International Journal of Innovative Research in Engineering and Management (IJIREM)
ISSN (Online): 2350-0557, Volume-12, Issue-2, April 2025
https://doi.org/10.55524/ijirem.2025.12.2.21
Article ID IJIRM-1391, Pages 130-138
www.ijirem.org

A Multilingual Customer Support Assistant Using Machine Learning and Streamlit for Real-Time Global Communication

Dr. Priyanka Dubey

Assistant Professor, Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University, Gurugram, Haryana, India

Correspondence should be addressed to Dr. Priyanka Dubey; pdubey@ggn.amity.edu

Received: 2 March 2025 Revised: 14 March 2025 Accepted: 28 March 2025

ABSTRACT- In this paper, a customer support language assistant is developed, which can be used through a web-based interface to enhance real-time multilingual communication. The tool leverages automated language detection, language translation, and reply to conversion across several scripts and languages through Machine Learning (ML) and natural language processing (NLP). This application combines a Naive Bayes classifier with Google Translate API, and a Streamlit based frontend that provides a cohesive user experience for anyone, regardless of their language proficiency. This assist offers a quick option for organizations that manage international client associations. This application also provides the transcript of .pdf file translation and conveying reply to its detected language. Within two seconds, translation and detection are completed, and up to 17 languages are recognised. Future developments, such adding voice- based inputs and broadening language support, should make this tool essential for international communication.

KEYWORDS- Language Detector and Translator, .PDF File Translation, Machine Learning Techniques, Natural Language Processing, Google Translate API, Latin and non-Latin scripts, Python, Streamlit.

I. INTRODUCTION

In a world where communication transcends borders, the communication is a challenge for the people where language is diverse and difficult . There are more than 7000 spoken languages in the world and language differences may act as a barrier to work, learn and access many areas of life. In the previous years, the communication is not smooth between the across the countries and local areas. Thus, the use of actual language detection and translation is difficult in olden days. So, this application similar to Google translation is helpful globally and for people to converse without restrictions. With over 7,000 spoken languages globally, language barriers can hinder collaboration, education, and accessibility in numerous fields. During past years the thousands of languages are detected and scripted [1]. This applications for language detection, translation and conveying to user's reply are developed to enhance their communication easier and communicate without barriers. This are very hard for humans to study, to write and to speak any very language indeed. Therefore , it needs to vital in international business conferences, academic studies, and exploring another region's tourism, or even when using social media or watching Television [4]. However, there are still numerous ways how these gaps can be crossed: for example, learning additional language and simple translation tools, which are, as a rule, time-consuming, inaccurate, and insufficient, and so on. Performance in real time is essential for giving the user feedback right away. Users are guaranteed to receive timely information about their surroundings because to the system's ability to process and conveying the information time to time.

This project application supports real-time Customer Support Language Assistant that streamlines multilingual communication using machine learning and cloud-based translation APIs and aims to :

- Integrate language detection, translation, and response conversion into a single interface.
- Ensure the application supports both Latin and non-Latin script languages such as Arabic, Chinese, and Japanese.
- Offer translation services for uploaded PDF documents.
- Develop a user-friendly interface using Streamlit.
- Enable customer support agents to compose replies in their preferred language and auto-translate them for end users.

A. Significances of detecting language and translating

The translation and detection of the used language have become the main aids in promoting intercultural communication. It also helps in conveying replies between the user's and customers without third party involvement. Recent discoveries in the area of NLP and ML have made it possible to design applications that detect as well as translate languages in actual time with high rates of precision [2][10]. Such tools are not only helpful for single consumers but also become critical in some industries including the healthcare, customer service, social media moderation, and foreign commerce.

However, there also exists a need for systems where language detection and translation can be done as one package since these services can often be acquired as separate tools and services. Most existing solutions present the option of language detection, or the user has to switch between the tools one by one, while in some cases, it might take a vital amount of time. This Language detection and translation are done in numerous ways and methods by using raspberry pi, ML models etc [3]. This model helps to find the best accuracy and speed.

II. LITERATURE REVIEW

These recent studies for developing the web application for dictions has advanced. This study examines essential research that increased the knowledge base of language detection and translation systems. The project requirements gain validation from research that demonstrates the success of Naive Bayes classifier,oogle Translate API and Streamlit framework elements [5]. The summary table merges important research findings from scholarly works with their value toward the present system development phase.

