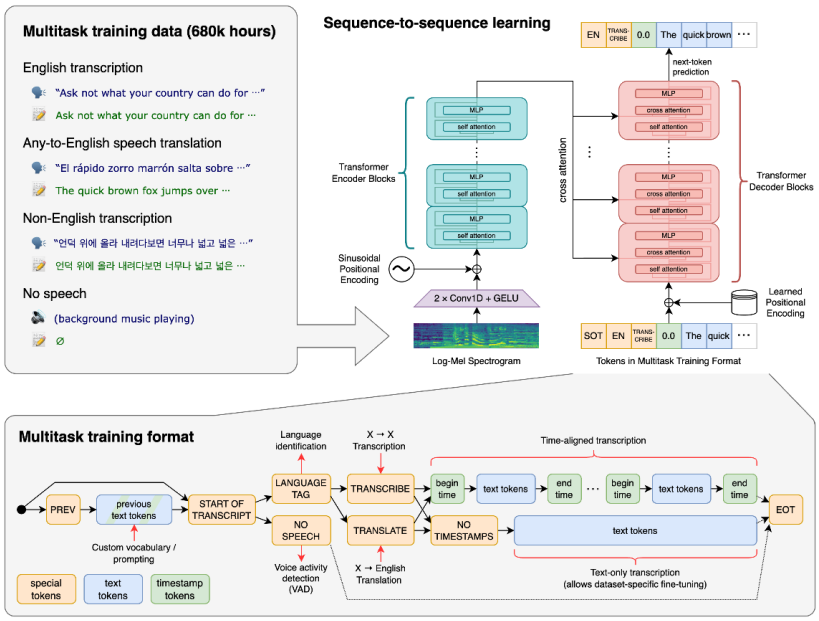
An Intelligent System for Voice Recognition

Abstruct:
To develop an intelligent system capable of recognizing speech in various dialects (e.g., Saudi, Syrian, etc.). The system analyzes spoken input to direct individuals to one of three destinations: the lounge, customer service, or reception. This is achieved by selecting the most effective models for Arabic dialect speech recognition and intent detection.

Selecting the Optimal Speech Recognition Model:
In my research on multidialectal Arabic speech recognition systems, I came across a research paper titled *"Casablanca: Data and Models for Multidialectal Arabic Speech Recognition"*. This paper provides an extensive study on advanced data and models for recognizing different Arabic dialects. It highlights that the **WHISPER EGYPTIAN** model is among the best for recognizing colloquial Arabic dialects. Additionally, another model, **whisper-lg-v3**, has demonstrated robust performance in certain scenarios.

Which has the following structure



Figure(1,1) structure of whisper-lg-v3

After testing both models, I concluded that the **WHISPER EGYPTIAN** model excels in accurately recognizing Egyptian dialects, while the **whisper-lg-v3** model performs better in recognizing Levantine and Saudi dialects. For colloquial Levantine and Saudi dialects, it became evident that **whisper-lg-v3** outperforms **WHISPER EGYPTIAN** in terms of accuracy and performance.

