PROJECT REPORT

Subject: Empowering Financial Accessibility:

Text-to-Speech Intent Detection for Visually Impaired Individuals in Banking Applications

Submitted to:

Indian Institute of Technology Guwahati

In Partial Fulfillment of the Requirement for the Internship

By:

Sandeepan Kalita

KIIT University, Bhubaneswar

Darshan Hazarika

CIT, Kokrajhar

Under the guidance of:

Dr. Jatindra Kumar Deka

CSE Department, IIT Guwahati

Date of Submission: July 10th, 2023

Internship Details:

University: Indian Institute of Technology Guwahati

Location: Guwahati, Assam, India

Acknowledgements

We would like to extend our sincere gratitude to the Indian Institute of Technology Guwahati for providing us with the opportunity to undertake this summer internship. The support and guidance we have received from the university have been instrumental in the successful completion of this project. We would like to express our heartfelt appreciation to our project supervisor, Dr. Jatindra Kumar Deka, for their invaluable insights, constructive criticism, and unwavering encouragement throughout this internship. Their expertise and mentorship have played a pivotal role in shaping our work and achieving the project objectives.

We would also like to acknowledge the Department of Computer Science and Engineering at IIT Guwahati for providing us with the necessary resources and infrastructure to carry out this project. The well-equipped library and research facilities have greatly facilitated our research process.

Lastly, we would like to thank the entire academic community at IIT Guwahati for fostering a conducive learning environment. The collaborative spirit and intellectual stimulation within the university have greatly contributed to our personal and professional growth.

We are truly grateful to IIT Guwahati and Dr. Jatindra Kumar Deka for their unwavering support and guidance, which have been instrumental in our development as aspiring researchers.

Sandeepan Kalita

Sandeepan Kalita

Darshan Hazarika

Darshan Hazarika

Abstract

Author: Sandeepan Kalita, Darshan Hazarika

Number of Pages: 31

Date: July 10, 2023

Degree: Bachelor of Technology

Internship Topic: Text-to-Speech Intent Detection for Visually Impaired Individuals in Banking Applications

Supervising Professor: Dr. Jatindra Kumar Deka

This report focuses on the implementation and training process of the BERT (Bidirectional Encoder Representations from Transformers) model for intent recognition in the banking domain. The BERT model utilizes transformers, which are encoder and decoder models that leverage the attention mechanism. Transformers offer advantages such as parallelization on GPUs/TPUs, the ability to process large amounts of data simultaneously, and the ability to process all tokens at once. The BERT model used in this project consists of 12 stacked transformer units.

The training data for the BERT model was collected from an open-source dataset available on the Hugging Face website. The dataset contains sentences commonly used in banks or financial institutions, along with their corresponding intents. Data preprocessing involved tokenizing the sentences into individual words and subwords and converting them into token IDs based on a vocabulary file. The tokens were then arranged sequentially with CLS (classification) and SEP (separator) tokens to mark the start and end of sequences. The input sequences were limited to a maximum length of 512 tokens, with longer sequences being split and shorter sequences being padded.

The BERT model was trained using the preprocessed data, and checkpoints and configuration files were loaded during model initialization. The model was trained to predict the intents based on the input sentences. Model validation was performed using a separate test set from the dataset, and accuracy was measured based on the predicted intents.

Finally, the trained BERT model was tested by feeding it with new sentences to predict their intents. The performance of the model was evaluated based on the accuracy and effectiveness of the intent recognition.

The implementation and training of the BERT model demonstrates its effectiveness in intent recognition tasks in the banking domain, showcasing the power of transformers and their ability to process and understand natural language.

Table of Contents

1.0 Introduction

In today's world, technological advancements have revolutionized the way we interact with information and perform daily tasks. One significant area of development is the application of voice recognition and natural language processing techniques to assist individuals with visual impairments. Such technologies aim to bridge the accessibility gap and provide blind individuals with a means to access information and interact with various services independently.
This report focuses on the development of a voice-based intent recognition system designed to assist blind individuals in navigating the complex process of applying for banking services. By leveraging machine learning and artificial intelligence techniques, this system aims to interpret users' spoken queries, determine their intent accurately, and provide appropriate responses or actions.

1.1 Background

The visually impaired community faces numerous challenges in accessing information and services that are often designed primarily for sighted individuals. Traditional graphical user interfaces and text-based systems pose significant barriers for blind individuals, limiting their ability to independently perform various tasks, including applying for banking services. These barriers can lead to dependence on others or a complete exclusion from critical services.
However, recent advancements in voice recognition, natural language processing, and machine learning technologies offer promising solutions to address these challenges. Voice-based systems enable blind individuals to interact with technology through spoken commands, providing them with greater independence and accessibility. By leveraging these technologies, we can develop applications tailored specifically to the needs of visually impaired individuals, facilitating their access to essential services.

1.2 Motivation

Intent classification using voice-to-text data has gained significant interest due to its potential in capturing rich information about an individual's intentions and desires. Voice, as a modality, provides valuable cues through speech patterns, intonations, and linguistic features, offering a complementary source of data for intent detection tasks.
The timely identification of intents can significantly benefit various applications, such as virtual assistants, customer support systems, and voice-controlled devices. Voice-based intent classification has numerous practical applications across various domains. In customer service, accurately understanding a customer's intent through their voice can enhance automated call routing, improve response times, and deliver personalized assistance. In virtual assistants and smart devices, accurately interpreting user intent from voice commands can enable seamless and intuitive control, creating a more user-friendly and efficient experience. Additionally, in healthcare and accessibility applications, voice-based intent classification can facilitate communication with individuals who have limited mobility or rely on voice input for interaction.By incorporating voice-to-text data into intent classification models, we can tap into the wealth of information conveyed through vocal cues. These cues can provide insights into emotional states, emphasis on specific words or phrases, and linguistic nuances that contribute to a more accurate interpretation of intent.

In summary, the motivation for intent classification using voice-to-text data lies in the desire to harness the rich contextual information carried by human speech. By leveraging voice data, we can improve the accuracy, personalization, and naturalness of intent classification systems across a range of applications. Voice-based intent classification holds the potential to transform human-machine communication, enabling more intuitive and effective interactions in various domains.

1.3 Objective

The primary objective of this project is to develop an application that utilizes voice recognition and natural language processing techniques to assist blind individuals in applying for banking services. The system will focus on accurately understanding users' spoken queries, extracting their intent, and providing appropriate responses or actions to meet their needs. By doing so, the application aims to empower blind individuals to navigate the complex process of applying for banking services independently and with confidence.

To achieve this objective, the project will involve training the system using a dataset comprising banking query data. This dataset will serve as the foundation for the machine learning algorithms, enabling them to learn the patterns and characteristics of different banking-related queries and intents. By training the system on a diverse set of queries, it will be able to recognize and classify a wide range of user intents accurately.

Additionally, the project will focus on optimizing the system for real-time performance, ensuring minimal latency between user queries and system responses. This aspect is crucial for providing a seamless and efficient user experience.
Overall, this project aims to leverage voice recognition and natural language processing technologies to empower blind individuals and enhance their accessibility to banking services. By enabling independent application processes, this system seeks to promote inclusivity and equal access to critical services for individuals with visual impairments.

2.0 Literature Review

1. The [5]**“Attention Is All You Need”** paper presents a groundbreaking advancement in the field of natural language processing (NLP) with the introduction of the Transformer network architecture. Unlike traditional models that heavily rely on recurrent or convolutional neural networks, the Transformer model is built exclusively on attention mechanisms. This innovative approach eliminates the need for sequential processing, making the model simpler, more parallelizable, and highly efficient.

Through extensive experiments on machine translation tasks, the authors demonstrate the superiority of the Transformer over existing models. The results reveal that the Transformer achieves state-of-the-art performance in terms of translation quality, surpassing previous models, including ensembles. The Transformer achieves a remarkable BLEU score of 28.4 on the English-to-German translation task, outperforming existing results by more than 2 BLEU. Similarly, on the English-to-French translation task, the Transformer establishes a new single-model state-of-the-art BLEU score of 41.8.

The success of the Transformer model extends beyond translation tasks. The authors also showcase its versatility and robustness by applying it to English constituency parsing, demonstrating its effectiveness even with limited training data. This highlights the generalization capabilities of the Transformer architecture across different NLP tasks.