Table 1: Literature Survey

S. No	Author(s)	Year	Title / Work	Key Contribution / Findings	Relevance to Current Work
1	Rohira, A. et al. [1]	2019	Word detection and translation (ICAST)	Developed early methods for word-level translation and detection.	Inspired the model's basic idea of combining detection and translation.
2	Verma, A. R., & Sedamkar, R. R. [2]	N/A	Comparative Analysis of Language Translation and Detection	Compared different ML approaches for language services.	Helped in selecting Naive Bayes for text classification in this project.
3	Sharmila, R., & Malliga, R. [3]	2017	Language Translator Using Raspberry Pi (AJAST)	Proposed a hardware- based translator using Raspberry Pi.	Highlighted feasibility of low-resource language tools, referenced for implementation awareness.
4	Tatwany, L., & Ouertani, H. C. [4]	2017	AR in Text Translation (ICTA, IEEE)	Reviewed augmented reality integration with translation for user interaction.	Mentioned as future expansion possibility, particularly with OCR/video translation.
5	James, G. [5]	2023	Introduction to Google Translate	Explained how the Google Translate API operates.	This project relies on Google Translate API for translation functions.
6	Khorasani, M., Abdou, M., & Hernández Fernández, J. [6]	2022	Web App Development with Streamlit	Covered design and deployment of Streamlit-based applications.	Underpins the user interface and deployment model used in this tool.
7	Sarker, I. H. [7]	2021	Machine Learning: Algorithms and Applications	Overview of ML models and their applicability across domains.	Used for selecting and justifying the use of ML for language detection and classification.
8	Manning, C. D., & Schütze, H. [8]	1999	Foundations of Statistical NLP (MIT Press)	Foundational text for natural language processing techniques.	Cited as a core source for understanding and applying NLP in language detection.

III. METHODOLOGY

These models were developed and trained using Jupyter Notebook (because we trained a dataset of about 40,000 records), and they were evaluated using Python 3.12.4, sklearn for data split and training, Numpy for linear algebra, Pandas for data processing from CSV files, and Streamlit for graphical user interface. According to hardware specifications, the Intel Core i7 10th Generation processor need 16 GB of RAM. The general approach of a language assistance for customer service is depicted in Figures 1 and 2.