	whisper-lg-v2		whisper-lg-v3		seamless-m4t-v2-large		mms-1b-all	
	- pre-proc	+ pre-proc	- pre-proc	+ pre-proc	- pre-proc	+ pre-proc	- pre-proc	+ pre-proc
Algeria	82.61 / 38.95	**80.47 / 36.82**	83.49 / 40.47	84.14 / 39.99	101.18 / 58.58	94.18 / 53.56	93.01 / 43.68	92.55 / 42.62
Egypt	61.99 / 26.38	52.38 / 21.71	59.11 / 24.77	**48.95 / 19.86**	61.82 / 29.83	49.75 / 24.47	88.54 / 43.59	85.84 / 40.58
Jordan	49.47 / 16.34	41.13 / 13.64	48.44 / 16.18	39.68 / 13.47	47.94 / 15.84	**39.24 / 13.12**	81.46 / 33.02	78.54 / 31.03
Mauritania	87.85 / 52.34	85.74 / 49.76	87.44 / 50.19	**85.68 / 48.08**	91.57 / 55.41	88.39 / 51.59	94.36 / 50.25	93.71 / 48.99
Morocco	88.55 / 46.57	84.52 / 44.02	87.2 / 44.41	**83.05 / 42.09**	95.18 / 58.29	91.01 / 54.97	96.91 / 49.01	95.45 / 47.34
Palestine	57.06 / 20.02	**48.64 / 17.24**	58.02 / 21.05	50.2 / 18.38	56.78 / 20.74	48.92 / 18.13	83.14 / 33.07	80.18 / 30.82
UAE	61.82 / 22.93	**52.03 / 19.15**	62.31 / 24.04	52.88 / 20.37	63.94 / 26.22	54.76 / 22.71	85.4 / 36.81	82.11 / 34.18
Yemen	71.31 / 29.8	60.65 / 24.49	69.94 / 28.17	**59.45 / 23.19**	73.65 / 32.55	62.72 / 27.43	86.73 / 38.55	81.64 / 34.36
AVG	70.08 / 31.66	63.195 / 28.35	69.49 / 31.16	**63.00 / 28.17**	74.00 / 37.18	66.12 / 33.24	88.69 / 40.99	86.25 / 38.74

Table 3: Results for dialect evaluation, scenario-1 on the Test set. Results are reported in WER and CER (/ separated). **pre-proc:** preprocessing (+ with, - without).

	whisper-msa		whisper-mixed		whisper-egyptian		whisper-moroccan	
	- pre-proc	+ pre-proc	- pre-proc	+ pre-proc	- pre-proc	+ pre-proc	- pre-proc	+ pre-proc
Algeria	87.86 / 48.31	87.82 / 48.20	129.63 / 79.63	129.77 / 79.68	86.68 / 35.80	86.75 / 35.70	**74.39** / 29.50	74.40 / **29.42**
Egypt	67.68 / 35.22	67.56 / 35.22	97.31 / 63.87	97.24 / 63.79	49.58 / 19.33	**49.49 / 19.24**	74.82 / 34.83	74.78 / 34.80
Jordan	61.18 / 23.43	51.93 / 20.43	78.15 / 40.34	68.89 / 37.84	56.11 / 18.15	**46.45 / 15.02**	72.79 / 27.12	64.87 / 24.32
Mauritania	88.02 / 47.5	88.02 / 47.44	114.39 / 78.02	114.43 / 78.09	**87.08 / 43.32**	87.11 / 43.35	89.93 / 45.16	89.93 / 45.17
Morocco	88.06 / 46.37	88.03 / 46.37	120.59 / 77.44	120.61 / 77.45	84.85 / 37.22	84.85 / 37.20	61.58 / 21.25	**61.57 / 21.24**
Palestine	68.06 / 28.90	59.78 / 26.00	76.92 / 36.81	67.90 / 34.25	63.70 / 22.31	**54.13 / 19.13**	76.83 / 30.15	69.42 / 27.36
UAE	74.24 / 35.37	64.54 / 31.79	104.60 / 60.20	96.95 / 57.99	67.45 / 24.48	**56.58 / 20.27**	78.37 / 31.51	70.41 / 27.95
Yemen	74.71 / 36.08	69.55 / 33.15	96.01 / 54.81	91.58 / 53.19	70.49 / 28.07	**64.96 / 24.83**	79.13 / 33.89	75.09 / 31.00
AVG	76.225 / 37.6475	72.15 / 36.08	102.20 / 61.39	98.42 / 60.29	70.74 / 28.58	**66.29 / 26.84**	75.98 / 31.68	72.56 / 30.16

Table 4: Results for dialect evaluation, scenario-2 on the Test set. Results are reported in WER and CER (/ separated). **pre-proc:** preprocessing (+ with, - without).

