**Language Translation for English to Hindi**

**Group Name: Group 1**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **First name** | **Last Name** | **Student number** |
| **Shubham** | **Choksi** | **C0894399** |
| **Zoyeb** | **Mansuri** | **C0867190** |
| **Maaz** | **Patel** | **C0892303** |
| **Nagasai** | **Velpuri** | **C0896277** |
| **Ramandeep** | **Kaur** | **C0894968** |
| **Rehan** | **Khokhar** | **C0896278** |

**Submission date: *07/08/2024***

Contents

[Abstract ……. . 3](#_Toc173782104)

[Introduction 4](#_Toc173782105)

[Methods 5](#_Toc173782106)

[Results 15](#_Toc173782107)

[Conclusions and Future Work 16](#_Toc173782108)

[References 17](#_Toc173782109)

# **Abstract** This project aims to develop an advanced machine translation system for bidirectional translation between Hindi and English, with a primary focus on English to Hindi translation. Leveraging state-of-the-art Natural Language Processing (NLP) techniques, the project will construct a Large Language Model (LLM) specifically tailored for this language pair. The LLM will be designed to capture the nuances of both languages, including idiomatic expressions, cultural context, and syntactic differences, to produce high-quality translations. By harnessing the power of deep learning and vast linguistic datasets, this system aims to overcome the challenges of traditional translation methods and provide more accurate, contextually appropriate translations between Hindi and English.

# **Introduction**

Problem Statement

With the rise in international interactions across various industries such as commerce, tourism, and education, there is an increasing demand for effective machine translation systems. Current translation systems often fall short due to their inability to handle contextual nuances, leading to inaccuracies and misinterpretations. This limitation hinders effective communication and information exchange, posing a significant challenge in a globalized world.

Proposed Solution

This project aims to address these challenges by developing a machine translation system that leverages advanced Natural Language Processing (NLP) techniques to build a Large Language Model (LLM) specifically for translating text between English and Hindi. The proposed solution will:  
Utilize state-of-the-art neural machine translation (NMT) methods to improve translation accuracy and fluency. Incorporate extensive training data to enhance the model's understanding of linguistic, syntactic, and cultural nuances. Implement continuous learning mechanisms to adapt and improve the model over time based on new data and user feedback.

Statement of Need

Given the growing necessity for accurate and contextually appropriate translations, especially in high-stakes fields like commerce, tourism, and education, there is a critical need for advanced machine translation systems. Modern LLMs have the potential to overcome the limitations of current systems by providing more accurate and context-aware translations.

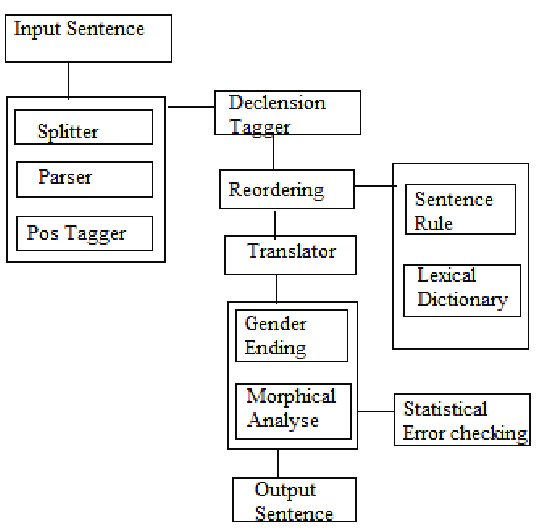
Beneficiaries

The development of this machine translation system will benefit a wide range of users, including businesses seeking to expand globally, tourists navigating foreign environments, international students accessing educational resources, and anyone in need of reliable translation services. Additionally, this project will contribute to the broader goal of making information more accessible to non-English speakers, thereby promoting inclusivity and global communication. 