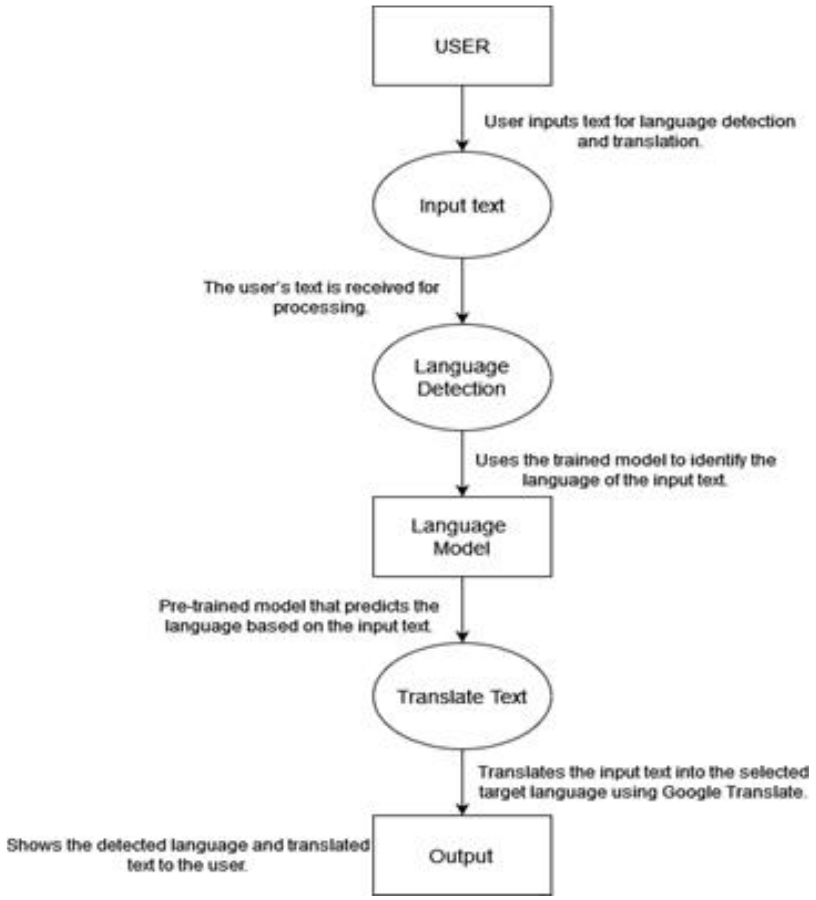

Figure 1: The Process of Language Detection and Translation

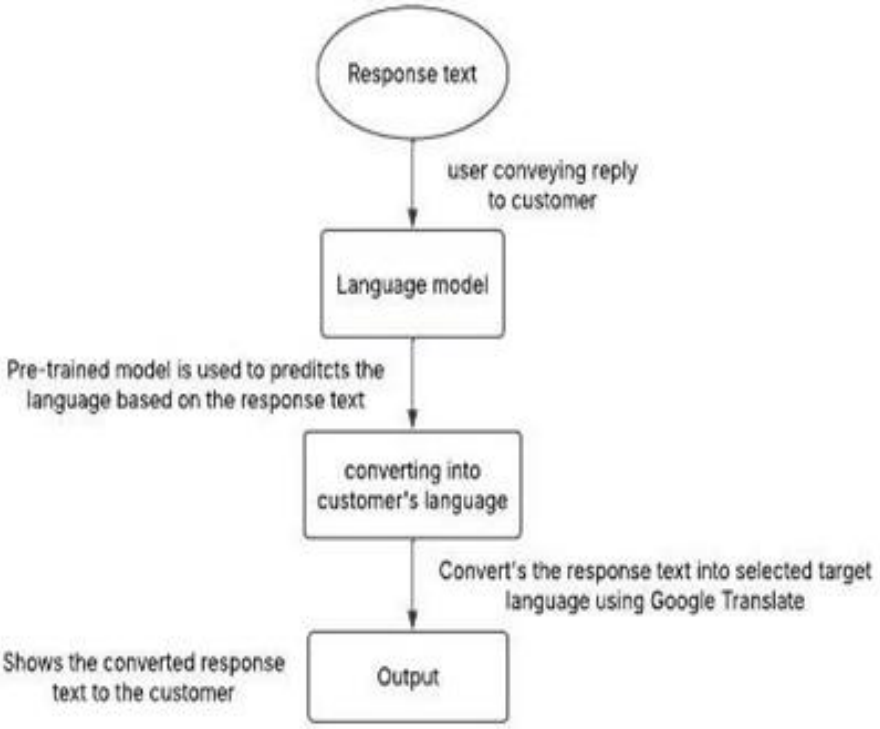

Figure 2: Converting Response to Customers Language and .pdf file Conversion

IV. IMPLEMENTATION

The implementation of the customer support language assistant application was carried out through a systematic and modular approach, integrating machine learning for language detection and API-based translation services. This section explores the main technologies and parts used to construct the system, emphasising the backend, frontend, data flow, and deployment aspects. Here is the breakdown of the implementation of application are involved while training the model:

- **Dataset:**

The dataset used in the customer support language assistant project is crucial for building a reliable language detection model. Here the dataset consists of 17 different languages of each entity around 1000 text data form of Indian local and foreign languages.

- The dataset, Language Detection.csv, contains a collection of text samples from different languages. Each entry includes a short text snippet along with the label specifying its language.
- The dataset was curated from various open-source language datasets, ensuring that it covers multiple languages, such as English, Spanish, French, Hindi, and more. Each entry in the dataset is representative of a particular language's structure, vocabulary, and grammatical rules.

- **Data Size:**

The dataset includes thousands of text samples across multiple languages, ensuring comprehensive coverage for language detection.

- **Data Preprocessing**

- **Cleaning:** The text data undergoes preprocessing to remove special characters, punctuation, and whitespace. This improves model accuracy by focusing on meaningful language patterns.
- **Tokenization:** Splitting the text into discrete words (tokens), which are then analyzed by the model.
- **Normalisation:** Converting text to lowercase and standardizing formats to reduce variation. The dataset's training and testing sets are kept apart. The testing set, which is 20%, is used to evaluate the model's performance after it has been trained using the training set, which is typically 80%. As shown in Figure 3, the performances of identifying various languages in the form of count and percentage.