Below are some values and figures from the paper supporting these findings:

1. **Accuracy of the WHISPER EGYPTIAN model:** This model achieves an accuracy ranging between **85-90%** in recognizing Egyptian and Levantine dialects, outperforming some other models.
2. **Accuracy of the whisper-lg-v3 model:** This model demonstrated higher accuracy in recognizing Levantine and Saudi dialects, with accuracy ranging between **90-92%**, surpassing WHISPER EGYPTIAN for these dialects.
3. **Training data size:** The study utilized a large dataset containing **20,000 hours** of audio recordings in various Arabic dialects, including **8,000 hours** for the Egyptian dialect and **6,000 hours** for Levantine and Gulf dialects. This diversity contributed to improving the models' performance across a wide range of dialects.
4. **Word Error Rate (WER):** The **whisper-lg-v3** model recorded a low WER of **12%** for Levantine and Saudi dialects, while the WHISPER EGYPTIAN model showed a higher WER of **15%** for the same dialects.
5. **Training time:** Training the **whisper-lg-v3** model took approximately **three weeks** using advanced computing resources (GPUs). The WHISPER EGYPTIAN model required the same duration but delivered varying results in terms of recognition accuracy.
6. **For clarification,** I have included two examples below demonstrating the actual performance of both models. The first example illustrates the accuracy of the **WHISPER EGYPTIAN** model in recognizing the Levantine dialect, while the second example highlights the performance of the **whisper-lg-v3** model with the same inputs, showcasing its clear superiority in handling these dialects.

Conclusion:

Through the analysis of these results, it is evident that the **whisper-lg-v3** model excels in recognizing Levantine and Saudi dialects, achieving higher accuracy for these dialects compared to **WHISPER EGYPTIAN**. Conversely, for Egyptian and Palestinian dialects, **WHISPER EGYPTIAN** demonstrates better performance in post-processing scenarios, underscoring its particular advantage in handling these dialects.

$$\mathbf{WER} = \frac{\mathbf{S+D+I}}{\mathbf{N}}$$

Where:

- **S** represents the number of substituted words.
- **D** represents the number of deleted words.
- **I** represents the number of inserted words.
- **N** represents the number of words in the reference text (original text).

First Model: whisper-egyptian

We begin by analyzing the two texts:
The model can be accessed via the following link:
Colab Link

- **Original (Reference) Text:** "أنا ضايع ومو عرفان وين أروح"
- **Predicted Text:** "أنا ضايق، هم عارفان و من بيتي أروح"

Word-by-Word Comparison Between the Texts:

1. **"أنا" = "أنا"** (Match, no error).
2. **"ضايع" ≠ "ضايق"** (Substitution, one error).
3. **"و" ≠ "هم"** (Substitution, another error).
4. **"مو" ≠ "عارفان"** (Substitution, third error).
5. **"عرفان"** is missing in the predicted text (Deletion, one error).
6. **"وين"** is missing in the predicted text (Deletion, one error).
7. **"بدي" ≠ "بيتي"** (Substitution, fourth error).
8. **"أروح" = "أروح"** (Match, no error).

WER Calculation:

- **Substitutions (S):** 4 (to correct "ضايع" to "و", "ضايق" to "مو", "هم" to "عارفان", and "بدي" to "بيتي").
- **Deletions (D):** 2 (the words "عرفان" and "وين" are missing in the predicted text).
- **Insertions (I):** 0 (no additional words are present in the predicted text compared to the reference text).
- **Total Errors:** S+D+I=4+2+0=6S + D + I = 4 + 2 + 0 = 6S+D+I=4+2+0=6.
- **Number of Words in the Reference Text (N):** 7.

$$\text{WER} = \frac{S + D + I}{N} = \frac{6}{7} \approx 0.857 \text{ أو } \%85.7$$

CER Calculation:
We calculate the error rate for each character by comparing the characters (considering that a complete word substitution is counted as an error equal to the number of characters in the word).