The Transformer's advantages, including its superior performance, parallelizability, and reduced training time, have made it a groundbreaking contribution to the literature. Its simplified architecture opens up avenues for further research and applications in various NLP domains. Future directions may involve exploring the Transformer's application beyond translation, optimizing training efficiency, and further investigating its generalization capabilities.

In conclusion, the "Attention Is All You Need" paper presents the Transformer as a transformative model architecture in NLP. Its attention-based approach replaces traditional recurrent and convolutional layers, leading to superior results in machine translation tasks. The Transformer's innovative design and exceptional performance have established it as a pivotal milestone in the field of NLP research.

2. The [6]**"LIDSNet: A Lightweight on-device Intent Detection model using Deep Siamese Network"** paper addresses the need for a lightweight, fast, and accurate intent detection model for on-device deployment in resource-constrained environments. Intent detection is a crucial task in Natural Language Understanding (NLU) systems and serves as the foundation for task-oriented dialogue systems.

The proposed model, LIDSNet, leverages a Deep Siamese Network to accurately predict the intent of a message. It utilizes character-level features to enhance sentence-level representations and incorporates transfer learning by leveraging pre-trained embeddings. The effectiveness of the model's modules is evaluated through an ablation study, leading to the identification of the optimal architecture.

Experimental results demonstrate that LIDSNet achieves state-of-the-art accuracy on public datasets, with 98.00% accuracy on the SNIPS dataset and 95.97% accuracy on the ATIS dataset. Remarkably, the model achieves these results with just 0.59M parameters, highlighting its lightweight nature.

Additionally, LIDSNet is benchmarked against fine-tuned BERT models, specifically MobileBERT, on a Samsung Galaxy S20 device. The comparison reveals that LIDSNet is at least 41 times lighter and 30 times faster during inference. This showcases the model's efficiency and suitability for resource-constrained edge devices.

In conclusion, the "LIDSNet" paper presents a novel lightweight on-device intent detection model that combines a

Deep Siamese Network, character-level features, and transfer learning. The model achieves state-of-the-art accuracy on public datasets while maintaining an efficient and lightweight architecture. LIDSNet's performance, combined with its resource efficiency, positions it as a valuable solution for on-device intent detection in edge environments.

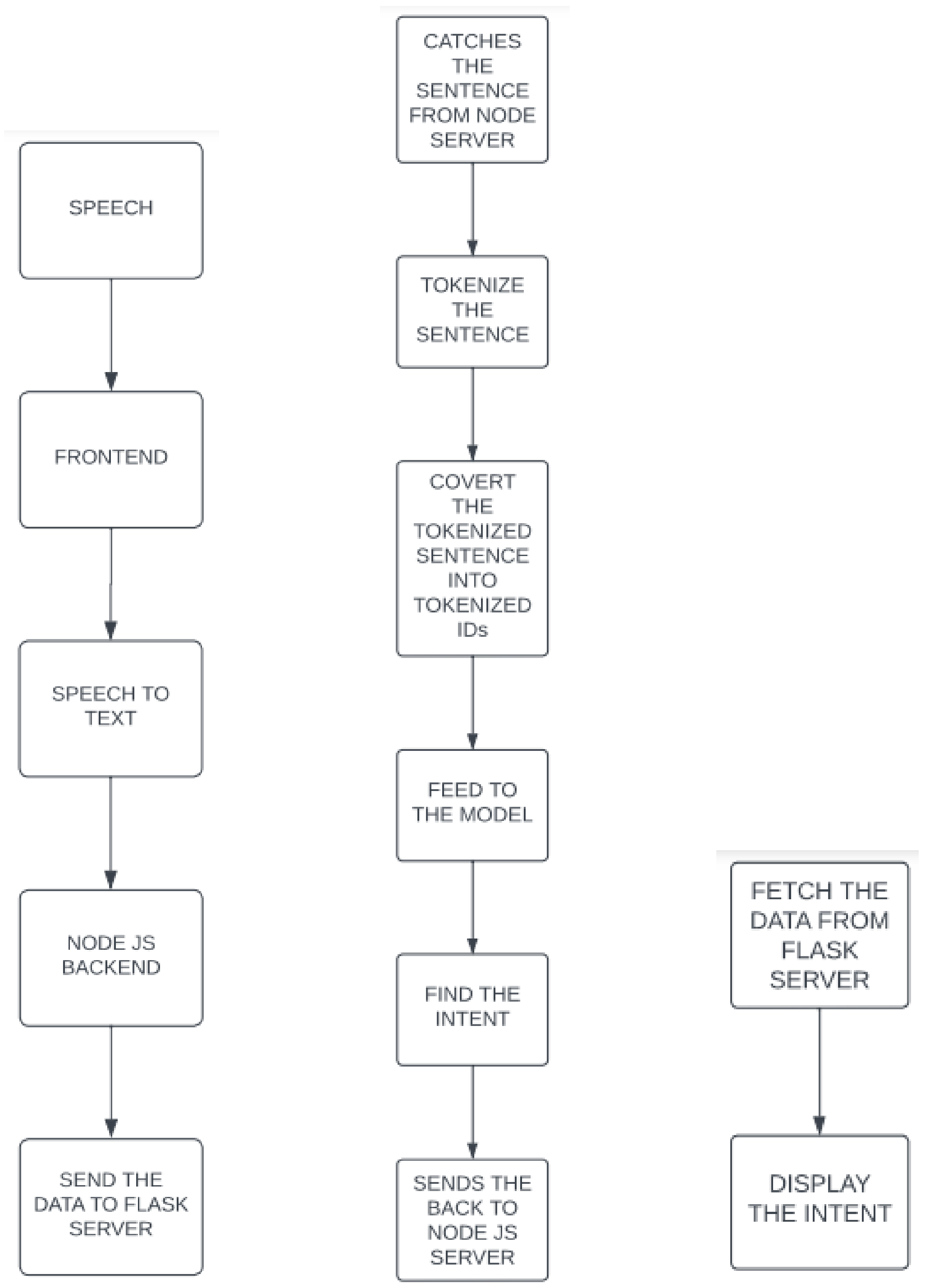
3. The [7]**“Zero-shot User Intent Detection via Capsule Neural Networks”** paper focuses on user intent detection, which plays a crucial role in question-answering and dialog systems. Traditionally, intent detection is treated as a classification problem where utterances are labeled with predefined intents. However, this approach is labor-intensive and time-consuming, as labeling utterances with diverse and emerging intents can be challenging. In response, the paper investigates the zero-shot intent detection problem, which aims to detect emerging intents for which no labeled utterances are currently available.

To address this challenge, the authors propose two capsule-based architectures: INTENT-CAPSNET and INTENT CAPSNET-ZSL. INTENT-CAPSNET extracts semantic features from utterances and aggregates them to discriminate existing intents. INTENT CAPSNET-ZSL builds upon INTENT-CAPSNET and incorporates zero-shot learning capabilities, enabling it to discriminate emerging intents by leveraging knowledge transfer from existing intents.

The proposed models are evaluated on two real-world datasets. The experimental results demonstrate that the models not only perform better in discriminating existing intents expressed in diverse ways but also have the ability to discriminate emerging intents even when no labeled utterances are available.

In conclusion, the "Zero-shot User Intent Detection via Capsule Neural Networks" paper introduces capsule-based architectures for zero-shot intent detection. These models offer advantages in discriminating both existing and emerging intents without the need for labeled utterances. The experimental results validate the effectiveness of the proposed models, highlighting their potential in addressing the challenges of user intent detection in real-world applications.