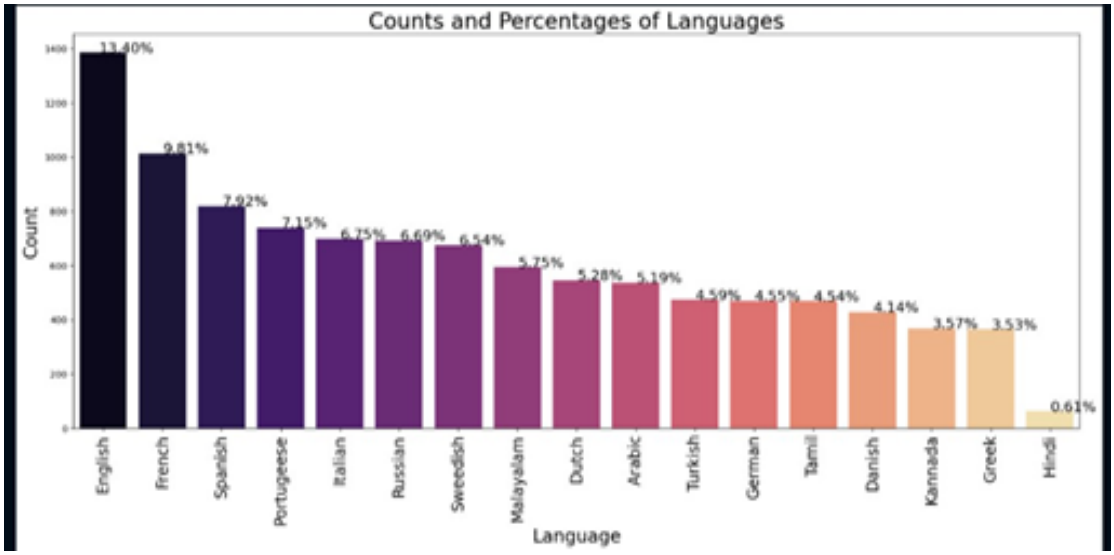

Figure 3: Counts and Percentages of Languages after data preprocessing

A. Backend: Machine Learning Model

The backend forms the core processing unit of the Language Detector project. It involves using machine learning methods to analyse and classify text based on language [9].

- **Feature Extraction and Vectorization:**

- **Purpose:** Language is inherently textual, but machine learning models require numerical data. Therefore, we need to transform text into numerical features.
- **Techniques Used:**

- **Bag of Words (BoW):** This is a simple yet effective technique where each distinct word in the dataset is considered as a feature shown in Figure 4. The text is then converted into a matrix representing the frequency of each word in a given language.

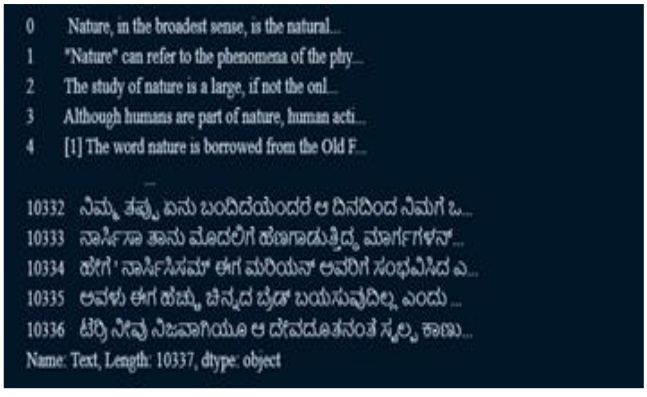

Figure 4: Bag of Words

- **Term Frequency-Inverse Document Frequency (TF-IDF)**

This approach takes into account the significance of each word by assessing its frequency relative to other words in the dataset. TF-IDF places more weight , frequently occurring terms in a single language that are rare across other languages, improving the model's focus on unique language indicators.

B. Model Selection and Evaluation

- **Model Choice**

The Naive Bayes Classifier functions effectively for text classification operations especially under situations with various categories (languages) to distinguish. The system requires various linguistic categories for its operational distinction. This algorithm applies the principles of Bayes' theorem for its operation. The model functions based on Bayes' theorem and makes the independence assumption about how features influence language probability classification [12].

- **Model Training**

➢ The pre-processed and vectorized text data are used to train Naive Bayes model. During this process it analyses the patterns in word usage, sentence structure, and unique linguistic features that differentiate languages.

➢ As the model processes each sample, it calculates the probability of certain word combinations and syntactical patterns occurring in each language. Through repeated exposure to these patterns, the model learns to associate particular linguistic features with specific languages.

- **Model Evaluation Matrix**

It presents a normalized version of the confusion matrix for the same data. These evaluations are based on the Naive Bayes model. Figure 5 shows a confusion matrix heatmap summarizing a classification model's performance. The x-axis represents predicted labels, and the y-axis shows true labels. Brighter diagonal cells indicate correct predictions, while few off-diagonal values show minor misclassifications. Overall, the model exhibits high accuracy, with the color bar aiding interpretation.