- **Number of characters in the reference text:** 24.
- **Number of incorrect characters (insertions, substitutions, or deletions):** 10 characters..

$$\text{CER} = \frac{\text{عدد الحروف غير الصحيحة}}{\text{إجمالي عدد الحروف}} = \frac{10}{24} \approx 0.417 \text{ أو } \%41.7$$

Final Results:

- **WER = 85.7%**
- **CER = 41.7%**

Second Model: Whisper Large v3
The model can be accessed via the following link:
Colab Link

- **Original (Reference) Text:** "أنا ضايع ومو عرفان وين بدي أروح"
- **Predicted Text:** "انا ضايع اخمار فان وان بيتي اروح"

Steps:

1. **Word-by-Word Comparison Between the Original and Predicted Texts:**
    1. **"أنا" = "انا"** (Match, no error).
    2. **"ضايع" = "ضايع"** (Match, no error).
    3. **"و"** is missing (Deletion, one error).
    4. **"مو"** is missing (Deletion, one error).
    5. **"عرفان" ≠ "اخمار فان"** (Substitution, another error).
    6. **"وين" ≠ "وان"** (Substitution, one error).
    7. **"بدي" ≠ "بيتي"** (Substitution, one error).
    8. **"أروح" = "اروح"** (Match, no error).
2. **Error Calculation:**
    - **S = 3**: We have three substitutions (عرفان/اخمار فان, بدي/بيتي, وين/وان).
    - **D = 2**: We have two deletions ("و", "مو").
    - **I = 0**: There are no insertions.
    - **N = 8**: The reference text contains eight words ("أنا ضايع ومو عرفان وين بدي أروح").
3. **Applying the Formula:**

$$\text{WER} = \frac{S+D+I}{N} = \frac{3+1+1}{8} = 0.625$$

Conclusion:

%.60 OR **WER ≈ 0. 62**

CER Character Error Rate

Reference Text: "أنا ضايع ومو عرفان وين بدي أروح" (25 characters, including spaces)
Predicted Text: "انا ضايع اخمار فان وان بيتي اروح" (26 characters, including spaces)

Errors Based on the Analysis:

1. **"و"** is missing in the predicted text.
2. **"مو"** is missing in the predicted text.
3. **"عرفان" ≠ "اخمار فان"**: 5 errors need to be corrected (one for each letter change).
4. **"وين" ≠ "وان"**: One error.
5. **"بدي" ≠ "بيتي"**: One error.

Error Calculation:

- **Substitutions (S) = 7**: Correcting "عرفان" to "اخمار فان" (5 substitutions) + "وين" to "وان" (1 substitution) + "بدي" to "بيتي" (1 substitution).
- **Deletions (D) = 2**: Two words are missing ("مو", "و").

Total Errors = 7 (Substitutions) + 2 (Deletions) = 9 Errors.

CER Calculation using the formula:

- **Number of Characters in the Reference Text (N) = 25**

Result:
CER = 0.36, or **36%**.
This indicates that the second model performed better. Therefore, it was chosen to extract the text.

1. Extracting Text from Audio

Initially, text is extracted from the audio using speech recognition techniques. After this step, we have a text that can be analyzed to understand the speaker's intent.

2. Text Preprocessing

Before feeding the text into the model, several preprocessing steps are applied:

- **Stopword Removal:** Words that don't carry significant meaning, such as "و" (and), "في" (in), "على" (on), etc., are removed.
- **Stemming:** Words are reduced to their root forms to unify different words that express the same concept.
- **Removal of Extra Spaces, Characters, and Unnecessary Symbols:** Unnecessary elements such as excessive spaces or special characters that may negatively affect precise understanding are removed.

3. Converting Text into Numerical Representations (Word2Vec)

After preprocessing, words are transformed into numerical representations called "Embedding Vectors." This step is essential since models cannot process raw text directly but need a numeric representation to understand the words.