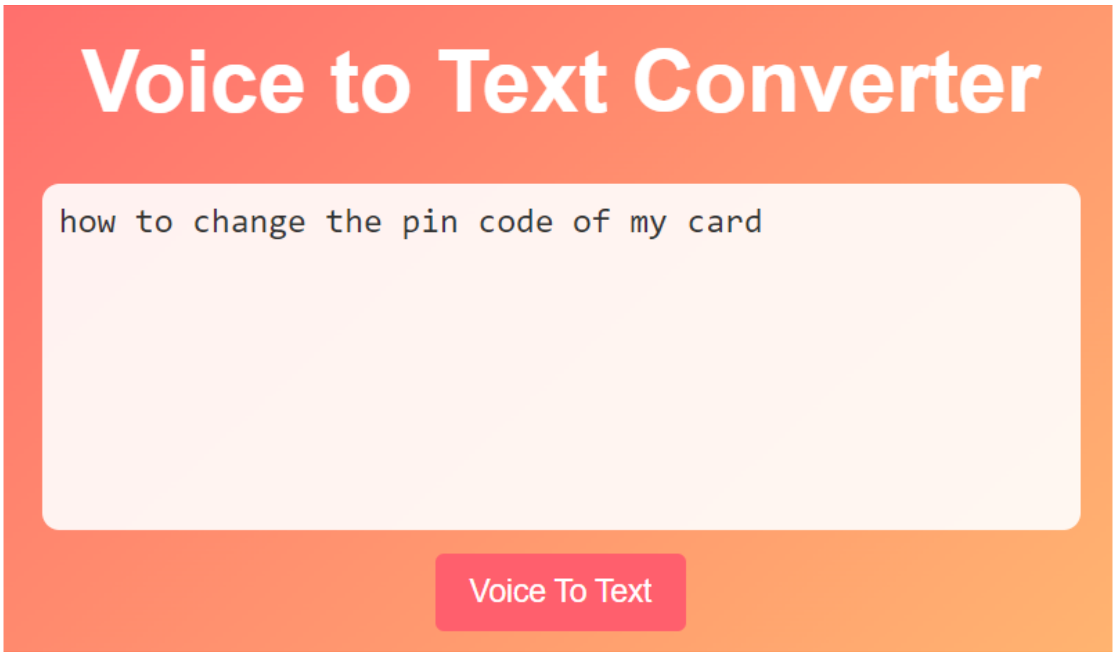
3.0 Flow Diagram


SPEECH
FRONTEND
SPEECH TO TEXT
NODE JS BACKEND
SEND THE DATA TO FLASK SERVER
CATCHES THE SENTENCE FROM NODE SERVER
TOKENIZE THE SENTENCE
COVERT THE TOKENIZED SENTENCE INTO TOKENIZED IDs
FEED TO THE MODEL
FIND THE INTENT
SENDS THE BACK TO NODE JS SERVER
FETCH THE DATA FROM FLASK SERVER
DISPLAY THE INTENT

NodeJs Server ➡️ Flask Server ➡️ NodeJs Server

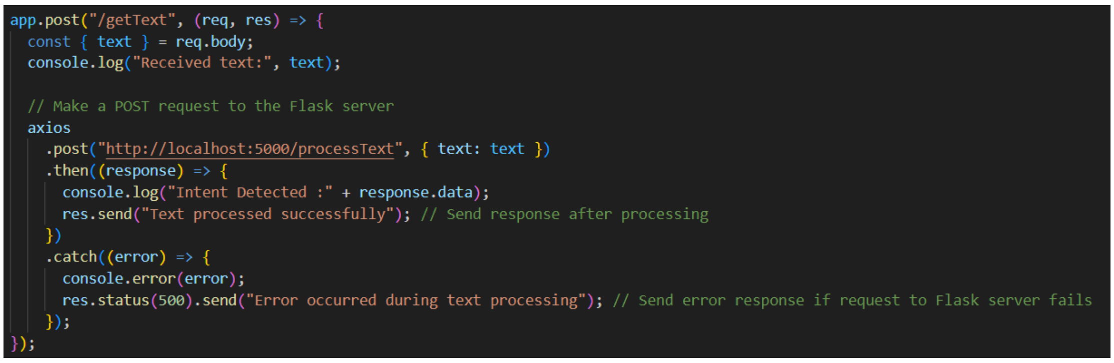
4.0 NODE.JS SERVER

The Node.js server serves as the central component in this application, responsible for handling both the frontend and backend functionalities. It acts as an intermediary between the user interface and the Flask server for intent determination.



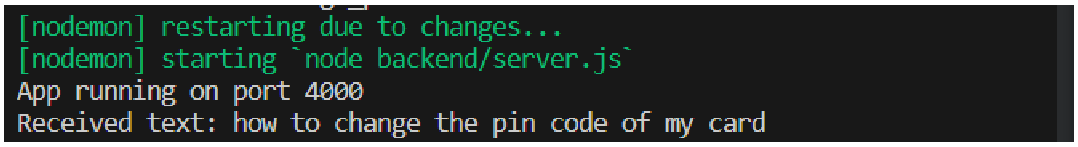
how to change the pin code of my card

At the frontend, the Node.js server provides a user-friendly interface where users can interact with the application. It serves HTML, CSS, and JavaScript files that are necessary for rendering the user interface in the browser. The server also hosts and serves static assets such as images, stylesheets, and client-side scripts.



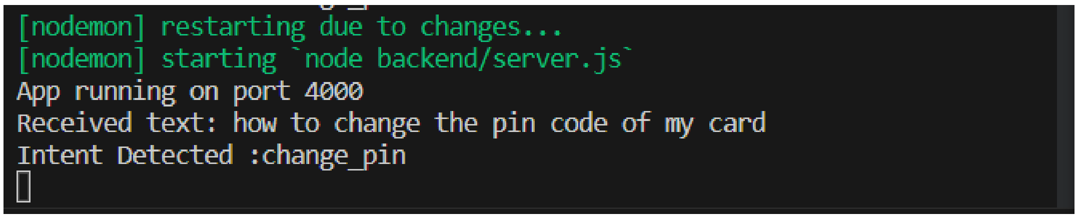
```
app.post("/getText", (req, res) => {
  const { text } = req.body;
  console.log("Received text:", text);

  // Make a POST request to the Flask server
  axios
    .post("http://localhost:5000/processText", { text: text })
    .then((response) => {
      console.log("Intent Detected :" + response.data);
      res.send("Text processed successfully"); // Send response after processing
    })
    .catch((error) => {
      console.error(error);
      res.status(500).send("Error occurred during text processing"); // Send error response if request to Flask server fails
    });
});
```

On the backend side, the Node.js server receives and processes API requests from the frontend. In the provided code snippet, the server listens for POST requests on the "/getText" endpoint. When a POST request is received, the server extracts the text from the request body. This text represents the user input or speech transcript that needs to be processed.



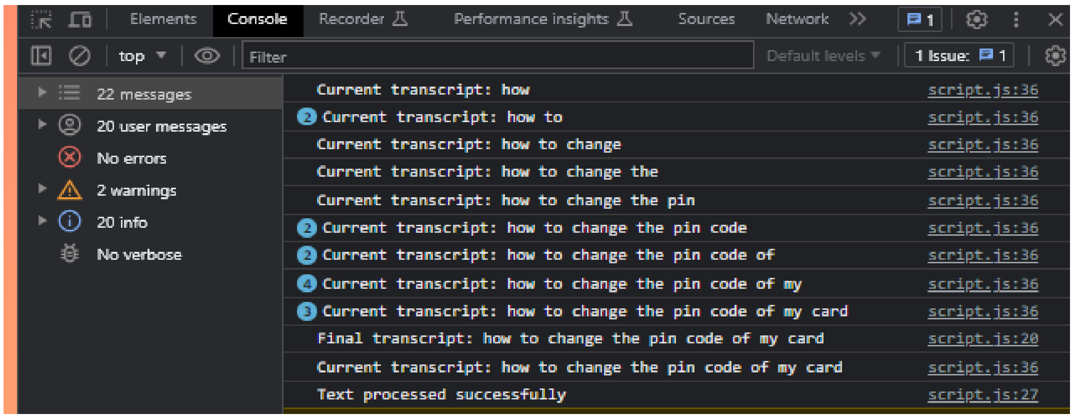
```
[nodemon] restarting due to changes...
[nodemon] starting `node backend/server.js`
App running on port 4000
Received text: how to change the pin code of my card
```

To determine the intent behind the user input, the Node.js server makes an API request to the Flask server. It utilizes the Axios library to send a POST request to the "/processText" endpoint of the Flask server. The extracted text is included in the request body as "text". This API request is asynchronous, allowing the Node.js server to continue processing other tasks while waiting for the response from the Flask server.
Once the response is received from the Flask server, the Node.js server logs the detected intent using the console.log function. The intent is retrieved from the response data. This logging serves as a means of monitoring and debugging the application, providing visibility into the intent recognition process.



```
[nodemon] restarting due to changes...
[nodemon] starting `node backend/server.js`
App running on port 4000
Received text: how to change the pin code of my card
Intent Detected :change_pin
```

Additionally, the Node.js server sends a response of "Text processed successfully" back to the client, indicating that the text has been successfully processed and the intent has been determined. This response is sent using the res.send function.