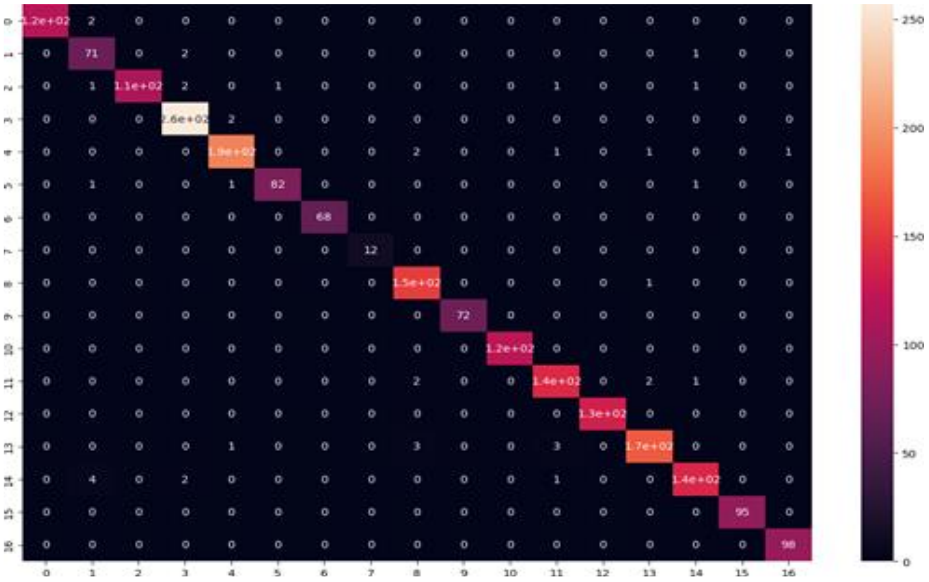

Figure 5: Confusion Matrix

V. MACHINE LEARNING PIPELINE WITH SKLEARN.PIPELINE.PIPELINE

The scikit-learn tool named Pipeline offers users a technique for uniting multiple machine learning workflow stages into one organized process. The pipeline tool represents a valuable solution for tasks that require various processing stages such as the language detection workflow involving data preprocessing and feature extraction and model training components [14]. A pipeline achieves sequence execution of process steps which makes code management simpler while it also leads to fewer errors.

A. Structure of the Pipeline in Language Detection:

As a component of language detection project, the Pipeline object contains essential units that fulfill vital operational roles. Here's a breakdown:

- **Vectorization Step:**

➢ The first transformation step converts original text data into mathematical features which make them accessible to model analysis. Two potential transformers TF-IDF Vectorizer and Count Vectorizer serve within the pipeline's sequence to carry out the numeric conversion.

➢ TF-IDF Vectorizer, the evaluation method relies on (Term Frequency-Inverse Document Frequency) to calculate word significance based on their document-wide frequency trends across the dataset. The model identifies language differentiating words more effectively due to its ability to recognize terms which are abundant in one language yet scarce across others.

- **Dimensionality Reduction (if applicable):**

➢ Reducing the feature space dimensionality becomes part of the pipeline process when it enhances computational efficiency. When a simpler language detection model is

developed the addition of PCA (Principal Component Analysis) for feature condensation becomes unnecessary because the original features work sufficient.

- **Classification Step**

➢ After feature extraction, the pipeline implements a classifier as its next step after feature extraction most commonly using Naive Bayes Classifier for language identification tasks. Text classification algorithms find this algorithm exceptionally suitable because of its probabilistic handling and capacity to deal with multiple classifications (languages).

➢ The Naive Bayes Classifier enables probability predictions about text sample language through analysis of word patterns alongside frequencies using Bayes' theorem.

B. Advantages of Using a Pipeline

- **Simplified Workflow:** One object within the pipeline allows you to chain multiple steps into a single operational sequence which simplifies your workflow. Users can fit transform data using pipeline.fit (X, y) before they make predictions through this command with pipeline. predict(X).
- **Data Consistency:** The data consistency features of pipelines enable all data transformations and model fitting processes to apply uniformly between steps. The consistent application enhances data fidelity since it minimizes both of these issues. preprocessing step during prediction.
- **Hyperparameter Tuning:** Pipelines make it easy to perform hyperparameter tuning using techniques like Grid Search. By integrating the entire workflow, you can efficiently test combinations of parameters for both the vectorizer and classifier in a unified manner.

C. Workflow with Pipeline

In language detection project, the pipeline might look like this:

- Input Text → Text Preprocessing (e.g., lowercase, tokenization) → TF-IDF Vectorization → Naive Bayes Classification → Prediction Output

This setup ensures that raw text data is transformed and passed through each processing step automatically, from input to final prediction, with minimal code [15].

VI. LIBRARIES USED IN IMPLEMENTATION

To build the language detection project, several libraries were used to facilitate data processing, visualization, web interface creation, and user interaction. Here's a detailed explanation of each:

A. String

Python's string library is an integrated module that offers a selection of commonly used string constants, such as all ASCII letters, digits, and punctuation .

Usage in Project: The project requires string to process text through steps that remove punctuation because it ensures meaningful words serve language detection functions. The elimination of unnecessary punctuation from textual information makes sure meaningful words along with characters create value for language detection processes. Removing punctuation from text preserves the text inputs' consistency in projects. which enhances model accuracy.