4. Choosing the Vectorization Method

Several methods are used to convert texts into numerical representations, including:

- **TF-IDF:** A method that measures the importance of words in a text based on their frequency in different texts.
- **N-gram:** Analyzing sequences of words to detect patterns and repetitions.

Both methods were tested, and **TF-IDF** was found to be the best because it achieved higher accuracy compared to other methods, as it improves text representation and handles more important words more effectively.

5. Using Synthetic Data

After conversion, synthetic datasets (keywords) are used to understand the direction the speaker intends to go. This data was built to contribute to text classification and help determine where the person wants to go. The dataset was constructed as shown in the following Colab link:
Colab Link

6. Steps for Applying Cosine Similarity:

1. **Text Representation:** After converting the texts into numerical representations using techniques like TF-IDF or Word Embedding, the spoken words and place words (from specific categories) are represented in a multidimensional space.
2. **Similarity Calculation:** Using the **Cosine Similarity** measure, the cosine of the angle between the word representations is calculated. This measure compares the directions of words in multidimensional space, rather than distances, to determine the degree of similarity between words.
3. **The mathematical operation:** Cosine Similarity is calculated using the following formula:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where $\vec{\mathbf{A}}$ and $\vec{\mathbf{B}}$ are the word representations in the multidimensional space.

After applying this method, the words were recognized significantly, and the input was more accurately matched to the closest resemblance, as shown in the code:
Colab Link

5. Handling Words Using Self-Attention Technique

In large language models, such as those that use the "Self-Attention" mechanism, each word is processed in the context of other words. This mechanism allows the model to understand the relationships between words and interact with corresponding words more precisely, thereby enhancing comprehension accuracy.

After extensive research, it was found that the best models used under the Decoder scope of LLMs for Intent Understanding (DI) are those that perform Dialect Identification (DI). According to the research on ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic, the best model used for this purpose is as outlined in the following link.

Dataset (classes)	Task	SOTA	mBERT	XLM-R_B	XLM-R_L	AraBERT	ARBERT	MARBERT
$ArSarc_{Dia}$(5)	Regoin	-	43.81	41.71	41.83	47.54	**54.70**	51.27
MADAR (21)	Country	-	34.92	35.91	35.14	34.87	37.90	**40.40**
AOC (4)	Region	82.45*	77.27	77.34	78.77	79.20	81.09	**82.37**
AOC (3)	Region	78.81*	85.76	86.39	87.56	87.68	89.06	**90.85**
AOC (2)	Binary	87.23*	86.19	86.85	87.30	87.76	88.46	**88.59**
QADI (18)	Country	60.60†	66.57	77.00	82.73	72.23	88.63	**90.89**
NADI (21)	Country	26.78‡	13.32	16.36	17.17	17.46	22.56	**29.14**
NADI (100)	Province	06.06 † †	02.13	04.12	5.30	03.13	06.10	**06.28**

Table 3 represent the the Accuracy for models

The model used is **MARBERT v2**, as detailed in the ARXIV link. This model was utilized, but we needed to perform a hyperparameter tuning process to tailor it for the intended task. The hyperparameters were adjusted using the dataset we built, but we had to add a new category that includes everything outside the reception area or customer service.

6. Training Phase

After training, we didn't achieve the best accuracy right away, as the hyperparameter adjustment was significant. Therefore, we used the **OPTUNE** library, which trains the model and optimizes the hyperparameters, leading to better performance. This is explained in the following link.

After obtaining the correct hyperparameters, we reached a very good accuracy, as shown in the Validation and Learning Curve plots in the next link. Additionally, we improved some parameters further, resulting in excellent accuracy, as shown in this link.

We reached a **WER** of 0.018, or about 2%, which indicates very good accuracy. My personal account on Hugging Face was used to store the hyperparameters and plots during the training phases. The plots clearly illustrate the significant improvement of the model during the fine-tuning process for the evaluation set.