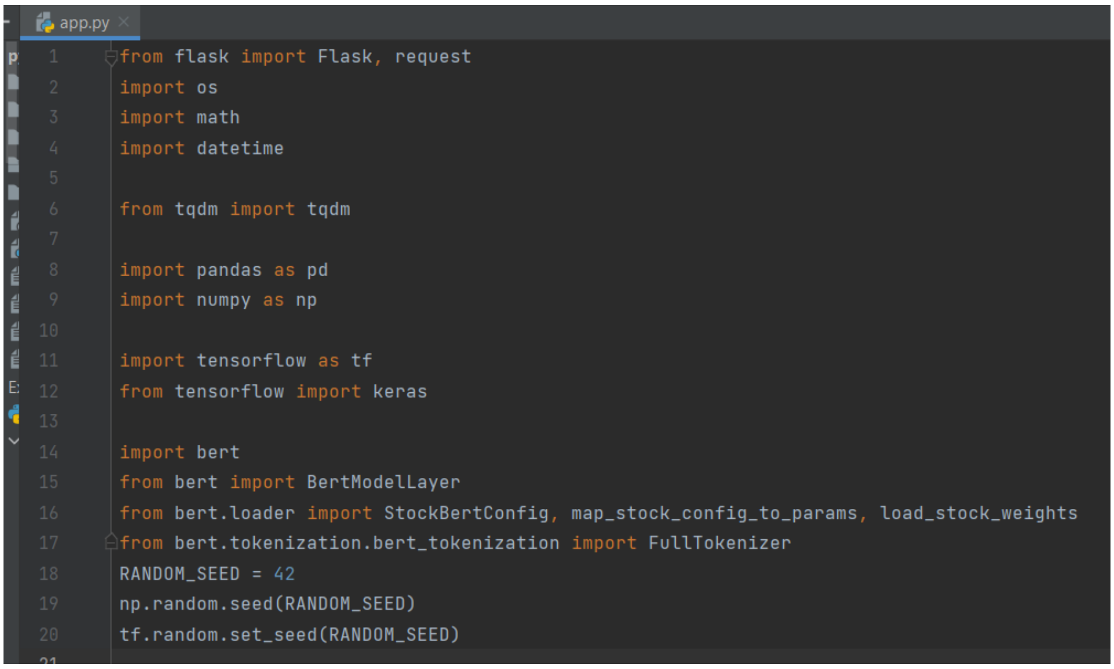


```
Current transcript: how                                      script.js:36
2 Current transcript: how to                                 script.js:36
Current transcript: how to change                            script.js:36
Current transcript: how to change the                        script.js:36
Current transcript: how to change the pin                    script.js:36
2 Current transcript: how to change the pin code             script.js:36
2 Current transcript: how to change the pin code of          script.js:36
4 Current transcript: how to change the pin code of my       script.js:36
3 Current transcript: how to change the pin code of my card  script.js:36
Final transcript: how to change the pin code of my card      script.js:20
Current transcript: how to change the pin code of my card    script.js:36
Text processed successfully                                  script.js:27
```

In the event of an error during the text processing or if the request to the Flask server fails, the Node.js server logs the error using console.error. It also sends an error response with a status code of 500, indicating that an error occurred during the text processing. This error response is sent using the res.status and res.send functions.
By handling the frontend and backend functionalities, the Node.js server enables seamless communication between the user interface and the Flask server. It facilitates the transmission of user input to the Flask server for intent determination and ensures a smooth user experience.

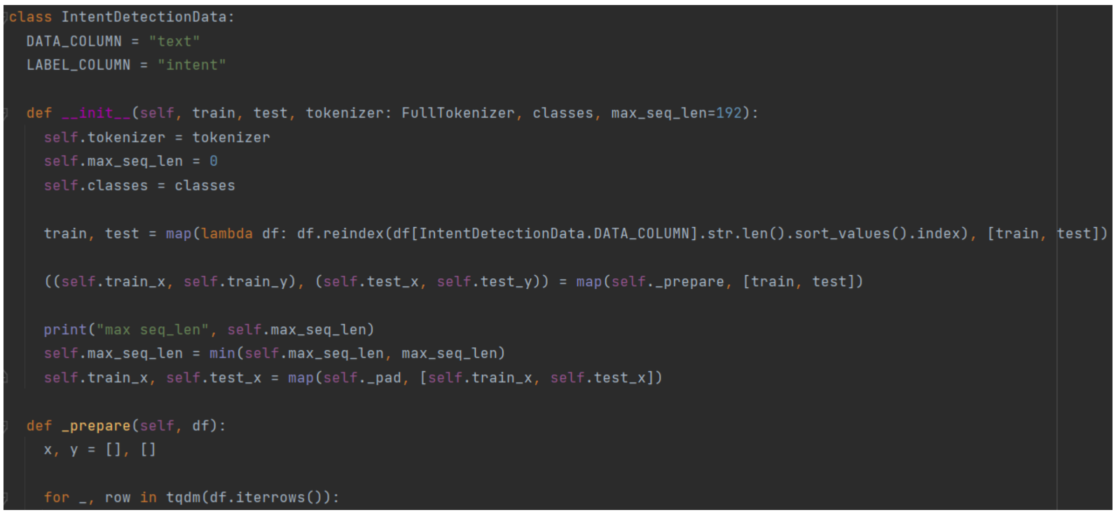
5.0 Flask Server:

The Flask server is a critical component in the application, responsible for receiving API requests from the Node.js server and determining the intent using a pre-trained model. Let's discuss the purpose and functionality of the Flask server based on the provided code snippets.
At the beginning of the Flask server code, necessary libraries and dependencies are imported, such as Flask, request, os, pandas, numpy, and tensorflow. These libraries enable various functionalities and data processing capabilities.

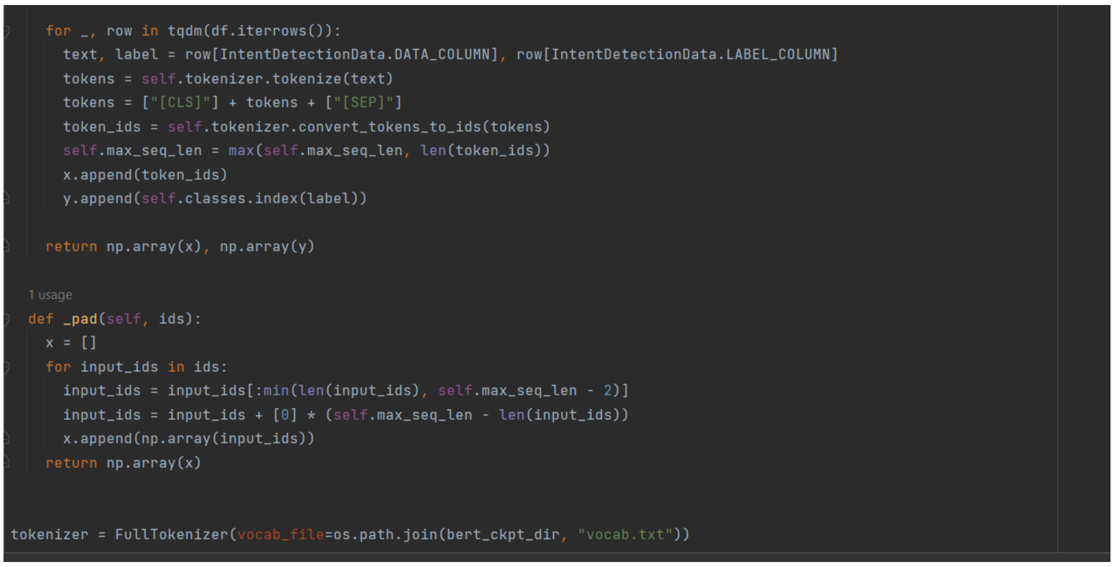

app.py

```python
from flask import Flask, request
import os
import math
import datetime

from tqdm import tqdm

import pandas as pd
import numpy as np

import tensorflow as tf
from tensorflow import keras

import bert
from bert import BertModelLayer
from bert.loader import StockBertConfig, map_stock_config_to_params, load_stock_weights
from bert.tokenization.bert_tokenization import FullTokenizer
RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
tf.random.set_seed(RANDOM_SEED)
```