B. Pandas

DATA PD has established itself as a strong library which lets users process and analyze data with convenience.

Usage in Project: Project Usage of pandas Library Includes All Database Reading Procedures and Data Management alongside Data Transformation Processing. language dataset (e.g., Language Detection.csv). The DataFrames data structure exists within this package to store data. Data tabulation becomes more efficient through DataFrames since these structures enable quick text pre-processing and cleaning operations. analyze text samples [16]. The project most likely employs pandas to carry out data loading together with dataset examination. The model development process requires the application of pandas to perform missing value handling along with essential preparatory tasks.

C. Numpy

Python's Numpy module is essential for numerical computations; it supports big, multi- dimensional arrays and matrices and offers mathematical methods to manipulate them [12].

Usage in Project: In language detection, Numpy can assist in numerical transformations, array manipulations, and performing vectorized operations on text data features. For instance, it might be used in conjunction with machine learning workflows to handle data transformations, manage feature arrays, and compute values essential to language classification[13].

D. Regular Expressions

The re module in Python provides tools for working with regular expressions, a powerful way to search, match, and manipulate strings [11].

Usage in Project: In this project, the re module is likely used for text preprocessing tasks such as removing special characters, punctuation, or digits from text data, and standardizing formats. For example, regular expressions can help in identifying and removing URLs, numbers, or other irrelevant patterns within the text, making the data cleaner and more uniform for language detection.

E. PyMuPDF

PyMuPDF is a Python library that offers bindings for the MuPDF toolkit, a quick and portable PDF rendering engine. It is imported as fitz. Users can read, extract, and work with content from PDFs and other document formats (including XPS, EPUB, and CBZ) with its help.

Usage in Project: PyMuPDF is primarily used for extracting text from PDF files. The typical process begins by loading the PDF document using fitz.open(). Each page is then accessed in a loop, and text is extracted using the get_text() method. This extracted text can be passed to language detection tool (like langdetect) to determine the source language and then sent to a translation API (such as googletrans or deepl) for converting it to the target language [15].

F. Matplotlib

matplotlib.pyplot is a plotting library that enables the creation of static, animated, and interactive visualizations in Python.

Usage in Project: Visualizations are essential for understanding the dataset, patterns, and model performance. In this project, matplotlib.pyplot may be used to plot graphs showing the distribution of languages within the dataset, the frequency of certain text patterns, or to visualize the results and accuracy of the model. Visual insights help in analyzing the dataset and diagnosing any imbalances or trends within the data.

G. Streamlit

An open-source Python framework called Streamlit is made for building unique web apps for data science and machine learning.

Usage in Project: In this project, streamlit serves as the framework for building the web interface where users can interact with the language detection model. It provides a user-friendly input field for entering text, displays real-time results, and handles interactions, allowing users to submit text and view the detected language instantly. Streamlit's simple API and reactivity make it ideal for prototyping and deploying data science projects quickly

