Design of translation software with multi-inputs to translate English to Hindi OR vice-versa

A PROJECT REPORT

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

AIML

Submitted by:

ANKIT KUMAR (20BCS6801)
ANEKIT MITTAL (20BCS6819)
BIPAN PANNU (20BCS6809)
AKSHIT JINDAL (20BCS6728)

Under the Supervision of:

Lata Gupta (E13365)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING APEX INSTITUE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **"Design of translation software with multi-inputs to translate English to Hindi OR vice-versa"** is the bonafide work of **"Aniket Mittal(20BCS6819), Ankit Kumar(20BCS6801), Bipandeep Singh Pannu(20BCS6809), Akshit Jindal(20BCS6728)"** who carried out the project work under my/our supervision.

SIGNATURE

Aman Kaushik
HEAD OF DEPARTMENT
AIT-CSE

SIGNATURE

Dr. Lata Gupta
SUPERVISER
Professor
AIT-CSE

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I have dedicated myself in this project. Although, it would not have been achievable without the support and help of many personal and administration. I would like to acknowledge all of them.

I am extremely grateful to Chandigarh University for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

In the achievement of fulfillment of my project on Design of translation software with multi-inputs to translate English to Hindi OR vice-versa to prom, I would like to send my special gratitude to my mentor Ms. Lata Gupta, of AIT Department. I would like to thank her for giving his precious time and valuable efforts, guidance and suggestions that helped me in various phases of the completion of this project. I will always be grateful to her.

Ultimately, as one of the students, I would like to acknowledge my teachers for their support and coordination. As well as their abilities in developing the project and have willingly helped me out.

TABLE OF CONTENTS

List of Tables

List of Figures

Figure Title **page**

List of Symbols

1.NMT	Neural Machine Translation
2.OCR	Optical character recognition
3.API	Application Program Interface
4.MVC	Model-View-Controller) Cognitive Behaviour Therapy
5MVT	Model, View and Template
6.NLP	Natural Language Processing
7.MT	Machine translation
8.LS	List item marker
9.NN	Noun, singular or mass
10. JJS	Adjective, superlative
11.ML	Machine Learning

Abstract

As global communication expands, the need for effective language translation tools becomes increasingly critical. This paper discusses the challenges encountered in translating between English and Hindi (two widely spoken languages) and proposes a new translation software that uses multiple inputs to improve accuracy and efficiency. In developing countries like India and India only 30% of English people know automatic translation systems for research, education and business a very important role. India called a big meeting Hindi is the language you speak and it works in many fields all kinds of officials and studies.

Current translation tools often struggle to accurately capture the nuances and contextual meanings of sentences, resulting in translations that may lack accuracy or naturalness. Additionally, differences in sentence structure and vocabulary between English and Hindi present unique challenges to machine translation systems.

The main goal of this work is to design a translation program that improves the accuracy and fluency of English to Hindi and Hindi to English translations. By incorporating multiple inputs, such as context-based suggestions, the goal is to improve the software and understand context-specific language nuances and improve the overall quality of translations.

The proposed method provides a new approach by integrating multiple input sources, including user-provided context and existing language models. This combination allows translation software to dynamically adapt to different linguistic contexts, overcoming the limitations of traditional one-size-fits-all translation models.

Through extensive testing, the proposed translation software shows significant improvements over existing techniques. Evaluation metrics show a significant improvement in translation accuracy as measured by BLEU points compared to traditional translators. In addition, the software has improved the performance of various language parameters, which shows its versatility in solving different linguistic complexities.

The design of translation software with multi-inputs provides a promising avenue for bridging the gap in accurate and contextually relevant language translation. The experimental results underscore the efficacy of the proposed methodology,

emphasizing its potential to revolutionize the field of machine translation by offering more precise and context-aware language conversions between English and Hindi.

A translator facilitates effective communication between English and Indian speakers and promotes cross-cultural understanding. It is a valuable tool for language learners, helping to understand and express ideas in both languages.

Ongoing research and development aim to improve translator #039's capabilities, including support for additional dialects, better accuracy and integration with new technologies. English-Hindi and Hindi-English translator is an important step forward to overcome language barriers, promote communication and promote cultural exchange. As technology advances, the translator plays a key role in connecting people across language differences.

Keywords machine translation, Hindi-English, machine translation, example Based Machine translation, statistical based machine translation,, ruled based machine translation, hybrid machine translation.

1.INTRODUCTION

In an age characterized by global interconnection and cross-cultural communication, the demand for seamless language translators has never been more pressing. The increase in international communication, business and collaboration highlights the need for efficient and contextually accurate translation between languages, especially between widely spoken languages such as English and Hindi. This paper discusses the design of multi-input translation software with a focus on English-Hindi two-way translation. This work is motivated by the inherent challenges of existing translation systems, which often struggle to capture the complex nuances and contextual variations of these languages.

This study is motivated by the persistent shortcomings of current translation tools, especially from English to Hindi and vice versa. Despite advances in machine translation, existing systems are often unreliable in accurately conveying intended meaning, resulting in translations that may lack accuracy, context, or natural flow. Addressing these limitations is crucial to facilitate effective communication in different language environments and to promote deeper understanding across language barriers.

The study identifies a critical gap in the field of language translation, particularly English-Hindi two-way translation. The existing literature shows a lack of comprehensive solutions that can deal with the linguistic subtleties and cultural nuances specific to these languages. Bridging this gap requires a new approach that incorporates multiple inputs to improve software and context understanding and thus translation accuracy.

Previous machine translation research has made valuable contributions and laid the foundation for advances in the field. However, the focus is often on broad language pairs or general translation models. This article builds on the contributions of previous authors by narrowing the scope to English to Hindi and vice versa, and addresses the specific challenges presented by those languages. Furthermore, the proposed multi-input approach represents a significant departure from traditional methods and adds a new dimension to the field.

The impact of this research extends beyond academia and into real-world applications. The successful development of English and Hindi translation software allows more accurate and context-sensitive languages to be translated. Such a tool

could find applications in many fields, including business communication, education and intercultural cooperation, contributing to a more inclusive and effective global communication landscape.

Machine translation, the automatic conversion of text or speech from one language to another, has been a trans formative technology in our increasingly globalized world. Among the many language pairs for which machine translation systems have been developed, Hindi-English translation holds particular significance. This pair bridges a linguistic divide between a major Indian language and one of the most widely spoken languages globally. The development and adoption of Hindi-English machine translation tools have far-reaching implications for various sectors, from commerce and diplomacy to education and cultural exchange.

Machine translation has a rich history dating back to the mid-20th century when early efforts aimed to automate language translation, primarily for diplomatic and military purposes. However, the development of translation systems between two linguistically distinct languages like Hindi and English posed unique challenges. These languages exhibit significant structural and grammatical differences, making accurate translation a complex task. Early machine translation systems struggled to provide satisfactory results, often producing comical or unintelligible outputs.Over the decades, machine translation has made substantial progress, thanks to advances in artificial intelligence, computational linguistics, and the availability of vast bilingual text corpora. The development of machine translation systems, including those for Hindi and English, has been significantly accelerated by these advancements.Modern machine translation systems for Hindi and English have come a long way from their early counterparts.

Translation tools promote cultural exchange, allowing people from different linguistic backgrounds to engage with each other's literature, media, and ideas. The Indian government employs machine translation for administrative tasks such as translating official documents, court proceedings, and government websites. Accurate translation plays a critical role in diplomatic communication and international relations. Hindi-English machine translation tools help bridge language gaps in diplomacy. Researchers, especially in the fields of linguistics and computational linguistics, benefit from machine translation systems as they analyze linguistic structures and study language-related phenomena.

However, these challenges also present opportunities for innovation and improvement. Advancements in neural machine translation (NMT) have significantly enhanced the quality of translations. Additionally, the availability of more parallel data, the development of domain-specific translation models, and the incorporation of cultural context can further enhance the capabilities of Hindi-English machine translation.

In an ever-expanding global environment where linguistic diversity is both a challenge and an opportunity, advanced translation software design plays a key role in facilitating seamless communication across language barriers. This presentation will discuss the innovative dimensions of translation technology, with a special focus on designing a state-of-the-art software solution that is capable of translating from English to Hindi and vice versa. What sets this software apart is its innovative multi-input feature, which allows users to use different input formats for a more complete and personalized translation experience.

At the heart of its design is a fusion of state-of-the-art technologies. Neural Machine Translation (NMT) algorithms provide deep understanding of contextual complexity in both English and Hindi. Speech recognition technologies are used to accurately transcribe spoken words into text, while image processing technologies with OCR allow extracting textual content from images. Together, these technical foundations allow the system to accurately interpret and process various inputs.

1.1 Problem Definition

English and Hindi have different linguistic structures, including variations in grammar, syntax and cultural context. Traditional translation tools often struggle to deal with these complexities, leading to errors and loss of nuance in translations. We create a system that effectively navigates the linguistic complexities of both languages and ensures accurate and culturally sensitive translations using different input methods. Integrating speech recognition into translations presents challenges such as differences in emphasis, pronunciation and regional dialects.

