16:13
I am unable to verify my idntity
Showing results for: "i am unable to verify my identity"
If you're unable to verify your identity, ensure your documents are valid and clearly visible. Contact support for further assistance.
16:14
FAQ
Settings
Prabas
mahendrabahubali@gmail.com
Start typing here...

Speech Integration: ipynb Notebook Showcase

The function speech_to_text() converts our vocal speech query into English text and it is demonstrated through the screenshot below.

```
1  speech_text=speech_to_text()
2  speech_text
```

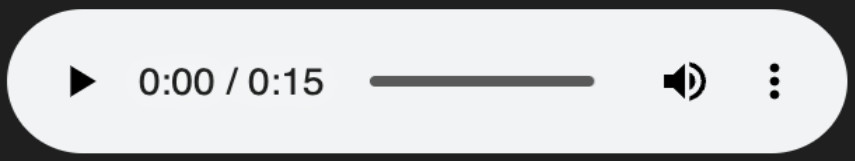
```
Please speak now...
You said: can you please find where my card is

'can you please find where my card is'
```

Now, we can convert the response generated from the application into a speech using text_to_speech() function as displayed below.

```
1  text_to_speech_response(res)
```

```
'response.wav'
```

```
1  ipd.display(ipd.Audio('response.wav', autoplay=True))
```



Conclusion

The development and deployment of the Cred Chatbot represent a significant achievement in leveraging Natural Language Processing to enhance the customer service experience within the credit card industry. Through meticulous model selection and rigorous testing, the chatbot has proven its capability to deliver accurate, context-aware interactions, exemplified by the successful integration of the Bidirectional LSTM for intent recognition. The incorporation of TextBlob as the chosen method for autocorrect further ensures that user inputs are understood with precision. Additionally, the chatbot's voice query integration broadens its accessibility, allowing users to engage in a manner that best suits their needs. Moving forward, the foundation laid by this project paves the way for continuous improvements and innovation in automated customer service, setting a new benchmark for future developments in the field.

Future Scope

As the Cred Chatbot evolves, enhancing its capabilities will focus on several key areas: improving natural language processing for better comprehension of complex and informal language, expanding multilingual support to cater to a diverse global audience, and personalizing user interactions based on individual preferences and history. Further development will also refine voice interaction features for more natural and conversational exchanges and enable proactive engagements like advice and reminders. Security measures will be fortified to protect user data as the chatbot handles more sensitive information. Additionally, integrating seamless transitions to human agents for complex or sensitive issues will ensure that users receive empathetic and expert support when needed, enhancing the overall user experience and maintaining the chatbot's relevance in the rapidly advancing technological landscape of customer service.

Github Repo

https://github.com/GvsSriRam/CIS-600-NLP-Project

Bibliography

[1]https://dl.acm.org/doi/full/10.1145/3547138?casa_token=bNPnMgyne8oAAAAA%3AwF2zORYon6IC8AxMfOYuasTZ8h-kQZQVX1sm04L241-k4xg1KbpZUdod0tHdk-ToTbHCWbMQpBT_
[2] https://link.springer.com/article/10.1007/s00500-020-05290-z
[3] https://www.sciencedirect.com/science/article/pii/S1877050918320374
[4]https://www.researchgate.net/publication/334176043_Bidirectional_LSTM_joint_model_for_intent_classification_and_named_entity_recognition_in_natural_language_understanding
[5] https://ieeexplore.ieee.org/document/10017592