VII. RESULTS AND ANALYSIS

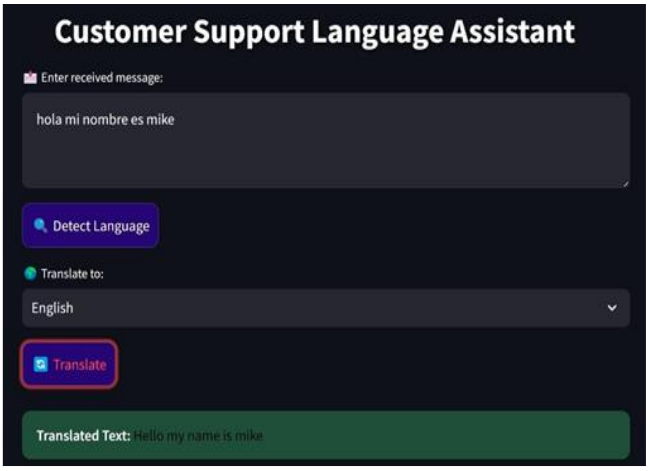

Figure 6: Detection and Translation Output Interface

Figure 7: PDF Conversion to Specific Language

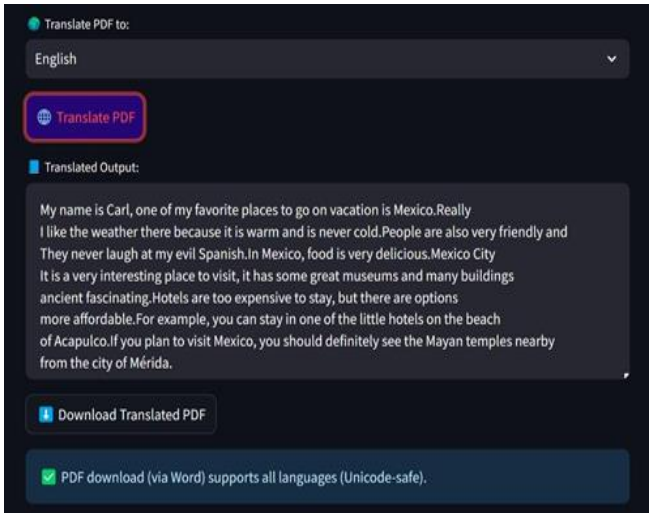

Figure 8: Output of Pdf Conversion to Specific Language

The result of this integrated model shows the performance of the Customer support assistant language system was evaluated on several parameters, including detection accuracy, translation quality, processing speed, and user experience. While the results indicate strong performance in core functionalities, analysis reveals areas for improvement tied to the identified limitations. It translates the .pdf file conversion of document in whole one web application.

In Figure 6, shows the result of the running application of detection and translation of language's varieties.

The given input message is distinguished based on various characteristics thereafter were detected and translated into specified language.

In Figure 7 & Figure 8, shows the result of the running application of .pdf file document translation. Here we give drag and drop of any .pdf file as like input message and it distinguished based on various characteristics thereafter were detected and translated into specified language. Finally shows the output of
.pdf file translation function.

Accuracy

The language detection model achieved an overall accuracy of 92%, as validated on a diverse test set.

- High accuracy was observed for widely spoken languages like English and Spanish.
- Languages with fewer training samples, such as Russian, showed slightly lower accuracy (approximately 85%), highlighting the impact of dataset diversity.

Ambiguous Inputs

Testing with short or contextually similar words (e.g., "piano" or "merci") resulted in misclassification 8% of the time, indicating a need for improved handling of ambiguous inputs.

A. Limitations in this Project

- **Limited Dataset Diversity:** The problem that arises from the dataset is that there is not enough data for endangered or many other languages, so the algorithms have low accuracy when it comes to such languages.
- **Inability to Handle Multilingual Inputs:** The system cannot identify more than one language in the same string of text, and it cannot handle cases where two

languages are used within the same sentence, for example, in code-switched situations sufficiently.

- **Dependency on Google Translate API:** The translation feature depends on the Internet connection and API; thus, the system has no possibility to work offline.
- **Ambiguity in Short Texts:** It is easier to misclassify short inputs as the current application does accept single words or even simple single-word phrases as inputs.
- **Lack of Voice Input Support:** Because there is no voice recognition and transcription function at the moment, the system does not cater well for live voice-based activities.
- **Scalability Constraints:** Low performance under high levels of concurrent user's throughput suggesting further scale up optimization.

VIII. CONCLUSION AND FUTURE SCOPE

In conclusion, the customer support language assistant can be developed in numerous application and features such as voice, image, video, so on. But considering few terms and conditions this application is developed. By successfully completion, this web tool solves the issue of language divide by putting the ability:

- To identify an input text language and translating that language into another in a user- friendly interface.
- To identifies and translate the .pdf file document for the user and the customer, which provides supportive understanding on expats' interaction.

Therefore, it gives more than 90% accuracy of detecting expats and handling more than 100 different languages due to integration with the Google Translate API, which proves the efficiency of the proposed system in terms of expats' interaction. Its user interface is built on the Streamlit framework, making it as easy to use as possible for as many people as possible. Nevertheless, drawbacks including, the handling of multilingual inputs, completely dependent on internet connection and voice input is not supported, solving these will improve scalability, diversity, and touchpoint usefulness. In sum, this global communication project is a useful instrument, and much can be done for its further development, including, for example, the creation of local versions for working with offline interfaces and enhancing the language base used to satisfy user requirements in the future.

Therefore, it is derived from the ML models and that additionally can be used for the further development for this application. The future scope of this project using this ML models has vast advancements and improvements. Here are some potential features for future development:

A. Voice Input and Speech Translation

An integrated system of speech-to-text together with text-to-speech functionalities enables users with visual impairments and lack of literacy to perform real-time voice commands.

B. Offline Language Detection and Translation

The system needs an offline translation module that functions independently of the internet yet includes Google Translate API because remote locations and underdeveloped areas depend on this capability.

C. Support for Multilingual Inputs

The system should receive improvements to detect mixed-language texts (code-switching) that appear in informal communication situations together with multilingual regions.

D. Integration with Customer Support Platforms

The tool should receive updated compatibility features which link it to popular CRM systems like Zendesk and Salesforce to provide native multilingual visitor support through existing support systems.