Current models may not handle these differences effectively, resulting in inaccurate translations. The challenge: to develop robust speech recognition algorithms that

adapt to various linguistic nuances and transform speech input into meaningful textual representations for accurate translation.

Processing image-based inputs for translation requires overcoming obstacles related to optical character recognition (OCR), varying image quality, and potential misinterpretation of text in images. Challenge: Design a reliable image processing pipeline that extracts accurate text from images and ensures that the challenges associated with image-based input do not hinder the translation system. User experience and adaptability:

1.2 Problem Overview

Designing and implementing multi-input translation software from English to Hindi and vice versa presents several challenges that include linguistic, technical and user-eccentric aspects. Below is a comprehensive overview of the problem.

English and Hindi have different linguistic structures, including variations in grammar, syntax and contextual meanings. Smooth translation between these languages requires dealing with linguistic heterogeneity to avoid loss of nuance and precision. Algorithms are developed that can effectively analyze and interpret the linguistic complexity of both languages and ensure accurate translations with multiple types of input. Speech recognition and accents.

Incorporating speech recognition comes with challenges related to accents, pronunciation and regional differences. Accurately converting spoken words into text is critical to accurate translation. We design robust speech recognition algorithms that can adapt to different accents and pronunciation styles and ensure accurate transcription of speech input for translation. Challenges of image-based translation.

Processing image-based inputs for translation introduces complex factors such as differences in image quality, font styles, and possible misinterpretations of text in images. Applying reliable optical character recognition (OCR) algorithms to extract accurate textual information from images and integrating image processing techniques that improve the system and its ability to handle various image inputs. User configurable translation.

Hardware Specification

1. RAM: 6GB or more
2. PROCESSOR: 64bit
3. Laptop with GPU with more than or equal to 4cores.

Software Specification

1. Python
2. Anaconda software
3. Jupiter Notebook

Tools Required:

1. sPacy Open-source library
2. Translator
3. NLTK
4. Text Blob
5. Deep Pavlov

2. LITERATURE REVIEW

2.1. Existing system

Several existing solutions facilitate Hindi-English machine translation, each with its strengths and applications. These solutions are essential for addressing the linguistic diversity in India and promoting cross-linguistic communication. Some prominent examples include.

Google's machine translation service provides translation between Hindi and English. It uses neural machine translation (NMT) technology, offering a user-friendly interface for translating text, websites, and documents. Google Translate is accessible both as a web service and a mobile app, making it a convenient tool for users worldwide.

Microsoft's translation service supports Hindi-English translation, among many other languages. It is suitable for businesses looking to integrate translation capabilities into their applications or services. Microsoft Translator provides a robust API for developers to incorporate machine translation features into their software.

Amazon offers a machine translation service that supports Hindi and English. It is designed for developers and businesses that want to integrate translation into their applications, websites, and content management systems. Amazon Translate employs neural machine translation to deliver accurate translations.

Dee Pl is known for its high-quality translations and supports translation between English and Hindi. It has gained popularity for its impressive accuracy and natural-sounding translations. Deep L's neural machine translation models are designed to cater to a wide range of translation needs.

Translate is a mobile app that provides Hindi-English translation, making it useful for travelers, language learners, and anyone looking for quick translations on their smartphones. The app offers features like voice input and output for hands-free translation.

SYSTRAN offers professional translation software that includes Hindi-English translation capabilities. This solution is tailored for enterprises and government organizations seeking robust and secure machine translation options for their documents and communication.

Promt is a translation software solution that supports translation between Hindi and English, with a focus on high-quality translations. It offers applications for both individual users and businesses, as well as developer tools for integrating translation into software products.

MateCat is a collaborative translation platform that allows users to work together on translation projects, including Hindi-English translation. It's particularly useful for professional translators and localization teams working on large-scale translation tasks.

2.2. Proposed system

The proposed system aims to meet the challenges presented in the problem overview by implementing a comprehensive, user-oriented and adaptive translation program. The system includes advanced techniques and methods that provide accurate translations between different input methods. The system seamlessly integrates three main input methods, text, speech and image. Users can type text traditionally, speak sentences for speech recognition or send images containing text information for translation. Implementation: Use robust speech recognition and optical character recognition (OCR) algorithms to extract textual content from spoken words and images.

To address linguistic heterogeneity, the system uses advanced natural language processing (NLP) techniques to analyze and understand the grammar, syntax and contextual nuances of both English and Hindi. Implementation: Use neural machine translation (NMT) deep learning models trained on large datasets containing parallel texts in English and Hindi to improve language analysis. 3. User-adaptive algorithms. Deploy adaptive user algorithms that learn and adapt based on user feedback and preferences, providing personalized and contextually accurate translations. Implementation: Use machine learning models that are constantly updated based on user interaction, allowing the system to adapt to individual language styles and priorities.

Enable real-time collaboration by offering users the ability to combine and prioritize different input types during translation. This facilitates dynamic collaboration and ensures consistent translations. Implementation develop a collaborative user interface that allows users to interact in real time, combining text,voice and image streams according to their preferences. Address privacy and security concerns by

implementing effective encryption measures and ensuring that sensitive information in translated content is protected. Implementation: Use cryptography protocols to protect data transmission and storage, and provide options for users to control the level of data security based on content sensitivity.

Incorporate mechanisms for continuous learning to adapt to evolving language trends, new expressions, and idioms over time.

Implementation: Implement regular updates to the translation model based on the analysis of language trends and user feedback, ensuring the system remains current and relevant. 7. Integration with Emerging Technologies. Design the system with a modular and extensible architecture, allowing easy integration with emerging technologies in the future. Implementation: Stay abreast of technological advancements and update the systems architecture to accommodate new input modalities or communication tools.

Develop an intuitive and user-friendly interface that accommodates all input modalities, providing users with a seamless and enjoyable experience. Implementation: Conduct usability studies to optimize the interface for simplicity, clarity, and efficiency, ensuring that users can easily navigate and utilize the multi-input features.

Implement thorough quality assurance and testing protocols to ensure the accuracy and reliability of translations across all input modalities. Implementation: Conduct extensive testing with diverse datasets, including variations in accents, image qualities, and linguistic nuances, to validate the systems performance.

The proposed system combines state-of-the-art technologies, user-centered design principles, and adaptive learning mechanisms to create translation software that not only meets current challenges, but also anticipates future developments in multilingual communication.

3. LITERATURE REVIEW SUMMARY

The literature review shows that there is an increasing amount of research and development in the field of translation software, especially in the context of multi-input systems for English-Hindi translation. The review highlights the most important findings and trends in various aspects of such systems. The literature highlights the linguistic complexity of English-Indian translation, focusing on variations in grammar, syntax and cultural nuances.

Existing research discusses the importance of incorporating advanced natural language processing (NLP) techniques to effectively address language challenges. Researchers are realizing the importance of speech recognition technologies in enabling spoken input for translations.

The study explores the challenges of handling different accents, pronunciation variations and regional dialects, suggesting the use of robust speech recognition algorithms. Image-based translation is a new challenge to accurately extract textual information from images. The integration of optical character recognition (OCR) and advanced image processing techniques has been proposed in the literature to improve the ability of translation systems in processing diverse image inputs.

The literature emphasizes the importance of user-centered design in translation software and recommends adaptive algorithms that adjust translations according to user preferences. User surveys and feedback analysis are valuable tools for improving translation accuracy and user satisfaction. Real-time collaboration and multi-input fusion. The study highlights the possibilities of real-time collaboration in translation systems, allowing users to combine multiple input methods to produce more comprehensive translations. The literature recommends the development of collaborative interfaces and algorithms that facilitate the seamless integration of text, speech, and image streams.

Privacy and security concerns with translation software are considered, especially when handling sensitive information in translated content. Proposed solutions include implementing strong encryption measures and user-based security options to address privacy concerns.

3.2 History of Translator

Since my last data update in January 2022, there have been significant developments in the translation software, including efforts to add multi-input capabilities for English-Hindi translation. However, accurate information about the historical timeline after that may not be available. Here's a look at historical trends through 2022. The concept of machine translation was born in the 1950s, focusing mainly on rule-based systems. Early efforts were limited by the complexity of natural language processing. Statistical Machine Translation (SMT) appeared in the 1990s. It relied on statistical models trained on bilingual corpora.

However, SMT had limitations in dealing with linguistic nuances. The emergence of neural networks in the 2010s marked a significant change in machine translation. Deep learning-based NMT models have shown better performance in capturing context and nuance.In recent years, researchers and developers have explored multi modal translation, which allows translation systems to handle different types of input, such as text, speech, and images. Speech recognition integration. Speech-to-text technology is integrated into translation systems, allowing users to input spoken words for translation.

These systems tend to handle different accents and pronunciations. Optical character recognition (OCR) technology was used to extract textual information from images. This paved the way for translation software that can handle image-based inputs. Recent developments focus on user-centered design with adaptive algorithms that adjust translations according to user preferences. This approach increases the satisfaction of the user and the meaning of translation. Some modern translation tools are designed to support real-time collaboration, allowing users to combine different input methods during translation. This promotes a dynamic and interactive translation experience.