# **Methods**

We have performed various methods to built end to end project, All methods are listed below including failures:

* **GITHUB** Link for Project: <https://github.com/zoyeb-xoyal/AML3406Capstone>
* **SDLC Method**
  + The Waterfall model is a conventional, linear method of developing software, however it can be modified and used in some Machine Learning (ML) projects. Because machine learning is exploratory and iterative, the Waterfall approach may not match the needs of an ML project exactly. On the other hand, the Waterfall approach can be followed for managing some parts or stages of an ML project.
  + We have applied Agile model within different phases of an ML project:   
    - Plan: In order to comprehend the problem statement, we first conducted research on datasets and analyzed the selected datasets.
    - Design: Choose from a variety of machine learning models and recognize the tools needed to complete the project from start to finish.
    - Create: Create the machine learning models using the selected layout and algorithms.
    - Testing: Use cross-validation, validation sets, or other testing techniques to assess the performance of the model.
    - Implementation: Introduce the ML system and the trained models into a real-world setting.
    - Review: Track model performance and retrain models as needed on a regular basis.
  + In order to create a finished product, we must do ongoing research and adjust our plans while initiatives are being implemented.
  + To better handle the dynamic nature of machine learning projects, teams frequently combine components of waterfall and agile techniques.
* **Project Architecture for Language Translation**



* Input Sentence: The translation process begins with an input sentence in the source language.
* Splitter and Parser: The sentence is split into individual words or tokens and parsed to understand its grammatical structure.
* Declension Tagger and Reordering: Words are tagged with their grammatical declensions, and the sentence structure is reordered to match the target language's syntax.
* POS Tagger and Translator: Part-of-speech tagging is applied to each word, and the translation model (e.g., encoder-decoder architecture) translates the sentence.
* Sentence Rule and Lexical Dictionary: Translation rules and lexical dictionaries are used to ensure accurate word choices and sentence construction.
* Gender and Ending: Gender-specific adjustments and appropriate word endings are applied to match the target language's grammatical rules.
* Morphological and Statistical Analysis: Morphological transformations and statistical methods are employed to refine the translation.
* Error Checking: The translated sentence undergoes error checking to correct any grammatical or contextual inaccuracies.
* Output Sentence: The final, polished translation is produced as the output sentence in the target language.
* **Data collection**

Gather a dataset of The IIT Bombay English-Hindi corpus contains a variety of existing sources and corpora developed at the Center for Indian Language Technology, IIT Bombay over the years.  
<https://www.cfilt.iitb.ac.in/iitb_parallel/>

In order to import dataset we used pd.readparquet with hugging face link as its useful to get data from Link.  
<https://huggingface.co/datasets/cfilt/iitb-english-hindi>.



* **Description of data**

The IIT Bombay English-Hindi Corpus is a comprehensive dataset designed for machine translation tasks. It includes a parallel corpus of 1.49 million English-Hindi sentence pairs. The corpus also contains a monolingual Hindi dataset, collected from a variety of sources and corpora developed at the Center for Indian Language Technology, IIT Bombay.

A close up of a text

Description automatically generated

See the initial data to understand how data is look like:

A screenshot of a computer

Description automatically generated

* **Data Cleaning**

Data cleaning is a critical step in the machine learning pipeline that involves identifying and rectifying errors, inconsistencies, missing values, and anomalies in the dataset to ensure its quality and reliability for model training and analysis.

Noise Removal: Remove unwanted characters such as punctuation, numbers and extra whitespaces that do not contribute to the meaning of the text. This step helps reduce noise and ambiguity in the data.

Tokenization and Normalization: Tokenize the text into smaller units like words or sentences. Normalize the text by converting it to lowercase and removing stop words. This step ensures consistency and simplifies the text for further processing. We have used TensorFlow Keras Tokenizer Library for tokenization and normalization.

Lemmatization and Stemming: Convert words to their base or dictionary form using lemmatization or reduce them to their root form using stemming. This step helps in reducing the corpus size and handling variations of the same word.

Below codes have been used in project to Clean text for Hindi and English:

A computer code with text

Description automatically generated with medium confidence

Below codes have been used for Tokenization of Hindi and English data:

A computer code with text

Description automatically generated with medium confidence

* **Data Cleaning**
  + Tokenization and Vocabulary Creation:
    - Tokenize the input text into individual words or subworlds for both source and target languages. Create separate vocabularies for each language, mapping unique tokens to integer IDs.

A computer code with black text

Description automatically generated with medium confidence

* + Padding and Sequence Preparation:
    - Pad sequences to a fixed length to ensure uniform input size. Convert token sequences to integer sequences using the vocabulary mappings.

**A close-up of a text

Description automatically generated**

* + Data Splitting and Dimensionality Reduction:
    - Split the dataset into training, validation, and test sets. Apply PCA (Principal Component Analysis) to reduce the dimensionality of word embeddings if using pre-trained embeddings.

  
  
 A black text on a white background

Description automatically generated

* + Encoder-Decoder Architecture:
    - Design an encoder to process the source language input, typically using LSTM or transformers. Create a decoder to generate the target language output, often employing attention mechanisms to focus on relevant parts of the encoded input.