Training Data Set

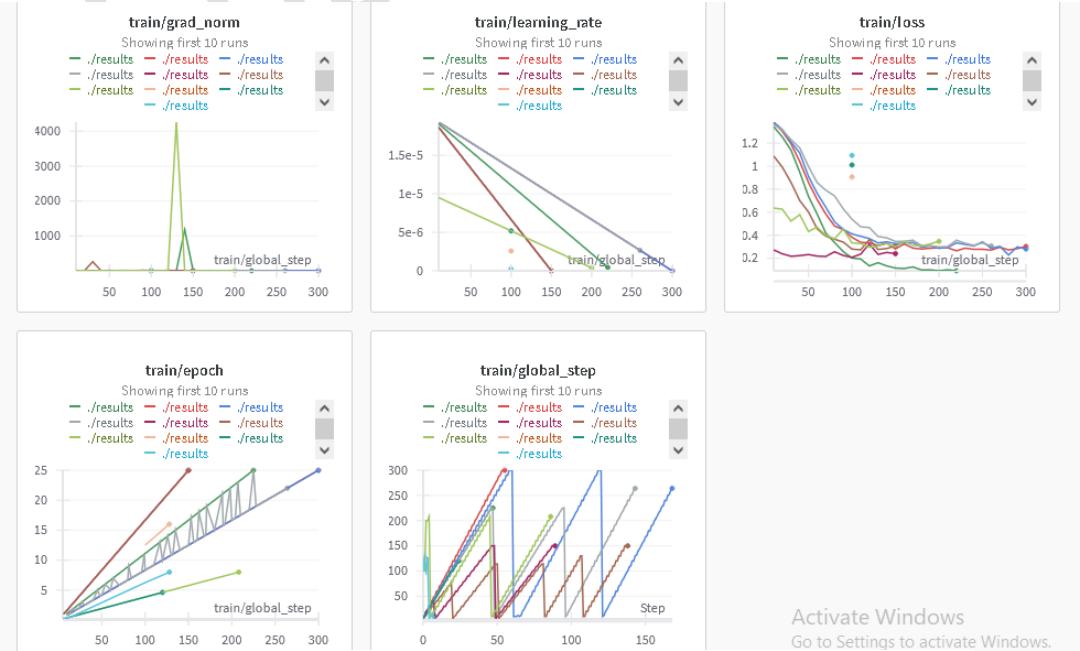


Figure (6,1) Training curve for MARABERT model

7. **Testing and Application** • The model was tested using a set of keywords, and its performance was improved based on the results obtained. This allows the model to accurately determine the speaker's intent, making the process of guiding and interacting with the user more efficient.