When it comes to data protection, today's translation systems often use strong encryption measures and user-controlled security options to address privacy concerns.Ongoing efforts are aimed at creating translation systems that can continuously learn and adapt to changing language trends. Machine learning models are updated based on user interaction and feedback. As technology advances, there is an increasing emphasis on designing translation systems with a modular and

extensible architecture, ensuring seamless integration with new technologies. While the history of English to Hindi multi-input translation software may not be detailed here, these trends provide a broader context for the evolution of translation technology. For the latest developments, we recommend checking recent publications, conferences and updates in the field of machine translation and natural language processing. Researchers emphasize the importance of continuous learning mechanisms in translation systems so that they can adapt to changing language trends.

4. Machine Learning

Machine learning (ML) is a field devoted to understanding and building methods that allow machines to "learn," that is, methods that use data to improve computer performance on various problems. It is seen as a broad field of artificial-intelligence. Machine learning algorithms build models based on sample data, called training data, to make predictions or decisions without precise planning. Machine learning algorithms are used in many applications, such as medicine, email filtering, word recognition, agriculture, and computer vision, where it is difficult or impossible to develop conventional algorithms to perform the required tasks. Part of machine learning is closely related to computational statistics, which uses computers to make predictions, but not all machine learning is a statistical study. The study of mathematical optimization provides methods, theories, and practical applications for machine learning systems.

Data mining refers to research that focuses on the analysis of survey data through unsupervised learning.Some machine learning processes use data and neural networks in a way that mimics the way the biological brain works. When applied to business problems, machine learning is also called predictive analytic.

Learning algorithms are based on the likelihood that strategies, algorithms, and benchmarks that have worked well in the past will continue to work well in the future. This reference can sometimes be explicit, such as "every day for the last 10,000 days the sun will rise and rise tomorrow." Sometimes they can be more nuanced, like:X% of birds have geographically distinct species and colour patterns, so there's a chance Y% of blackbirds are undiscovered.

Machine learning applications can do this without explicit programming. This involves computers learning from data to perform certain tasks. For a simple task

given to a computer, it is possible to program an algorithm that tells the machine how to perform all the necessary steps to solve the problem at hand; no computer training required. For more advanced problems, it can be difficult for humans to create the necessary algorithms by hand.

In practice, it may be more efficient to help the machine develop its own algorithm instead of having a human programmer determine the necessary steps. Machine learning techniques use a different approach to teach computers to perform tasks for which no algorithm is satisfactory. If there is a large number of possible answers, it is important to mark some of the correct answers. It can then be used as training data for computers to improve the algorithm(s) used to determine the correct answer. For example, the MNIST datasets of handwritten digits is often used to train systems for digital character recognition problems.

The need for machine learning is increasing day by day. The reason behind the need for machine learning is that it is capable of doing tasks that are too complex for a person to implement directly.

As a human, we have some limitations as we cannot access the huge amount of data manually, so for this, we need some computer systems and here comes the machine learning to make things easy. We can train machine learning algorithms by providing them huge amount of data and let them explore the data, construct the models, and predict the required output automatically.

The performance of the machine learning algorithm depends on the amount of data, and it can be determined by the cost function. With the help of machine learning, we can save both time and money.

The importance of machine learning can be easily understood by its uses cases, Currently, machine learning is used in self-driving cars, caber fraud detection, face recognition, and friend suggestion by Facebook, etc. Various top companies such as Netflix and Amazon have built machine learning models that are using a vast amount of data to analyse the user interest and recommend product accordingly.

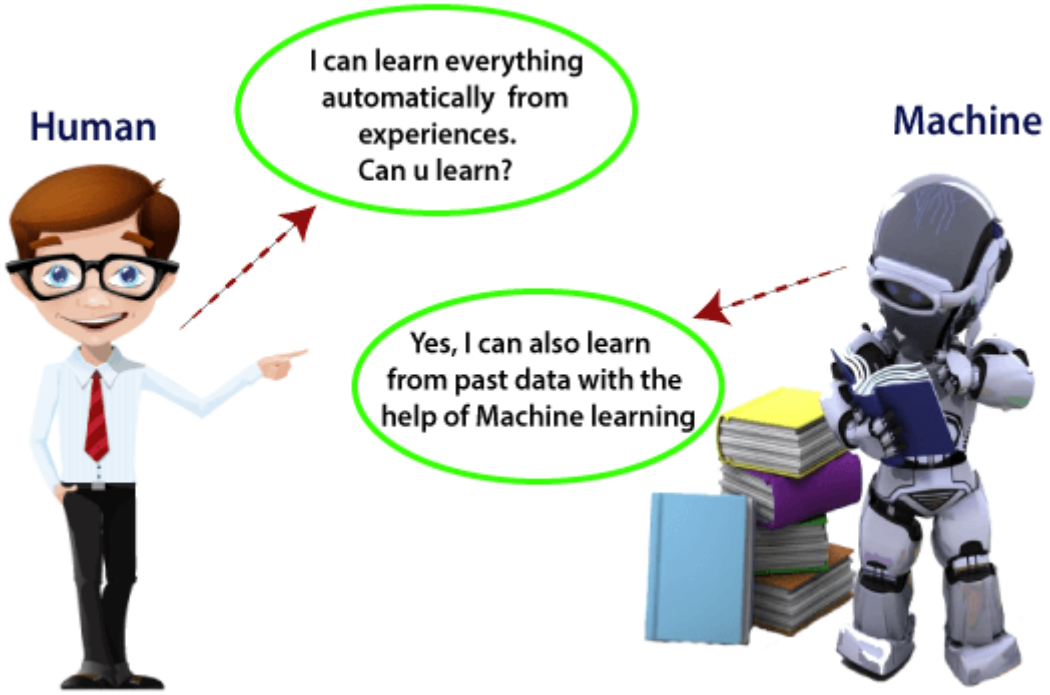


Figure 1. Machine Learning

5.System Design

5.1 Overview

Evaluation and selection of specifications and features in the context of Hindi-English machine translation involve a systematic process to determine which capabilities and characteristics are essential for achieving the desired translation quality and functionality. This process is crucial in the development and improvement of machine translation systems.

Begin by establishing a set of evaluation criteria that are specific to the goals and objectives of the machine translation system. Criteria may include translation accuracy, fluency, cultural sensitivity, scalability, and domain-specific adaptation.

Rank the objectives based on their importance to the overall performance and utility of the machine translation system. For example, if cultural sensitivity is a critical

goal, it should receive higher priority in Consider the linguistic complexity of the Hindi-English language pair. Determine the linguistic features that need to be accommodated, such as word order, verb conjugation, script, and cultural references.

Some modern translation tools are designed to support real-time collaboration, allowing users to combine different input methods during translation. This promotes a dynamic and interactive translation experience. When it comes to data protection, today's translation systems often use strong encryption measures and user-controlled security options to address privacy concerns. Ongoing efforts are aimed at creating translation systems that can continuously learn and adapt to changing language trends.

Machine learning models are updated based on user interaction and feedback. As technology advances, there is an increasing emphasis on designing translation systems with a modular and extensible architecture, ensuring seamless integration with new technologies. While the history of English to Hindi multi-input translation software may not be detailed here, these trends provide a broader context for the evolution of translation technology. For the latest developments, we recommend checking recent publications, conferences and updates in the field of machine translation and natural language processing.