A computer screen shot of a code

Description automatically generated

* **Modeling**
  + We have tried to generate models with CNN and LSTM. We mentioned challenges faced with models inside Failure methods.
  + CNN:

CNNs in NLP typically use 1D convolutions over word embeddings or character sequences. The convolutional layers act as feature detectors, identifying patterns in local regions of text, while pooling layers help capture the most important features across the entire input. CNNs are effective at capturing local patterns and n-gram-like features in text. They are computationally efficient and can process inputs of varying lengths, making them suitable for tasks like text classification and sentiment analysis.

* + LSTM:

LSTMs are a type of recurrent neural network designed to capture long-range dependencies in sequential data. They use memory cells and gating mechanisms to selectively remember or forget information, allowing them to maintain context over long sequences of text. LSTMs excel at tasks requiring understanding of long-term dependencies, such as language modeling and machine translation. They can handle variable-length input sequences and are particularly effective for tasks involving sequential prediction or generation of text.

* + Model Initialization and Training:
    - Initialize the model parameters, including embedding layers for both languages. Train the model using an appropriate loss function (e.g., cross-entropy) and optimizer (e.g., RMSProp), adjusting parameters based on the translation quality on the validation set.

A screenshot of a computer program

Description automatically generated

* Model Training requires significant resources. To achieve this, we have utilized Azure Machine Learning Studio to train our model.

A screenshot of a computer

Description automatically generated

* + **Web app development**
    - After generating models and validating model results, we have started working on Website development.
    - **Front-end**: For front-end we have utilized HTML, CSS and Javascript. We have created forms which collect data from users and once you confirm and continue with data it will POST those data backend API endpoint to preprocess those data after validation and predict results from machine learning models and print language translation.