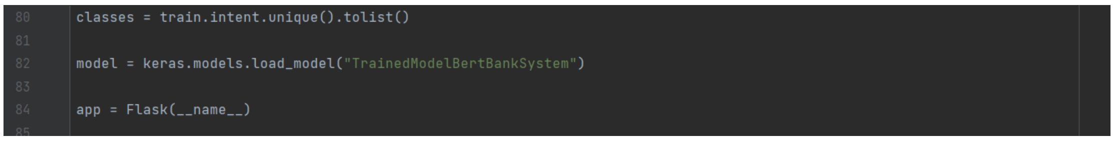
The Flask server is initialized using the Flask(__name__) command, creating an instance of the Flask application.



```python
class IntentDetectionData:
    DATA_COLUMN = "text"
    LABEL_COLUMN = "intent"

    def __init__(self, train, test, tokenizer: FullTokenizer, classes, max_seq_len=192):
        self.tokenizer = tokenizer
        self.max_seq_len = 0
        self.classes = classes

        train, test = map(lambda df: df.reindex(df[IntentDetectionData.DATA_COLUMN].str.len().sort_values().index), [train, test])

        ((self.train_x, self.train_y), (self.test_x, self.test_y)) = map(self._prepare, [train, test])

        print("max seq_len", self.max_seq_len)
        self.max_seq_len = min(self.max_seq_len, max_seq_len)
        self.train_x, self.test_x = map(self._pad, [self.train_x, self.test_x])

    def _prepare(self, df):
        x, y = [], []

        for _, row in tqdm(df.iterrows()):
```

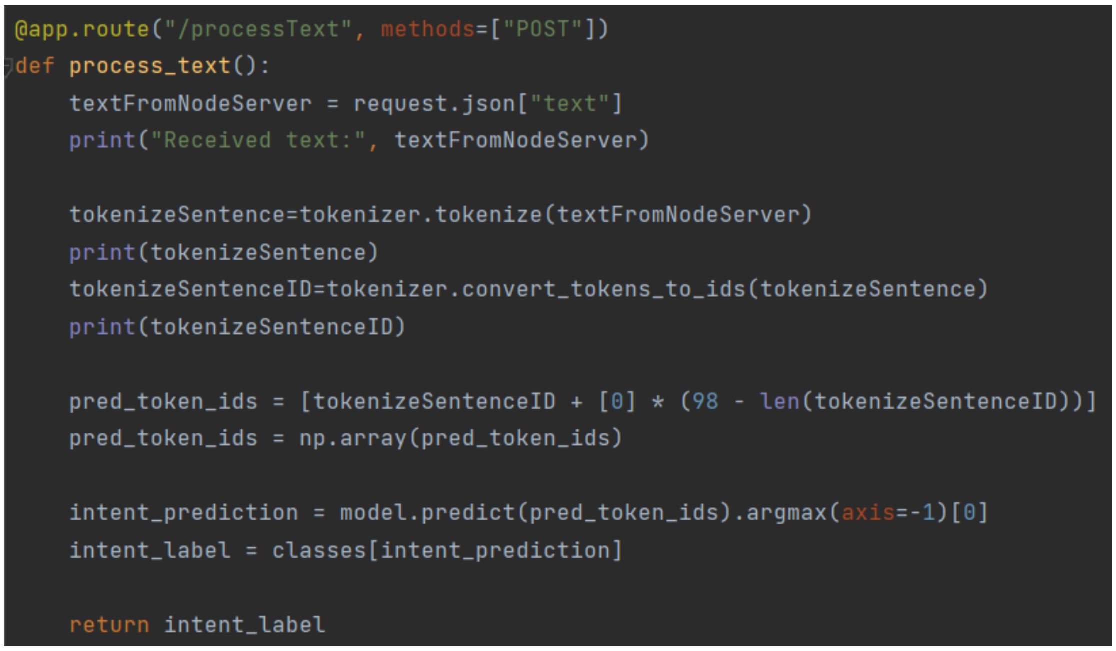


```
        for _, row in tqdm(df.iterrows()):
            text, label = row[IntentDetectionData.DATA_COLUMN], row[IntentDetectionData.LABEL_COLUMN]
            tokens = self.tokenizer.tokenize(text)
            tokens = ["[CLS]"] + tokens + ["[SEP]"]
            token_ids = self.tokenizer.convert_tokens_to_ids(tokens)
            self.max_seq_len = max(self.max_seq_len, len(token_ids))
            x.append(token_ids)
            y.append(self.classes.index(label))

        return np.array(x), np.array(y)

    1 usage
    def _pad(self, ids):
        x = []
        for input_ids in ids:
            input_ids = input_ids[:min(len(input_ids), self.max_seq_len - 2)]
            input_ids = input_ids + [0] * (self.max_seq_len - len(input_ids))
            x.append(np.array(input_ids))
        return np.array(x)


tokenizer = FullTokenizer(vocab_file=os.path.join(bert_ckpt_dir, "vocab.txt"))
```



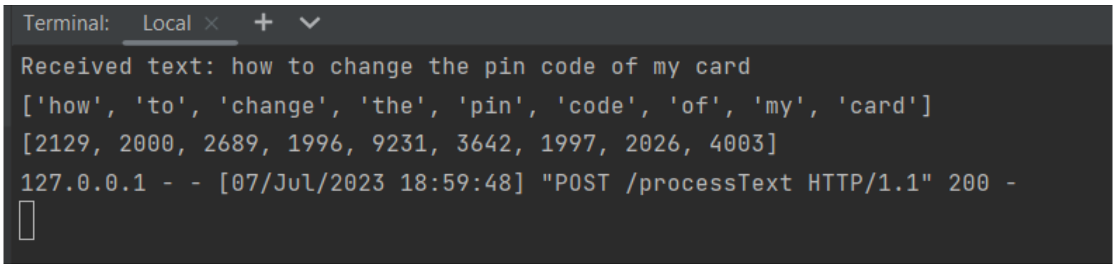
```
classes = train.intent.unique().tolist()

model = keras.models.load_model("TrainedModelBertBankSystem")

app = Flask(__name__)
```

Next, a class called IntentDetectionData is defined. This class handles the preparation and processing of the input data for intent detection. It takes the training and testing datasets, a tokenizer, the classes (intents), and a maximum sequence length as input parameters. Within this class, the text data is tokenized using the provided tokenizer and converted into token IDs. The maximum sequence length is determined based on the length of the tokens. The class also performs data reindexing and padding to ensure consistent sequence lengths. The prepared data is stored in the train_x, train_y, test_x, and test_y attributes.
The Flask server then initializes the tokenizer using the FullTokenizer class, specifying the path to the vocabulary file.
The classes (intents) are extracted from the training dataset and stored in the classes variable.
Next, the pre-trained intent detection model is loaded using keras.models.load_model.
The "TrainedModelBertBankSystem" represents the path to the saved model.



```python
@app.route("/processText", methods=["POST"])
def process_text():
    textFromNodeServer = request.json["text"]
    print("Received text:", textFromNodeServer)

    tokenizeSentence=tokenizer.tokenize(textFromNodeServer)
    print(tokenizeSentence)
    tokenizeSentenceID=tokenizer.convert_tokens_to_ids(tokenizeSentence)
    print(tokenizeSentenceID)

    pred_token_ids = [tokenizeSentenceID + [0] * (98 - len(tokenizeSentenceID))]
    pred_token_ids = np.array(pred_token_ids)

    intent_prediction = model.predict(pred_token_ids).argmax(axis=-1)[0]
    intent_label = classes[intent_prediction]

    return intent_label
```

The Flask server application is then defined using the @app.route decorator, specifying the endpoint "/processText" and the HTTP method "POST". This endpoint receives the API requests from the Node.js server, containing the text to be processed.
Within the "process_text" function, the received text is extracted from the JSON payload of the request using request.json["text"]. The text is then tokenized using the previously initialized tokenizer, resulting in a list of tokens.



```
Received text: how to change the pin code of my card
['how', 'to', 'change', 'the', 'pin', 'code', 'of', 'my', 'card']
[2129, 2000, 2689, 1996, 9231, 3642, 1997, 2026, 4003]
127.0.0.1 - - [07/Jul/2023 18:59:48] "POST /processText HTTP/1.1" 200 -
```

The tokenized sentence is further converted into token IDs using tokenizer.convert_tokens_to_ids. The token IDs are stored in the tokenizeSentenceID variable.