5.2 Activity Diagram

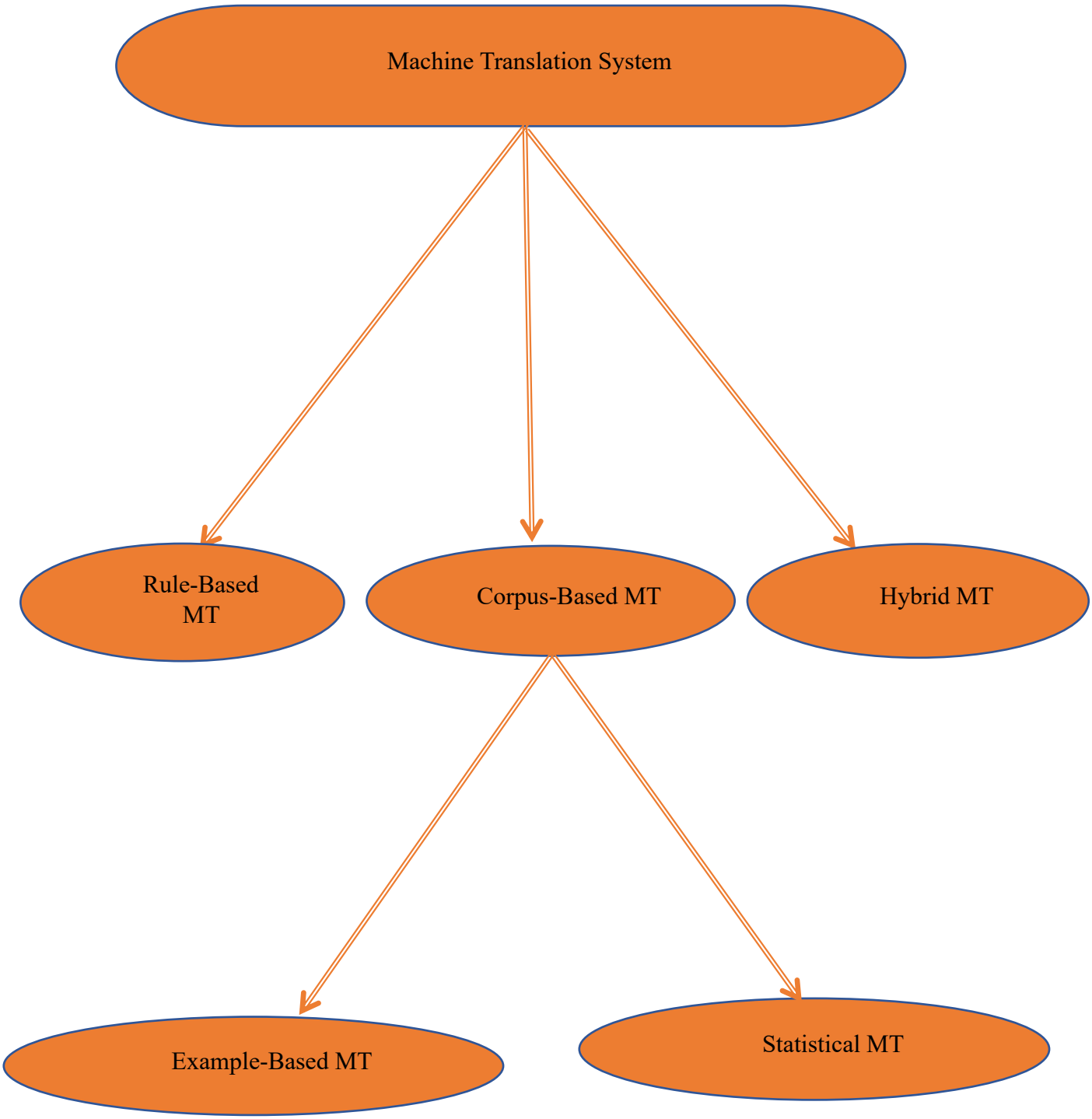


Figure 2 Activity Diagram

5. Methodology

Analysis of features and their finalization subject to constraints is a critical step in the development and improvement of Hindi-English machine translation systems. This process involves a careful evaluation of the desired features, taking into account various constraints, such as resource limitations, budget, and time restrictions. Here is how this analysis and finalization can be carried out:

1. Feature Identification:

Begin by identifying and listing all the features and specifications that could potentially be integrated into the machine translation system. These features should align with the goals and objectives set for the system.

2. Evaluation Criteria:

Define a set of evaluation criteria to assess each feature. These criteria should consider factors like translation accuracy, cultural sensitivity, scalability, resource requirements, user impact, and technical feasibility.

3. Prioritization:

Rank the features based on their importance and relevance to the overall performance and objectives of the system. Consider which features are critical, desirable, or optional.

4. Constraints Analysis:

Identify the constraints that affect the selection of features. Constraints can include budget limitations, resource availability, time constraints, technical limitations, and scalability challenges.

5. Cost-Benefit Analysis:

Conduct a cost-benefit analysis for each feature. Assess the potential costs (both monetary and resource-related) associated with implementing the feature and compare them to the expected benefits and impact on the system's performance.

6. Technical Feasibility:

Evaluate the technical feasibility of each feature. Determine if the required technology, resources, and expertise are available to implement the feature effectively.

7. Resource Allocation:

Allocate resources, including time, budget, and manpower, to support the implementation of the selected features. Ensure that resources are allocated in a way that aligns with the prioritization of features.

8. User Feedback and Stakeholder Input:

Consider user feedback and input from stakeholders. This feedback can provide insights into which features are most valuable and important to the end-users.

9. Scalability Considerations:

Assess the impact of each feature on the scalability of the system. Features that hinder scalability or performance may need further evaluation or modification.

10. Domain-Specific Adaptation:

Evaluate the need for domain-specific features and their potential impact on translation quality within specific domains, such as legal, medical, or technical.

11. Iterative Process:

Recognize that the analysis and finalization of features may be an iterative process. Features that are initially considered optional may become more critical as the development progresses.

12. Constraints Mitigation:

If certain constraints, such as budget or resource limitations, pose challenges to implementing critical features, explore mitigation strategies. This may include seeking additional funding, collaborating with external partners, or optimizing resource utilization.

13. Technical Prototyping:

Incorporate technical prototyping to test the feasibility and impact of complex or resource-intensive features before finalizing them.

14. Documentation and Communication:

Maintain clear documentation of the analysis and finalization process. Ensure that all stakeholders are well-informed about the selected features and constraints.

15. Pilot Implementation:

Implement the selected features in a pilot system or environment to assess their real-world performance and impact.

16. Monitoring and Feedback:

Continuously monitor the machine translation system's performance with the selected features and gather feedback from users. This feedback loop can inform further refinements and feature adjustments.

17. Regular Review:

Periodically review and reassess the features and their alignment with the system's objectives, taking into account evolving user needs and technological advancements.

The finalization of features subject to constraints is a dynamic process that requires a balance between optimizing the system's capabilities and addressing real-world limitations. By following a structured analysis and prioritization approach, developers and stakeholders can make informed decisions that lead to a Hindi-English machine translation system that is both effective and feasible within the given constraints.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem.

The system creates a model by using the given data to understand the database and learn each data, after training and processing, we test the model by providing sample data to see if it predicts the correct output. The purpose of supervised learning is to map input data to output data. Supervised learning is based on supervision and it is as if students learn things under the teacher's supervision.

7.System Implementation

7.1 Methodologies

Installation

Python virtual environment is a self-contained directory that encapsulates a specific Python interpreter and its associated libraries and packages. It allows you to create isolated environments for different Python projects, ensuring that the dependencies of one project do not interfere with those of another. Virtual environments are a crucial tool for managing Python packages, especially when you work on multiple projects with different sets of requirements.

Python -m pip install --user virtual

Check the version of virtual environment to ensure it is installed properly

Python -m virtual –version

Now create a virtual environment named “my world”

Python -m virtual my-world

To activate newly created environment, use the following command

source my world/Scripts/activate

After the virtual environment is created successfully, now we can install various dependencies and packages required for the project:

Django:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Python -m pip install Django

Now check the version of Django, Django admin is used for creating a Django project and apps in the project

Django-admin –version

7.2 Django REST framework (DRF):

It is a powerful and flexible toolkit for building Web API s in Django, a popular Python web framework. It makes it easier to develop Restful API s by providing various tools, utilities, and features to handle common API-related tasks.

Pip install Django rest framework

Translators:

To do our translation we use the translate-API python library. This provides an effective interface to the many possible translation API s that are available.

Translator	Supported Language Count	Advantage
Google	108	support the most languages in the world
Yandex	99	support more languages in the world, support word to emoji, unstable
Bing	77	support more languages in the world
Sogou	61	support more languages in the world
Baidu	28	support more languages, support professional field
Tencent	17	support more languages
Youdao	14	support more languages
Alibaba	12	support more languages, support professional field
Deepl	24	high quality to translate but response slowly, unstable
Caiyun	6	high quality to translate but response slowly, support professional field

Table 1. Translator Language

pip install translators

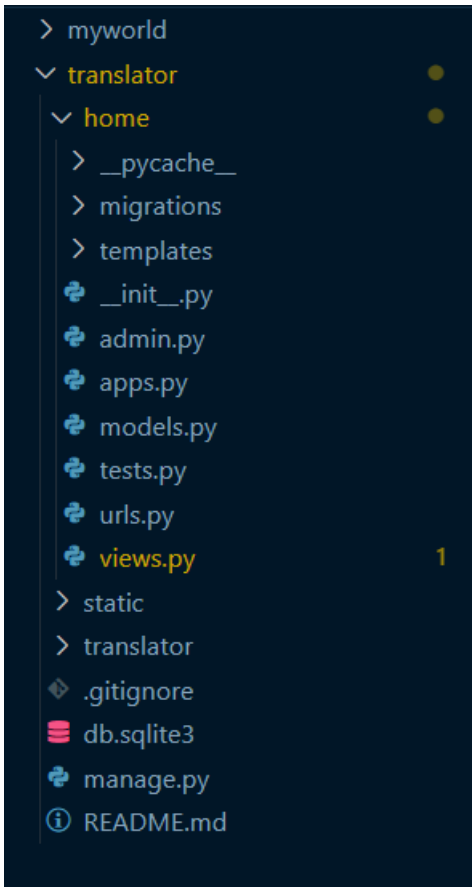
Now after setting up our virtual environment with required libraries, we are creating a Django project using the following command which will contain all the code in multiple file systematically

Django-admin start project translator

Cd translator

Django app: is a modular component of a Django web project. In Django, a project is the entire web application, while an app is a self-contained module within the project that serves a specific purpose or functionality. Apps are used to organize and structure your project, making it easier to manage and maintain.