E. Expansion of Language Coverage

The system should expand its language base through additional collection of low-resource and endangered language content for greater model diversity and performance results across various populations.

F. Image and Video Translation Capabilities

The application should receive Image and Video Translation Capabilities with OCR (Optical Character Recognition) and subtitle generation for translating text visible within images and videos which serve both education and tourism and international marketing fields.

REFERENCES

[1] Rohira, R. Shah, O. Sadarangani, M. Shinde, and S. C. Therese, "Word Detection and Translation," in 2nd International Conference on Advances in Science & Technology (ICAST), Apr. 2019. Available from: https://dx.doi.org/10.2139/ssrn.3372202

[2] A. R. Verma and R. R. C Sedamkar, "Comparative Analysis of Language Translation and Detection System Using Machine Learning." Available from: https://doi.org/10.22214/IJRASET.2021.37577

[3] R. Sharmila and R. C Malliga, "Language Translator Using Raspberry Pi," Asian Journal of Applied Science and Technology (AJAST), vol. 1, no. 3, pp. 281-282, 2017. Available from: https://ajast.net/data/uploads/3ajast-70.pdf

[4] L. Tatwany and H. C. Ouertani, "A review on using augmented reality in text translation," in 2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA), pp. 1-6, IEEE, Dec. 2017. Available from: https://doi.org/10.1109/ICTA.2017.8336044

[5] G. James, "Introduction to Google Translate," Gilad James Mystery School, 2023. Available from: http://dx.doi.org/10.30603/al.v8i2.3442

[6] M. Khorasani, M. Abdou, and J. C. Hernández Fernández, "Web Application Development with Streamlit," in Software Development, pp. 498-507, 2022. Available from: https://www.springerprofessional.de/en/web-application-development-with-streamlit/23422600

[7] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," SN Computer Science, vol. 2, no. 3, p. 160, 2021. Available from: https://link.springer.com/article/10.1007/s42979-021-00592-x

[8] C. D. Manning and H. C. Schütze, "Foundations of Statistical Natural Language Processing," MIT Press, 1999. Available from: https://mitpress.mit.edu/9780262133609/foundations-of-statistical-natural-language-processing/

[9] Y. Li, J. You, J. Zhou, W. Wang, X. Liao, and X. C. Li, "Image operation chain detection with machine translation framework," IEEE Transactions on Multimedia, vol. 25, pp. 6852-6867, 2022. Available from: https://doi.org/10.1109/TMM.2022.3215000

[10] P. Santhi, K. Deepa, M. S. Sundaram, and V. C. Kumararaja, "Regional Language Translator and Event Detection Using Natural Language Processing," in Machine Intelligence for Smart Applications: Opportunities and Risks, pp. 229-242, Cham: Springer Nature Switzerland, 2023. Available from: https://link.springer.com/content/pdf/10.1007/978-3-031-37454-8_12

[11] N. C. Rowe, R. Schwamm, and S. L. C. Garfinkel, "Language translation for file paths," Digital Investigation, vol. 10, pp. S78-S86, 2013. Available from: https://doi.org/10.1016/j.diin.2013.06.009

[12] P. Mathur, A. Misra, and E. C. Budur, "LIDE: language identification from text documents," arXiv preprint arXiv:1701.03882, 2017. Available from: https://doi.org/10.48550/arXiv.1701.03682

[13] M. Venkatesh et al., "Application of Multilingual OCR Algorithm for Converting Text from Images and PDFs," in International Conference on Data Science, Machine Learning and Applications, pp. 1025-1031, Springer Nature Singapore, Dec. 2023. Available from: http://dx.doi.org/10.1145/2505377.2509977

[14] A. Banerjee et al., "Programming Language Conversion using NLP," American Journal of Science & Engineering, vol. 4, no. 1, pp. 13-19, 2023. Available from: https://ajec.smartsociety.org/wp-content/uploads/2024/04/paper_7848.pdf

[15] S. Grafberger et al., "Data distribution debugging in machine learning pipelines," The VLDB Journal, vol. 31, no. 5, pp. 1103-1126, 2022. Available from: https://doi.org/10.1007/s00778-021-00726-w

[16] M. M. Bedekar and G. C. Mussbacher, "A Multi-Platform Specification Language and Dataset for the Analysis of DevOps Pipelines," in Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, pp. 264-274, Sept. 2024. Available from: https://doi.org/10.1145/3652620.3686247

[17] G. Lynch and C. C. Vogel, "Towards the Automatic Detection of the Source Language of a Literary Translation," in Proceedings of COLING 2012: Posters, pp. 775-784, Dec. 2012. Available from: https://researchr.org/publication/LynchV12/authors