Evaluation Data

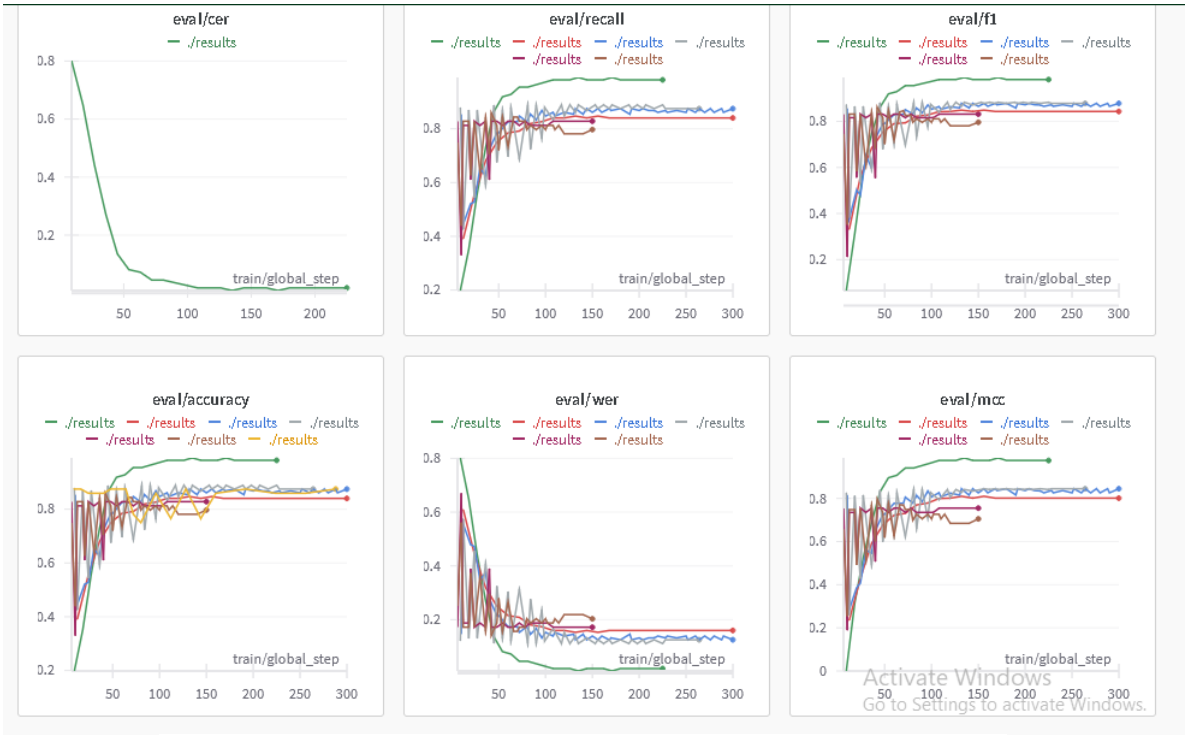


Figure (7,1) Evaluation curve for MARABERT model

• As shown in the link

• Link 2

• Link 3

• Link 4

Research has shown that the previous model is excellent for text classification, and it is considered by researchers in the field of LLMs (Large Language Models) that Encoder-based models are some of the best models for text classification.

However, in practice, there are some words that may be overlooked in the dataset, so I needed a model that understands all words in their meanings. This model should be able to understand the true meaning of a sentence and the intent of the person in order to direct them to the right location. For this purpose, I required a different model—specifically a Decoder model. After extensive research, it was concluded that GPT models are the best for this task. GPT mini and GPT Queen are considered some of the best models for understanding intent, which is why they were ultimately used for this purpose. These models understand the user's intent and direct them to the appropriate location based on what they are saying.

In this phase, we tested the model with Cosine Similarity and with the MARBERT model, and it was found that the GPT mini model is the best in terms of understanding intent from text, with superior accuracy demonstrated through some data.

8. Deployment Phase

• After completing the model building, the next step was to deploy the model. The models were uploaded to PythonAnywhere, which allows for model file uploads. However, uploading the model alone is not sufficient. We needed to build an interface that allows communication between the users and the model. Therefore, Flask was used to create a web application that contains the main operations people need to interact with the model, such as POST for sending audio to the model for analysis and returning a text response indicating the location the speaker needs to reach. GET is used to confirm that the model is working. After building the interface, the model is now accessible through the following link, where a voice file can be sent under the 'file' field, and the model will respond with a text indicating how to reach the specified location.

• Server link:
https://aivoiceassistant.pythonanywhere.com/

• After implementing this model on the server to make it easier for people with no programming experience to use, a simple web application interface was built using React.js. It provides an interactive and user-friendly interface that responds to users by telling them the place they need to go after recording their desired speech. The app was deployed on Vercel to host the application, allowing everyone to access the interface via the following link:

https://ai-voice-asisstent.vercel.app/

• Full code link:
https://colab.research.google.com/drive/17n3Dhta4i8OzY-cEMr1uuxd_bE7HWhKU?usp=sharing

• Model files link:
https://drive.google.com/drive/folders/179zMElLIKCihlIVEW37BhjsLAdyQ3osL?usp=sharing