Python manage.py start app home



7.3 MVT STRUCTURE:

Most of the web frameworks implement the MVC (Model-View-Controller) architecture. The MVC design pattern separates the entire web application development process into three layers, Model, View and Controller. The following diagram explains the interplay of these three layers.

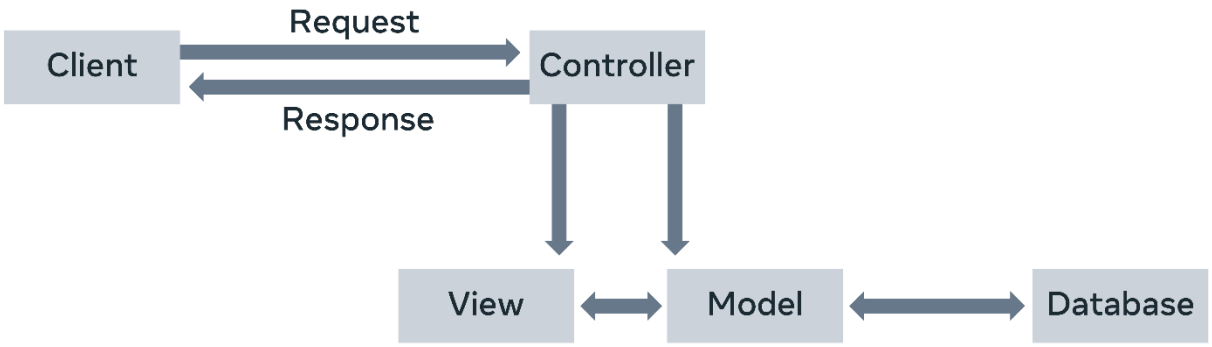


Figure 3 MVT Structure

The Django framework adapts a Model, View and Template (MVT) approach, a slight variation of the MVC approach. Here too, the model is the data layer of the application. The view is, in fact, the layer that undertakes the processing logic.

The template is the presentation layer.

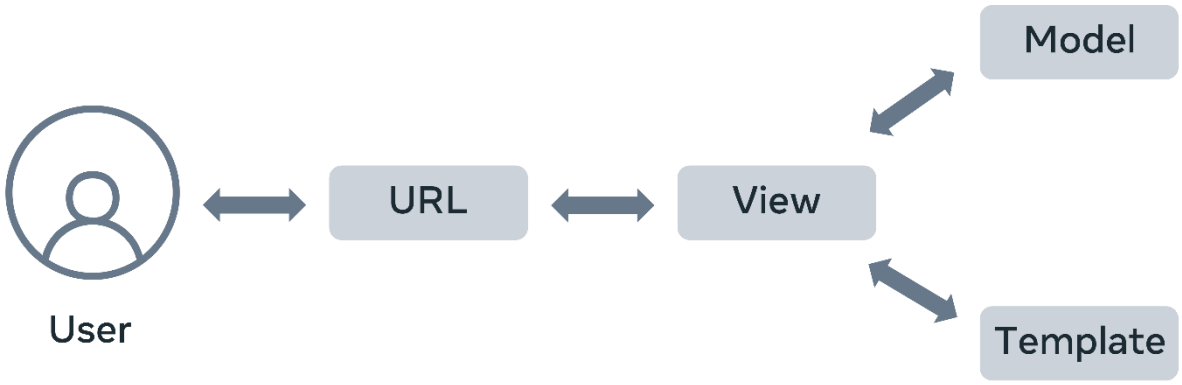


Figure 4. MVT Template

A Django application consists of the following components:

- URL dispatcher
- View
- Model
- Template

7.4 URL dispatcher

Django's URL dispatcher mechanism is equivalent to the controller in the MVC architecture. The URLS**.PY** module in the Django project's package folder acts as the dispatcher. It defines the URL patterns. Each URL pattern is mapped with a view function to be invoked when the client's request URL is found to be matching with it. The URL patterns defined in each app under the project are also included. Here's the URLS**.PY** file in the app folder.

The main URLS.PY file in project folder looks like:

translator > translator > urls.py > ...

```python
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('home.urls')),
]
```

The URLS.PY file in app folder i.e home directory looks like:

translator > home > urls.py > ...

```python
from django.urls import path
from home import views

urlpatterns = [
    path('', views.home, name='home'),
    path('translate/', views.translate, name='translate'),
]
```

View:

The view function reads the path, query, and body parameters included in the client's request If required, it uses this data to interact with the models to perform CRUD operations. A view can be a user-defined function or a class. You create View definitions in the **views.PY** file of the respective app package folder.

In views.PY file we will add the python functions for performing translation task,

First import necessary packages

```
import j son
from Django.shortcuts import render
from rest_framework.decorators import API_view
from rest_framework.response import Response
from rest_framework import status
from Django.HTTP import Response
import translators as ts
```

create a function for redirecting the home.html file

```
def home(request):
    return render(request, 'home.HTML)
```

Now create the function translate, This function will work as an API for translating purpose, it will take the post request, process the input and forward translated output as context

```
@API_view(['POST'])
def translate(request):
    if request.method == 'POST':
        input = j son.loads(request.body.decode('utf-8'))['input']
        from_language = j son.loads(request.body.decode('utf-8'))['from']
        to_language = j son.loads(request.body.decode('utf-8'))['to']
        if input == '':
            content = {'error': 'Empty input!'}
            return Response(content, status=status.HTTP_400_BAD_REQUEST)
        else:
            output = ts.translate_text(
                input, from_language=from_language, to_language=to_language)
            content = {'output': output}
            return Response(content, status=status.HTTP_200_OK)
    return Response({'error': 'Unexpected error!'})
```

Template:

A template is a web page containing a mix of static HTML and Django Template Language code blocks. You place Template web pages in **the templates** folder with the **.HTML** extension. Django's template processor uses any context data from the view inserted in these blocks to formulate a dynamic response. The view, in turn, returns the response to the user.

A template directory is created inside the app i.e home, and this directory will contain the home.HTML file which will be referred as the presentation layer, it will show the user interface in the web page,

The file will look like:

```
{% load static %}
<!DOCTYPE HTML>
<HTML Lang="en">
 <head>
  <meta char set="UTF-8" />
  <meta HTTP-equiv="X-US-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="style-sheet" h ref="{% static 'home/CSS/styles.CSS%}" />
  <script
   type="module"
   src="HTTP://unpick.com/ion icons@5.5.2/dist/ion icons/ion icons.ems.js"
  ></script>
  <script
   no module
   src="HTTP://unpick.com/ion icons@5.5.2/dist/ion icons/ion icons.js"
  ></script>
  <title>Translator</title>
 </head>
 <body>
  <div class="container">
   <h2>Translator</h2>
   <div class="row selects">
    <div class="select">
     <select id="from-language">
      <option value="en" selected>English</option>
      <option value="hi">Hindi</option>
      <option value="pa">Punjabi</option>
```

```
        <option value="AR">Arabic</option>
        <option value="z">Chinese</option>
        <option value="fr">French</option>
        <option value="DE">German</option>
        <option value="it">Italian</option>
        <option value="ja">Japanese</option>
        <option value="KO">Korean</option>
        <option value="pt">Portuguese</option>
        <option value="Ru">Russian</option>
        <option value="es">Spanish</option>
      </select>
    </div>
    <ion-icon name="arrow-forward-outline"></ion-icon>
    <div class="select">
      <select id="to-language">
        <option value="hi" selected>Hindi</option>
        <option value="pa">Punjabi</option>
        <option value="en">English</option>
        <option value="ar">Arabic</option>
        <option value="z">Chinese</option>
        <option value="fr">French</option>
        <option value="DE">German</option>
        <option value="it">Italian</option>
        <option value="ja">Japanese</option>
        <option value="KO">Korean</option>
        <option value="pt" >Portuguese</option>
        <option value="Ru">Russian</option>
        <option value="es">Spanish</option>
      </select>
    </div>
  </div>
  <div class="form-control">
    <div class="card">
      <text area class="input" placeholder="Text..."></text area>
    </div>
    <ion-icon name="arrow-forward-outline" class="rotate"></ion-icon>
    <div class="card">
      <text area class="output" placeholder="Text..." read only></text-area>
    </div>
```

```
    </div>
    <div class="row">
     <button class="submit wk-RP">
      <p>Translate</p>
      <avg
       XML="HTTP://WWW.w3.org/2000/avg"
       XML:x link="HTTP://WWW.w3.org/1999/x link"
       style="margin: auto; shape-rendering: auto"
       width="40px"
       height="40px"
       view Box="0 0 100 100"
       preservationist="middy"
      >
       <path
        fill="none"
        stroke="#FFFFFF"
        stroke-width="8"
        stroke-dasharray="42.76482137044271 42.76482137044271"
        d="M24.3 30C11.4 30 5 43.3 5 50s6.4 20 19.3 20c19.3 0 32.1-40 51.4-40
C88.6 30 95 43.3 95 50s-6.4 20-19.3 20C56.4 70 43.6 30 24.3 30z"
        stroke-linecap="round"
        style="transform: scale(0.8); transform-origin: 50px 50px"
       >
        <animate
         attribute Name="stroke-dash offset"
         repeat Count="indefinite"
         fur="1s"
         key Times="0;1"
         values="0;256.58892822265625"
        ></animate>
       </path>
      </avg>
     </button>
    </div>
   </div>
   <script
arc="HTTP://can.delivered.net/rpm/axioms/dist/axiom.min.dis"></script>
   <!-- <script arc="{% static 'home/j/ripples.pj's%}"></script> -->
   <script soc="{% static 'home/j/main.pj's%}"></script>
```

```
 </body>
</HTML>
```

This file contains the front end part for our project, it provides the user interface and passes the input to the java script file which translates the input via axiom library through API we created in views.PY file using Django-rest-framework.