Subsequently, the token IDs are passed through the pre-trained model for intent prediction. The token IDs are reshaped to match the input shape expected by the model, and the model's predict method is called to obtain the intent prediction probabilities. The index with the highest probability is selected as the predicted intent, and the corresponding intent label is retrieved from the classes list.

The predicted intent label is then returned as the response from the Flask server.

Finally, the Flask server is run using app.run(), starting the server on the local machine.

In summary, the Flask server receives API requests from the Node.js server, tokenizing the input text, converts it to token IDs, and passes it through a pre-trained model for intent prediction. The server responds with the predicted intent label. This communication between the Node.js and Flask servers enables efficient intent determination and response generation in the application.

6.0 Communication between Servers:

The Node.js server communicates with the Flask server using HTTP-based API requests and responses. This allows seamless transmission of data between the two servers. The Axios library is utilized for making API requests from the Node.js server to the Flask server.

To send a request to the Flask server, the Node.js server uses the axios.post method. It specifies the URL of the Flask server's endpoint, which is "http://localhost:5000/processText" in this case. The request is sent as a POST request, and the data to be sent is provided as the request body. In this scenario, the text extracted from the user input is sent as the request body in JSON format. The axios.post method automatically serializes the data into JSON.
Once the request is made, the Node.js server waits for the response from the Flask server. The .then method is used to handle the successful response, while the .catch method is used to handle any errors that may occur during the request. In case of a successful response, the .then callback function is executed, where the response data is accessed and logged

using console.log. In this specific case, the response data contains the detected intent from the Flask server.
The communication between the Node.js server and the Flask server predominantly relies on the HTTP protocol. This protocol defines the structure and rules for data transmission over the internet. The request and response data are typically encoded using JSON (JavaScript Object Notation), which is a lightweight data interchange format. JSON provides a readable and standardized way to represent data, making it ideal for transferring information between the Node.js and Flask servers.

7.0 Displaying Intent in Console:

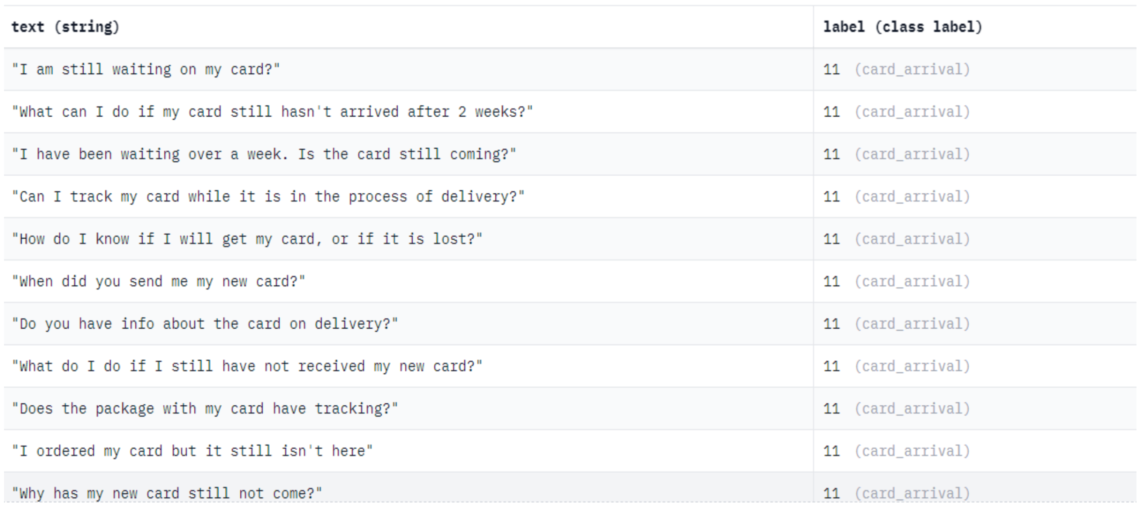
To display the intent in the console, the Node.js server utilizes the console.log function. After receiving the response from the Flask server, the .then callback function is executed. Within this function, the detected intent is accessed from the response data. The intent value is then concatenated with the string "Intent Detected :" using the + operator.

By logging this concatenated string to the console using console.log, the Node.js server effectively displays the detected intent in the console. This enables developers and administrators to monitor the application's performance, evaluate the accuracy of the intent recognition, and gain insights into the processing of user input.
The console.log statement can be found in the code snippet within the .then callback function, where it logs the concatenated string "Intent Detected :" along with the response data. This enables the server to receive the response from the Flask server and output it to the console for further analysis and debugging.

By leveraging the console.log function, the Node.js server provides a convenient means of displaying the intent information, aiding in the development, testing, and monitoring of the application.

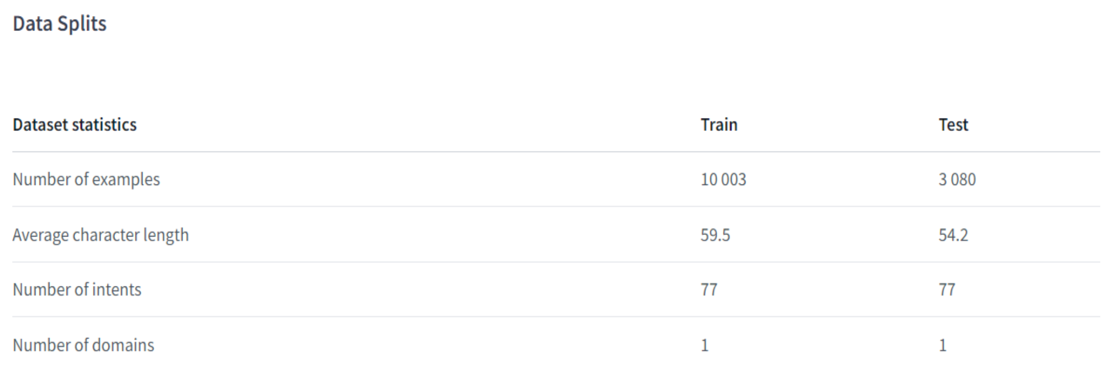
8.0 Dataset Detail

Dataset composed of online banking queries annotated with their corresponding intents. Dataset provides a very fine-grained set of intents in a banking domain. It comprises 13,083 customer service queries labeled with 77 intents. It focuses on fine-grained single-domain intent detection.



text (string)	label (class label)
"I am still waiting on my card?"	11 (card_arrival)
"What can I do if my card still hasn't arrived after 2 weeks?"	11 (card_arrival)
"I have been waiting over a week. Is the card still coming?"	11 (card_arrival)
"Can I track my card while it is in the process of delivery?"	11 (card_arrival)
"How do I know if I will get my card, or if it is lost?"	11 (card_arrival)
"When did you send me my new card?"	11 (card_arrival)
"Do you have info about the card on delivery?"	11 (card_arrival)
"What do I do if I still have not received my new card?"	11 (card_arrival)
"Does the package with my card have tracking?"	11 (card_arrival)
"I ordered my card but it still isn't here"	11 (card_arrival)
"Why has my new card still not come?"	11 (card_arrival)

The dataset is splitted in the following manner



Data Splits

Dataset statistics	Train	Test
Number of examples	10 003	3 080
Average character length	59.5	54.2
Number of intents	77	77
Number of domains	1	1

Dataset creation

Dataset composed of online banking queries annotated with their corresponding intents.

BANKING77 dataset provides a very fine-grained set of intents in a banking domain. It comprises 13,083 customer service queries labeled with 77 intents. It focuses on fine-grained single-domain intent detection.

Dataset Structure

```
{

'label': 11, # integer label corresponding to "card_arrival" intent

'text': 'I am still waiting on my card?'

}
```

Data Fields

text: a string feature.

label: One of classification labels corresponding to unique intents.