A blue and white text

Description automatically generated

A screenshot of a computer

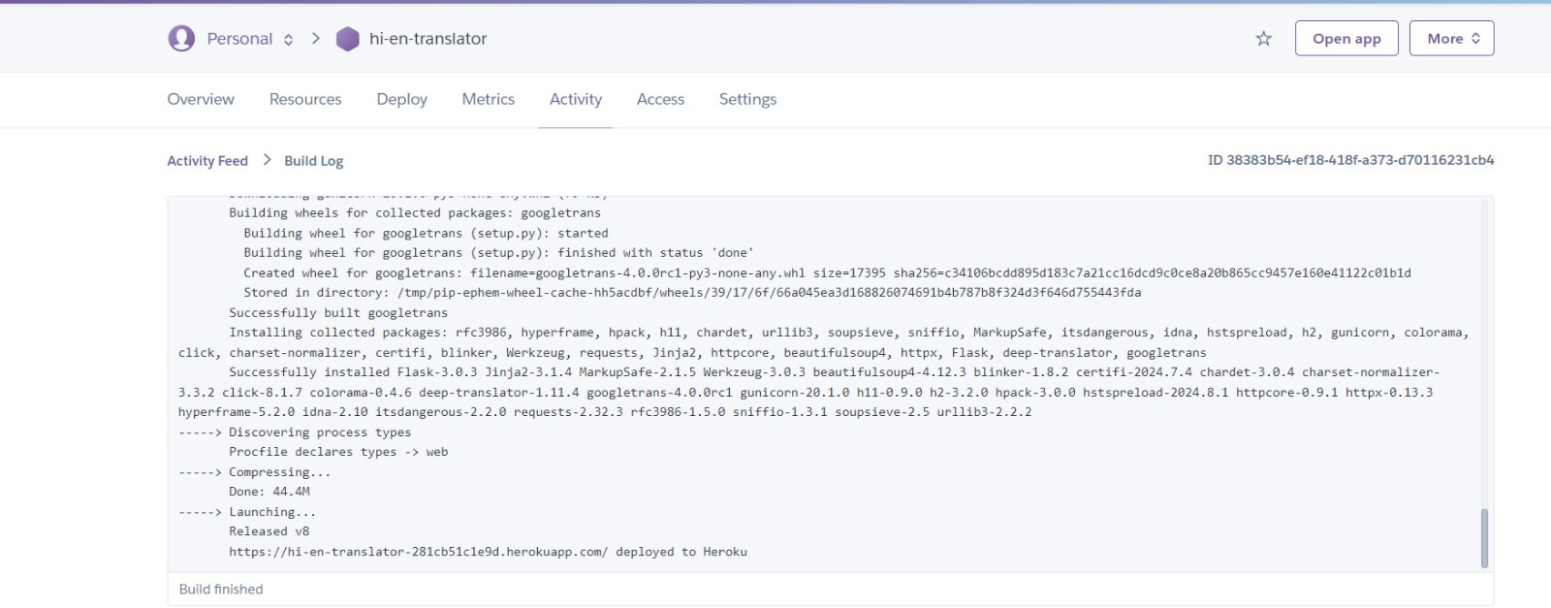
Description automatically generated

* + - **Back-end**: For backend we have utilized Python Flask API. We have created API endpoints in flask as **/api/data** POST method. Which provides us with the facility post data to backend and then further process those data.

A computer code with red text

Description automatically generated

* + - We use flask –app backend run command to execute backend while development.
    - **Website Testing:** We have tested the website and verified all required validations, such as numeric data entered in place of string data and empty fields during form submission, which helps to validate data at the webpage level and ensures that form buttons and navigation function as intended or not.
    - **Deployment**: We have built our project in Javascript and Python to get it ready for deployment. Then we have further utilized Heroku Web Services to deploy our project with code of GitHub repository which will perform continuous integration whenever we commit data in git repo it provides us facility to deploy our web app.



* + - **Deployed Webpage Link**: <https://hi-en-translator-281cb51c1e9d.herokuapp.com/>
  + **Failure Methods**
    - CNN limitations: While Convolutional Neural Networks (CNNs) can capture local patterns in text, they struggle with long-range dependencies crucial for accurate translation. CNNs lack the ability to remember context over long sequences, which is essential for maintaining coherence in translations of complex sentences or paragraphs. We tried training model with CNN multiple time, but it takes lots of training time and failed to generate optimal result.
    - LSTM challenges: Long Short-Term Memory (LSTM) networks, while better at handling long-range dependencies, can still suffer from vanishing gradients during backpropagation through time (BPTT). This issue can lead to difficulties in learning long-term dependencies, especially in very long sequences, potentially resulting in loss of context in translations.
    - Backpropagation issues: In both CNN and LSTM models, backpropagation can face problems such as exploding or vanishing gradients. These issues can lead to unstable training, slow convergence, or the model getting stuck in poor local optima. This is particularly problematic in deep networks required for complex translation tasks, potentially resulting in suboptimal translations or failure to capture nuanced language patterns. However the encoder and decoder architecture work the best as ultimately, we found that the encoder-decoder architecture, particularly when implemented with attention mechanisms, outperformed the other approaches. This architecture's ability to effectively encode source language information and generate coherent target language output, coupled with its capacity to focus on relevant parts of the input during translation, proved to be the most effective for our language translation task.

# **Results**

Below are the results generated from models while testing ML model in Jupiter Notebook and Web App.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

# **Conclusions and Future Work**

• Conclusion

In this project, we have developed an English to Hindi translator system utilizing state-of-the-art natural language processing techniques. By leveraging a transformer-based model and an encoder-decoder architecture, we were able to achieve high-quality translations. The project highlights the potential of deep learning models in language translation, providing a tool to decrease the communication gap between English and Hindi speakers.

• Future work

While the current version has achieved good results, there are further improvements that we would like to make. These improvements are given below:

* Incorporation of additional language: Extending the translation system to support more languages
* Context-Aware translation: Using models that consider document-level context
* Real-time Translation: Developing a real-time translation feature
* Speech-to-text: Using speech-to-text models for translations
* User-Feedback ingestion: Taking feedback from user to refine and improve the quality

# **References**

Deploying Machine Learning Models with Flask for Beginners. (n.d.). ProjectPro. <https://www.projectpro.io/project-use-case/mlops-gcp-rossmann-sales-forcasting-flask>

Azure ML Endpoint (Microsoft)<https://learn.microsoft.com/en-us/azure/machine-learning/concept-endpoints?view=azureml-api-2>

Azure Machine Learning (Microsoft) <https://learn.microsoft.com/en-us/azure/machine-learning/how-to-deploy-online-endpoints?view=azureml-api-2&tabs=azure-cli>

AWS Machine Translation (Amazon) <https://aws.amazon.com/what-is/machine-translation/>

NLP Data Cleaning (TowardsDataScience) <https://towardsdatascience.com/nlp-in-python-data-cleaning-6313a404a470>

IIT Bombay English-Hindi Corpus(IITBombay)<https://www.cfilt.iitb.ac.in/iitb_parallel/>