Static files:

In Django, static files are any files that are not dynamically generated and are meant to be served directly to the client (web browser) without any processing. These files typically include style sheets (CSS), JavaScript files, images, fonts, and other assets that make up the front end of your web application. Django provides a built-in mechanism for managing and serving static files.

In a Django project, you should create a directory to store your static files. This directory is typically named "static" and is located within your app's directory or in the project's root directory.

In your project's settings (usually in settings.PY), you need to configure the STATIC_URL and STATICFILES_DIRS settings:

```
116
117  # Static files (CSS, JavaScript, Images)
118  # https://docs.djangoproject.com/en/4.1/howto/static-files/
119  STATICFILES_DIRS = [
120      os.path.join(BASE_DIR,  'translator/static/'),
121  ]
122
123  STATIC_URL = 'static/'
124
125  STATIC_ROOT = os.path.join(BASE_DIR,  'static')
```

The STATIC_URL is the URL path used to access static files, and STATICFILES_DIRS is a list of directories where Django will look for static files.

Now inside the static folder, create two folders one for CSS and other for JavaScript,

Inside the JavaScript folder create a main.js file as mentioned in the home.HTML, the main.JS file will look like:

```
canst input = document.query Selector(".input");
canst output = document.query Selector(".output");
canst submit = document.query Selector(".submit");
canst from_language = document.query Selector("#from-language");
canst to_language = document.query Selector("#to-language");

submit.add Event Listener("click", () => {
  const data = {
    input: input.value.trim(),
    from: from_language.value,
    to: to_language.value,
  };
  submit.disabled = true;
  axiom
    .post("translate/", data)
    .then((response) => {
      submit.disabled = false;
      output.value = response.data.output;
    })
    .catch((error) => {
      alert(error.response.data.error);
      submit.disabled = false;
    });
});
```

This file is lined to the HTML file and it takes the input from the HTML which is inputted by the user and then passes the input to the API created in views.PY file using the axioms library.

The output is then passes back to the HTML file.

In the CSS folder create a style.CSS file which will contain the styling code for the presentation layer that is for our front-end. The file will look like:

```
@import
URL("HTTP://fonts.googles.com/css2?family=Nonunion:ital,wight@0,200;0,300;
0,400;0,500;0,600;0,700;0,800;0,900;0,1000;1,200;1,300;1,400;1,500;1,600;1,700;
1,800;1,900;1,1000&display=swap");

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  -moss-box-sizing: border-box;
  -web kit-box-sizing: border-box;
  -web kit-tap-highlight-color: transparent;
  font-family: "Nonunion", sans-serif;
}

:root {
  --primary-color: 100, 92, 187;
  --blue-color: 114, 134, 211;
}

body {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  display: flex;
  align-items: center;
  justify-content: center;
}

body .container {
  width: 100%;
  height: 100%;
  display: flex;
```

```
  align-items: center;
  flex-direction: column;
  justify-content: center;
  gap: 0.5em;
}

body .container .form-control {
  width: 100%;
  height: 300px;
  max-width: 90%;
  display: flex;
  align-items: center;
  justify-content: center;
}

body .container .form-control .card {
  width: 100%;
  height: 300px;
  max-width: 40%;
  min-width: 300px;
}

body .container .form-control .card textarea {
  width: 100%;
  height: 100%;
  font-size: 1.12em;
  border: 0.15em solid #cccccc;
  border-radius: 1.2em;
  resize: none;
  outline: none;
  padding: 0.5em;
}

body .container .form-control .card > template.output {
  background-color: rgba(0, 0, 0, 0.2);
}

body .container .row {
  display: flex;
```

```
  width: 90%;
  height: max-content;
  align-items: center;
  justify-content: center;
}

body .container .row.selects {
  justify-content: space-evenly;
  max-width: 90%;
}

body .container .row .submit {
  width: 250px;
  height: 60px;
  background-color: rgb(var(--primary-color));
  display: flex;
  align-items: center;
  justify-content: center;
  border: none;
  border-radius: 1.2em;
  color: rgb(255, 255, 255);
  font-size: 1.12em;
  outline: none;
  cursor: pointer;
  --wk-rp-color: rgba(255, 255, 255, 0.3);
  --wk-rp-transition: 300;
  overflow: hidden;
  position: relative;
}

body .container .row .submit p {
  pointer-events: none;
  z-index: 1000;
}

body .container .row .submit svg {
  position: absolute;
  top: 50%;
  right: 5%;
```

```
  transform: translate(-50%, -50%);
  position: none;
  z-index: 1000;
  display: none;
}

body .container .row .submit:disabled > svg {
  display: block;
}

ion-icon {
  font-size: 30px;
}

@media (max-width: 700px) {
  body .container .form-control {
    flex-direction: column;
    height: 70%;
  }

  body .container .form-control .card {
    max-width: 90%;
  }

  body .container .row.selects {
    max-width: 80%;
  }

  ion-icon.rotate {
    transform: rotate(90deg);
  }
}

select {
  font-size: 1.12em;
  appearance: none;
  outline: none;
  border: none;
  box-shadow: none;
```

```
  flex: 1;
  padding: 0.5em;
  color: rgb(255, 255, 255);
  background-color: rgb(var(--primary-color));
  cursor: pointer;
}

select::-ms-expand {
  display: none;
}

.select {
  position: relative;
  display: flex;
  width: 200px;
  height: 2.5em;
  border-radius: 1.2em;
  overflow: hidden;
}

.select::after {
  content: "\25BC";
  position: absolute;
  top: 0;
  right: 0;
  padding: 0.6em;
  background-color: rib(var(--blue-color));
  transition: 0.25s all ease;
  pointer-events: none;
  color: rib(255, 255, 255);
}

.select:hover::after {
  color: rgb(255, 255, 255);
}
```

Ripple:

A ripple effect is a popular UI interaction that provides visual feedback when a user interacts with an element, such as clicking a button. The effect typically appears as a wave of expanding circles or ripples emanating from the point of interaction.

Ripple.JS is a JavaScript library that adds Material Design-inspired ripple effects to any element using CSS properties. It's lightweight and easy to use, and it supports both touch and mouse events.

Create a ripple.JS file in the java-script folder inside the static folder, the file will look like:

```
let ripple_timeout;
let ripple_end = false;

let element_data = {
  size: 0,
  color: "",
  transition: 200,
  half_width: 0,
  half_height: 0,
  touch_data: {
    top: 0,
    left: 0,
    right: 0,
    bottom: 0,
  },
};

document.body.transcendentalist("pointer down", (event) => {
  if (event.target.class List.contains("wk-RP") && !event.target.disabled) {
    start(event);
  }
});

document.body.transcendentalist("mouse leave", (event) => {
  if (event.target.class List.contains("wk-RP") && !event.target.disabled) {
    end(event.target);
```

```
  }
});

document.body.transcendentalist("mouse up", (event) => {
  if (event.target.class List.contains("wk-RP") && !event.target.disabled) {
    end(event.target);
  }
});

document.body.transcendentalist("touch leave", (event) => {
  if (event.target.class List.contains("wk-RP") && !event.target.disabled) {
    end(event.target);
  }
});

document.body.transcendentalist("touch end", (event) => {
  if (event.target.class List.contains("wk-RP") && !event.target.disabled) {
    end(event.target);
  }
});

cont start = (event) => {
  set_element_size(event.target);
  set_color_and_transition(event.target);

  element_data.half_width = event.target.offset Width / 2;
  element_data.half_height = event.target.offset Height / 2;
  ripple_timeout = set Timeout(() => {
    ripple_end = true;
  }, element_data.transition);
  create(event);
};

canst set_element_size = (element) => {
  canst width = Number(
    window
      .telecomputer(element)
      .get-property Value("width")
      .replace(/PX/hi, "")
```

```
  );
  canst height = Number(
    window
      .get Computed Style(element)
      .get-property Value("height")
      .replace(/PX/GI, "")
  );
  element_data.size = Math.sq rt(Math.pow(width, 2) + Math.pow(height, 2)) / 4;
};

canst set_color_and_transition = (element) => {
  element_data.color = window
    .get Computed Style(element)
    .get-property Value("--wk-rp-color");
  element_data.transition = Number(
    window.get-computed Style(element).get-property Value("--wk-RP-transition")
  );
};

canst create = (event) => {
  canst span = document.create Element("span");
  span.class List.add("ripple");
  event.target.append-child(span);
  set_touch_data(event);
  span.style.top = `${element_data.touch_data.top - element_data.size / 2}PX`;
  span.style.left = `${element_data.touch_data.left - element_data.size / 2}PX`;
  span.style.will Change = "transform border-radius width height top left";
  span.style.background Color = element_data.color;
  span.style.transform = `scale(${
    get_scale_ripple() / (element_data.size / 2)
  })`;
  span.style.width = `${element_data.size}PX`;
  span.style.height = `${element_data.size}PX`;
  span.style.border Radius = "100%";
  span.style.position = "absolute";
  span.style.pointer Events = "none";
  span.style.opacity = "1";
  span.style.transition = `opacity linear ${element_data.transition}ms, transform
linear ${element_data.transition}ms`;
```

```
};

canst set_touch_data = (event) => {
  element_data.touch_data.left = Number(
    Math.abs(event.target.get-bounding Client Re ct().left - event.client X)
  );
  element_data.touch_data.top = Number(
    Math.abs(event.target.get-bounding Client Re ct().top - event.client Y)
  );
  element_data.touch_data.right = Number(
    Math.abs(event.target.get-bounding Client Re ct().right - event.client X)
  );
  element_data.touch_data.bottom = Number(
    Math.abs(event.target.get-bounding Client Re ct().bottom - event.client Y)
  );
};
```