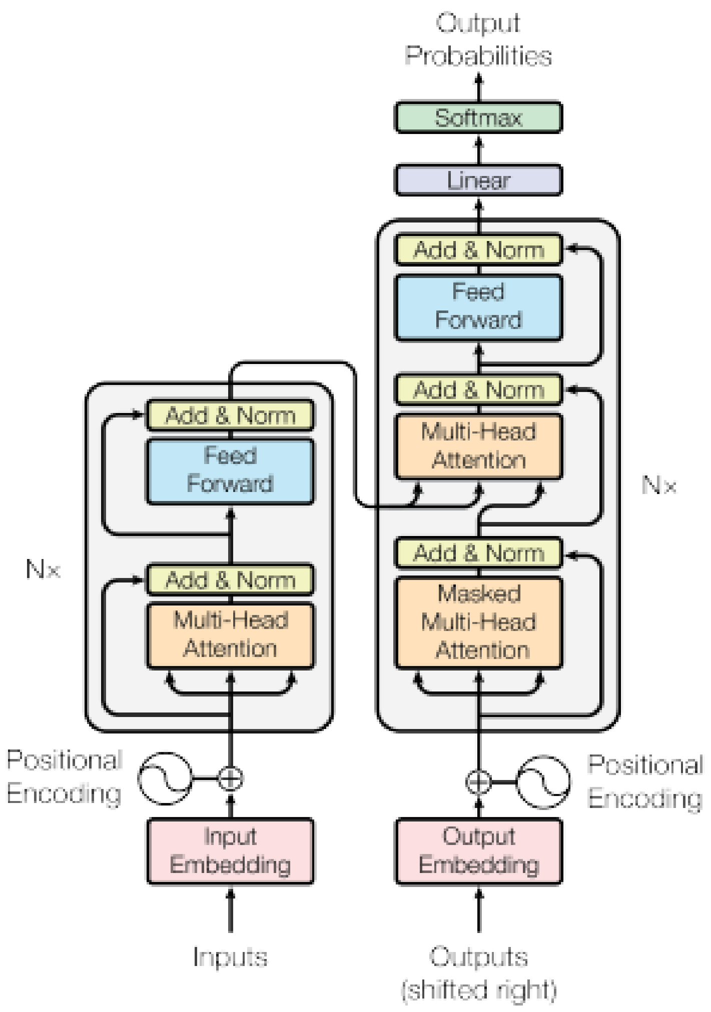
9.0 Bert Model

BERT (Bidirectional Encoder Representations from Transformers) is a powerful language representation model that has made significant advancements in the field of natural language processing (NLP). It was introduced by Google researchers in 2018 and has since become a widely adopted model due to its impressive performance in various NLP tasks. The key innovation of BERT lies in its bidirectional training approach. Unlike previous models that processed text in a unidirectional manner, BERT is designed to capture the contextual information from both the left and right contexts of a word or token. By training on large amounts of unlabeled text data using a masked language modeling objective, BERT learns to predict missing words in a sentence by considering the surrounding context. This bidirectional training enables BERT to capture deep contextual relationships and produce highly effective contextualized word embeddings.

The BERT uncased model has 12 layers, 768 hidden size and 12 attention heads. Uncased means that the text used during training and inference is converted to lowercase, treating all uppercase and lowercase letters as the same. The 12 layer of the BERT model signifies that it has 12 transformer blocks and each transformer block consists of encoders and decoders connected together. Each encoder block consists of a multihead attention unit and a feed forward neural network unit. Initially all the text is inserted to the encoder system where in the tokenized sequences are passed through these layers and then are passed to the decoder. The Decoder similarly has two multi head attention units and a feed forward neural network unit, Initially the target text sequences are preprocessed and feeded to the decoder similar to the encoder . The tokenized sequences are passed through a layer of multihead attention unit layer. The output of this layer

along with the output of the encoder layer is passed through another layer multi head attention unit which maps the text accordingly depending upon the task the model is performing. The results are again passed through a final layer of feed forward unit and a linear layer which predicts the probabilities generated by the model. These probabilities are then passed through a softmax activation function to give us the final result.
The H-768 denotes the hidden size or embedding dimensionality of the BERT model, which is 768. In other words, each token or word in the input sequence is represented by a 768-dimensional vector that includes information about its semantics and context.
The A-12 denotes that there are 12 attention heads in the BERT model. With the use of attention heads, the model is able to focus on numerous portions of the input sequence at once, concurrently collecting a variety of relationships and patterns.

The BERT uncased_L-12_H-768_A-12 model is pre-trained utilizing a masked language modeling goal and next sentence prediction task on a large corpus of text data, including Wikipedia and BookCorpus. The model can acquire general language comprehension and recognise word and phrase contexts thanks to this pre-training. The BERT model can be pre-trained and then fine-tuned on particular downstream tasks, such as Intent Detection, sentiment analysis, named entity recognition, or question answering, by training it on data that has been labeled for those purposes. By fine-tuning, the pre-trained BERT model adjusts to the particular task, allowing it to take use of its language understanding abilities and capture task-specific patterns.



[5]The Transformer - model architecture

The advantages of BERT in NLP tasks include:

Contextual Word Representations: BERT generates word embeddings that capture the contextual information of each word, taking into account its surrounding words. This enables better understanding of word meaning and enhances the performance of downstream NLP tasks.

Relevance of BERT in Intent Classification:

Intent classification is a fundamental task in natural language understanding, where the goal is to determine the underlying

intent or purpose of a user's input. BERT's contextual embeddings and bidirectional training make it highly relevant and effective in intent classification tasks.
By leveraging BERT, intent classification models can better understand the nuances of user queries and capture the contextual information that influences the intent. This enables more accurate classification of user intents, even for queries with complex or ambiguous phrasing. BERT's ability to consider both preceding and following context helps in capturing dependencies between words and improves the overall performance of intent classification systems.
In the context of the voice-based intent recognition system for banking applications, incorporating BERT can enhance the accuracy and reliability of intent classification. By utilizing BERT's contextual embeddings, the system can better distinguish between different banking-related intents and provide more precise and tailored responses to user queries. This improves the user experience and enables blind individuals to interact with the system more effectively when applying for banking services.

9.1 Transformers in BERT model

A transformer is an encoder and decoder model that uses the attention mechanism. It has massive advantage over RNN based encoder and decoder architecture since it allows us to:
Take advantage to parallelization GPU/TPU
Process much more data in the same amount of time
Process all tokens all at once
Our BERT(pre-trained) model has 12 transformer units stacked accordingly.

9.2 Pre-Training of BERT model

BERT (Bidirectional Encoder Representations from Transformers) is pretrained using two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

In the MLM task, 15% of the tokens in each sequence are randomly masked, and the model is trained to predict these masked tokens using the surrounding context. This approach allows BERT to learn the contextual information and relationships between words. To adapt the model for fine-tuning, 80% of the time, the masked tokens are actually replaced with the [MASK] token, 10% of the time, they are replaced with a random token, and 10% of the time, they are kept as is.

In the NSP task, BERT is trained to determine whether a second sentence follows the first sentence in a given corpus. This binary classification task helps the model understand the relationship and coherence between two sentences. 50% of the time, the second sentence is indeed the next sentence in the corpus, while the other 50% of the time, a random sentence is chosen from the corpus.

Fine-tuning BERT involves using the pretrained model on specific downstream tasks. For sequence classification, the model's output is pooled using the [CLS] token, which represents the classification for the entire sequence. This pooled output is then passed through a fully-connected layer to obtain the labeled output.

In Named Entity Recognition (NER), the hidden state outputs of BERT are fed into a classifier layer with the number of tags as the output units, and the predicted class of each token is obtained using argmax.

For Natural Language Inference (NLI) or Textual Entailment tasks, BERT is trained similarly to the NSP task, with the text and hypothesis separated using the [SEP] token. The [CLS] token is used to obtain the classification result.

In Question Answering, BERT is trained to find the start and end tokens of an answer within a given paragraph. The dot product of each token with the start and end tokens is used to calculate the probability of being the start or end of the answer. The maximum scoring span is chosen as the prediction.

Overall, BERT's pretraining process involves learning contextual representations of words through the MLM and NSP tasks. Fine-tuning then involves using the pretrained model on specific downstream tasks by pooling the [CLS] token or using the hidden states for classification or prediction.

9.3 Implementation of the Data to the BERT model

9.4 Data Collection

The data has been collected from the Hugging Face website. It was an open source data set which comprises sentences commonly used in a bank or financial institution and their corresponding intends.