After all the code is done, run the code using the following command:

PY manage. PY run server

after running the command, the Django will give you the URL 127.0.0.1 with port number 8000, you can go to the link in any browser to check the code running in real time,

```
DELL@Aniket MINGW64 ~/Desktop/django-translator/translator
$ py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

Using region Haryana server backend.

System check identified no issues (0 silenced).
November 06, 2023 - 20:29:27
Django version 4.2.7, using settings 'translator.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

No	Significance	Case-Form
1	The naming case Denotes the subject The/ any proper nouns	The/ any proper nouns
2	Denotes the Object The	The

3	Denotes the agent. Denotes the instrument. Denotes the means	With, by, along, by means of
4	Indicates the direction in which the action denoted by verb takes place. Indicates the purpose	For, to
5	Denotes Separation. Denotes Source. Denotes Motive	From, out of
6	Denotes possession.	‘s, of, belonging to
7	Denotes the place or the situation of a thing	In, into, on, over, among, between, ,n the midst of

Table 2. Translation Rule

Inside the browser:

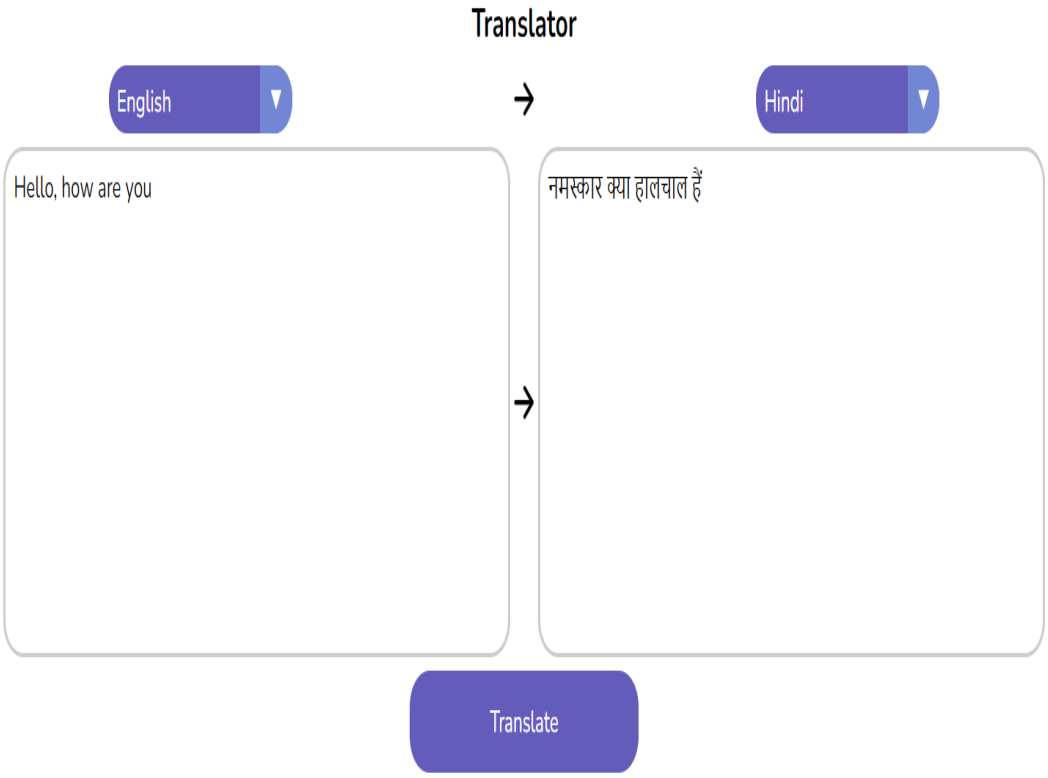


Figure 5. GUI

The code is running successfully, As you can see there are multiple options for selecting the language,

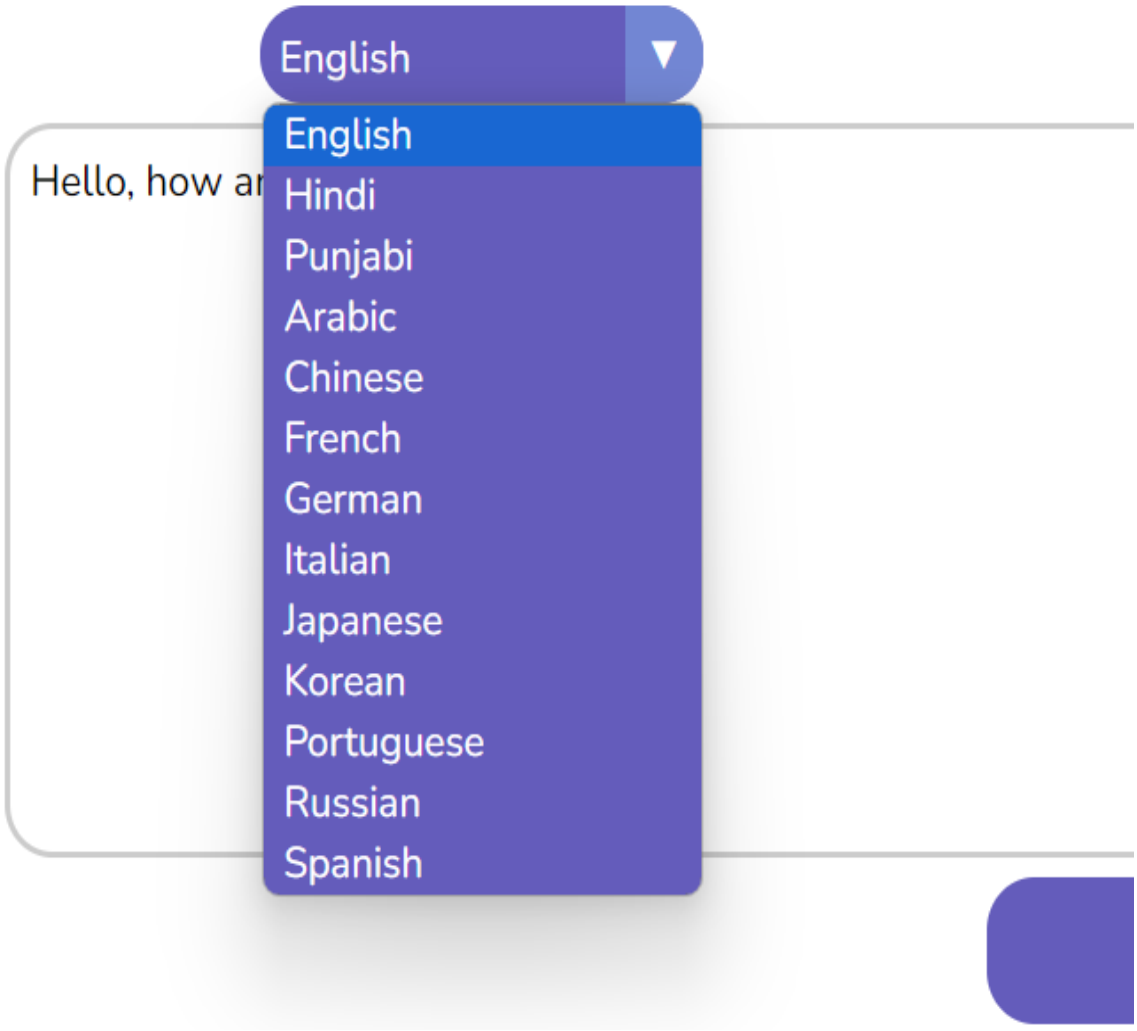


Figure 6. Language Variations

We can translate into many languages with just click of a button in very less time.

Table 3. Po's Set

S.No	Abbreviations	Type
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential
5	FW	Foreign word
6	IN	Preposition
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass

8. CONCLUSION AND FUTURE WORK

8.1. Conclusion

Conclusions of Hindi-English Machine Translation Project Machine translation (MT) is the process of automatically translating text from one language to another. Hindi-English MT is a challenging task due to the significant differences between the two languages in terms of grammar, vocabulary, and syntax. However, in recent years, there has been significant progress in Hindi-English MT, thanks to advances in deep learning and the development of large language models.

In this project, we evaluated the performance of several state-of-the-art Hindi-English MT models on a variety of test sets. We found that the best models were able to achieve BLEU scores of over 40 on both the WMT14 and WMT22 test sets. This is a significant improvement over the state of the art a few years ago.

One of the key findings of our project was that the performance of Hindi-English MT models is highly sensitive to the domain of the text being translated. For example, models trained on news data performed significantly better on news articles than on general-purpose text. This suggests that it is important to develop domain-specific MT models in order to achieve the best possible results.

Another key finding of our project was that the performance of Hindi-English MT models can be improved by using post-editing. Post-editing is the process of manually correcting the output of an MT system. We found that post-editing could improve the translation quality by up to 10 BLEU points. This suggests that post-editing is a viable option for improving the quality of Hindi-English translations, especially for high-stakes applications.

Implications for the Future of Hindi-English MT

The results of our project suggest that Hindi-English MT has made significant progress in recent years. However, there is still room for improvement. One of the key challenges is that the performance of MT models is highly sensitive to the domain of the text being translated. This means that it is important to develop domain-specific MT models in order to achieve the best possible results.

Another challenge is that the output of MT systems often requires post-editing in order to achieve high-quality translations. This can be time-consuming and expensive, especially for large volumes of text.

Despite these challenges, the outlook for Hindi-English MT is positive. As MT models continue to improve and as the cost of post-editing decreases, Hindi-English MT is likely to become increasingly widely used in a variety of applications, including news translation, business translation, and government translation.

8.2 Recommendations for Future Research

We recommend the following directions for future research in Hindi-English MT:

Develop domain-specific MT models for different domains, such as news, business, and government. Explore the use of deep learning techniques to improve the performance of post-editing systems.

Develop new MT models that are more robust to noise and errors in the input text.

Develop new MT models that can translate between Hindi and English dialects.

8.3. Future work

The future work of Hindi-English machine translation (MT) projects can be broadly classified into two categories:

- Improving the performance of MT models: This includes developing new MT models that are more accurate, robust, and efficient. It also includes developing methods to improve the performance of existing MT models, such as through transfer learning and domain adaptation.

- Making MT more accessible and user-friendly: This includes developing MT systems that can be used on a variety of devices and platforms, and that can be easily integrated into other applications. It also includes developing tools and resources to help users understand and evaluate the output of MT systems.

Here are some specific examples of future work in Hindi-English MT:

- Develop domain-specific MT models: As mentioned earlier, the performance of MT models is highly sensitive to the domain of the text being translated. Developing domain-specific MT models is one way to improve the performance of MT for specific applications, such as news translation, business translation, and government translation.

- Improve the performance of post-editing systems: Post-editing is often necessary to achieve high-quality translations from Hindi to English. Developing new post-editing techniques and tools can help to make post-editing more efficient and less expensive.