9.5 Data Preprocessing

Large Language Models like BERT cannot take text inputs in high level language like English, Hindi, French etc. So the training texts have been tokenized into individual words and subwords. These tokens are then converted into token ID based on the ID's provided in the Vocab.txt file. These tokens are then sequentially arranged based on the original text sequence and CLS, SEP tokens are inserted to mark the starting and ending of sequences respectively. The sequences are then feeded to the BERT model in sequence length of 512 tokens at a time. If the sequence length is more than 512 then they are splitted accordings and if they are smaller than they are padded.

9.6 Model Training

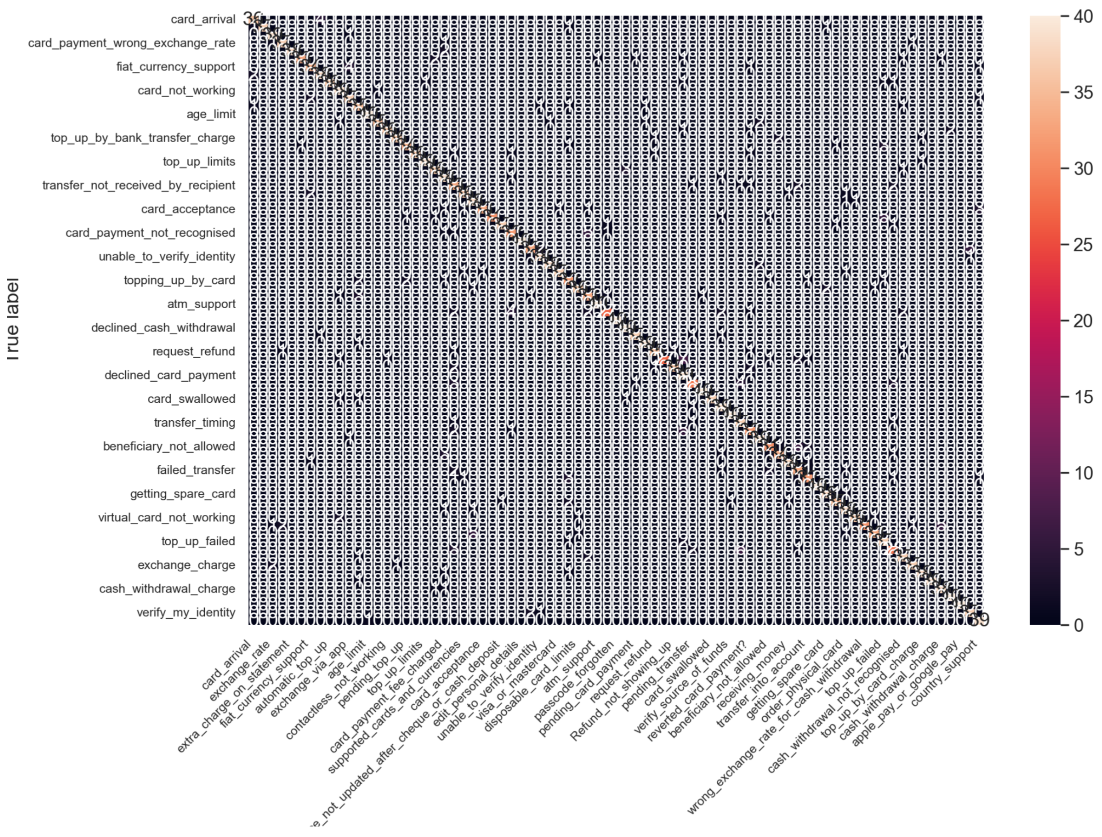
The BERT model is initialized and all the checkpoints files and configuration files are loaded while initialization of the model. The model is then trained with the sentences(tokens) with their corresponding intents.

9.7 Model Testing

Model is tested with the test set provided in the dataset and the accuracy is seen depending upon the intents predicted when sentences are feeded from the test sets.

10.0 Results

Confusion matrix:



11.0 Hardware Tool

Here are the hardware components we have used:

Processor: We have used intel core i5(10 gen) and Ryzen 5 processor
Memory(RAM): Our device memory(RAM) is 16 GB
Storage: Our device storage is 512 GB
Graphics Processing Unit(GPU): Our device GPU is Nvidia RTX 2060.

12.0 Software tools

In our project we use Jupyter notebook and Python 3.9.13 as the major programming language.Jupyter Notebook is a widely used open-source web-based interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and explanatory text. It provides a flexible platform for data analysis, scientific computing, and machine learning tasks. Jupyter Notebook eliminates the need for complex software installations by providing a browser-based interface for writing, executing, and documenting code.Python 3.9.13 offers a solid foundation for your project, combining its versatility, ease of use, extensive library support, and the advantages of the latest Python advancements.

13.0 Conclusion

In conclusion, the speech-to-text intent detection project for visually impaired individuals in a banking scenario offers a powerful solution to enhance accessibility and independence. By leveraging advanced speech recognition and intent classification technology, this project enables visually impaired individuals to interact with banking services using their voice. It aims to bridge the accessibility gap, promote financial inclusion, and provide personalized and efficient banking experiences. With ongoing advancements in speech recognition and intent classification, this project has the potential to transform the way visually impaired individuals access and manage their finances.

14.0 Future Scope

The implementation of the voice-based intent recognition system using BERT in the banking domain opens up several possibilities for future enhancements and developments. Here are some potential areas of future scope:

Expansion to Other Domains: The current system focuses on intent recognition in the banking domain. However, the methodology and techniques can be extended to other domains such as healthcare, e-commerce, customer service, and more. By adapting the training data and fine-tuning the model for specific domains, the system can be applied to a wide range of applications, catering to diverse user needs.

Multilingual Support: BERT has shown remarkable performance in various languages. Expanding the system to support multiple languages would enable a more inclusive user experience, allowing individuals from different linguistic backgrounds to interact with the system effectively. This would involve training the model on multilingual datasets and incorporating language-specific tokenizers and vocabularies.

Continuous Learning: Implementing a mechanism for continuous learning would enhance the system's adaptability and accuracy over time. By periodically updating the training data with new user queries and intents, the system can stay up-to-date with evolving user needs and improve its performance over time. This would require incorporating techniques such as online learning and incremental training into the system architecture.

Integration with Voice Assistants: Integrating the voice-based intent recognition system with popular voice assistants like Google Assistant, Amazon Alexa, or Apple Siri would enable seamless access to banking services through voice commands. Users could simply interact with their voice assistant and perform banking tasks without the need for a separate application. This integration would require developing appropriate APIs and interfaces to facilitate communication between the systems.

Accessibility Features: Enhancing the system's accessibility features for individuals with visual impairments or other disabilities would be a valuable future addition. This could involve integrating text-to-speech capabilities to provide audio feedback and instructions, incorporating accessibility guidelines for user interface design, and ensuring compatibility with assistive technologies.

Enhanced Natural Language Understanding: Further improving the system's natural language understanding capabilities would enable more accurate intent recognition and response generation. Techniques such as semantic parsing, entity recognition, sentiment analysis, and context-aware processing could be integrated to enhance the system's ability to understand user queries in a more nuanced manner.

Real-time Contextual Assistance: Expanding the system to provide real-time contextual assistance to users during their banking interactions would be beneficial. This could involve utilizing contextual information such as user transaction history, account details, and preferences to provide personalized and relevant recommendations or suggestions. By leveraging machine learning algorithms and user data, the system could offer proactive assistance and optimize the user experience.
Security and Privacy Enhancements: Strengthening the security and privacy aspects of the system would be crucial, considering the sensitive nature of banking interactions.

Implementing robust authentication mechanisms, data encryption, and compliance with relevant security standards would instill user trust and ensure the confidentiality and integrity of user information.

18.0 References

[1] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

[2] The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)

[3] PolyAI/banking77

[4] BERT for dummies — Step by Step Tutorial

[5] Attention Is All You Need
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

[6] LIDSNet: A Lightweight on-device Intent Detection model using Deep Siamese Network
Vibhav Agarwal; Sudeep Deepak Shivnikar; Sourav Ghosh; Himanshu Arora; Yashwant Saini

[7] Zero-shot User Intent Detection via Capsule Neural Networks
Congying Xia1, Chenwei Zhang, Xiaohui Yan, Yi Chang, Philip S. Yu