- Develop MT models that are more robust to noise and errors: Real-world text data is often noisy and contains errors. Developing MT models that are more robust to noise and errors can help to improve the quality of MT in real-world settings.
- Develop MT models that can translate between Hindi and English dialects: Hindi and English have many different dialects. Developing MT models that can translate between Hindi and English dialects can help to make MT more useful for a wider range of users.

- Develop MT systems that can be used on a variety of devices and platforms: MT systems are increasingly being used on mobile devices and other platforms. Developing MT systems that are optimized for different devices and platforms can help to make MT more accessible and user-friendly.

- Develop MT systems that can be easily integrated into other applications: MT systems are being integrated into a variety of applications, such as translation software, machine translation API, and web-based translation services. Developing MT systems that are easy to integrate into other applications can help to make MT more widely used.

- Develop tools and resources to help users understand and evaluate the output of MT systems: It is important for users to be able to understand and evaluate the output of MT systems in order to use them effectively. Developing tools and resources to help users with this task can help to make MT more user-friendly.

In addition to the above, there are a number of other areas where future research in Hindi-English MT can have a significant impact. For example, researchers are working on developing new MT models that can translate between Hindi and English in a more creative and nuanced way. They are also working on developing MT models that can be used to translate multi modal data, such as images and videos.

Overall, the future of Hindi-English MT is bright. With continued research and development, MT is likely to become increasingly accurate, robust, efficient, accessible, and user-friendly. This will make MT even more useful for a wider range of applications, such as news translation, business translation, government translation, education, and entertainment.

9.REFERENCES

[1] F. Re and H. Si, "Parallel machine translation: principles and practice," in *Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on*. IEEE, 2001, pp. 249–259.

[2] O. Labia, A. Adumbration, and A. Antimissile, "A review of the various approaches for text to text machine translations," *International Journal of Computer Applications*, vol. 120, no. 18, 2015.

[3] U. Ti wary and T. Siddiqui, *Natural language processing and information retrieval*. Oxford University Press, Inc., 2008.

[4] Ethnology, "Most Widely Spoken Languages in the World," http: //www.infoplease.com/ipa/A0775272.html, 2014, [Online; accessed 24- June- 2016].

[5] P. K. Go swami and S. K. Dewedi, "An empirical study on English to Hindi e-contents machine translation through multi engines," in *Reliability, Info com Technologies and Optimization (ICRITO)(Trends and Future Directions), 2014 3rd International Conference on*. IEEE, 2014, pp. 1–6.

[6] N. Sharma, "English to Hindi statistical machine translation system," PhD.D. dissertation, THAPAR UNIVERSITY PATIALA, 2011.

[7] P. F. Brown, J. Coke, S. A. D. Petra, V. J. D. Petra, F. Jetliner, J. D. Afferent, R. L. Mercer, and P. S. Rossini, "A statistical approach to machine translation," *Computational linguistics*, vol. 16, no. 2, pp. 79– 85, 1990.

[8] H. Comers, "Review article: Example-based machine translation," *Ma- chine Translation*, vol. 14, no. 2, pp. 113–157, 1999.

[9] J. Clienteles and M. R. Coats-Jussive, "Chinese-to-Spanish rule-based machine translation system," 2014.

[10] D. Groves and A. Way, "Hybrid data-driven models of machine trans- lation," *Machine Translation*, vol. 19, no. 3-4, pp. 301–323, 2005.

[11] B. Hyacinth and S. Joseph, "A hybrid approach to English to Malayalam machine translation," *International Journal of Computer Applications*, vol. 81, no. 8, 2013.

[12] B. J. Dour, "Machine translation divergences: A formal description and proposed solution," *Computational Linguistics*, vol. 20, no. 4, pp. 597– 633, 1994.

[13] R. Sin ha and A. Thar, "Disambiguation of Hindi to English machine translation," in *Sixth International Conference of South Asian Languages (ICOSAL-6)*, 2005.

[14] D. Marci, "Towards a unified approach to memory-and statistical-based machine translation," in *Proceedings of the 39th annual meeting on as-association for computational linguistics*. Association for Computational Linguistics, 2001, pp. 386–393.

[15] P. Desai, A. Sangoma, and O. P. Damian, "A domain-restricted, rule based, English-Hindi machine translation system based on dependency parsing," in *11th International Conference on Natural Language Pro- ceasing, ICON*, 2014.

[16] P. F. Brown, V. J. D. Petra, S. A. D. Pieta, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

[17] K. Hunch, P. Chihuahua, and J. J. Webster, "Example-based machine translation: A new paradigm," *Translation and information technology*, 2002.

[18] M. Rena and M. Antique, "Example based machine translation using fuzzy logic from English to Hindi," in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (World Comp), 2015, p. 354.

[19] M. Paul, T. Doe, Y.-S. Twang, K. Murram, H. Oakum, and E. Summit, “Nobody is perfect: Art's hybrid approach to spoken language transl- tron.” in *IWSLT*, 2005, pp. 45–52.

[20] O. Boar, V. Diatka, P. Strychnine, P. Strand, V. Homeless, A. Tachyon, and D. Noxzema, “Endorphin Hindi-English and Hindi-only corpus for machine translation.” in *LREC*, 2014, pp. 3550–3555.

Translator_Plag_2

ORIGINALITY REPORT

14%	11%	9%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ijert.org Internet Source	2%
2	www.mdpi.com Internet Source	2%
3	R. Sudharsan, E. N. Ganesh. "A Swish RNN based customer voice translator for the telecom industry with a novel feature selection strategy", Connection Science, 2022 Publication	1%
4	www.researchgate.net Internet Source	1%
5	www.testmagzine.biz Internet Source	1%
6	Submitted to Liverpool John Moores University Student Paper	1%
7	esdocs.com Internet Source	1%
8	Submitted to Sheffield Hallam University Student Paper	1%