Final Project Report - SE22UARI087

Team16-Hinglish Autosuggestions

Introduction:

The goal of this project is to develop a next-word suggestion system specifically for Hinglish, a mix of Hindi and English commonly used in everyday conversations. The challenge is to create a model that can accurately predict the next word while handling the unique nature of Hinglish, including code-switching and informal communication. This report outlines the **data analysis**, **model selection**, and **experimentation** process. I explored NLP techniques like **bigrams**, **DistilBERT**, and **TinyBERT**. Additionally, I experimented with **TinyBERT** to optimize processing speed and reduce model size, although GPU limitations in Colab prevented me from fully running and fine-tuning it. The aim was to build an effective next-word suggestion system that captures both the context and the nuances of Hinglish language use.

Motivation:

I chose this project to tackle the challenge of processing Hinglish, a hybrid language that blends Hindi and English, commonly used in digital communication. Traditional language models struggle with this code-switched language, and I wanted to build a system that could accurately predict the next word in Hinglish sentences. By developing this auto-suggestion system, I aim to improve communication on digital platforms, making interactions smoother and more intuitive for users who frequently switch between languages.

Prior Related Work:

Several studies and models have been developed to handle multilingual and code-switched language data. Approaches like Google's BERT and its variants have shown promise in understanding the context of mixed-language text, making them an ideal choice for Hinglish. Previous work has focused on adapting large language models like BERT for multilingual tasks, but applying them to Hinglish-specific datasets is still underexplored.

Dataset: (Dataset Link)

For this project, I used the **Hinglish** dataset from Hugging Face, which contains 189,000 rows in the training set and 2,070 rows in the validation set. The dataset consists of two columns:

- **en**: English text
- **hi_ng**: Hinglish text

For this project, I used only the **hi_ng** column.

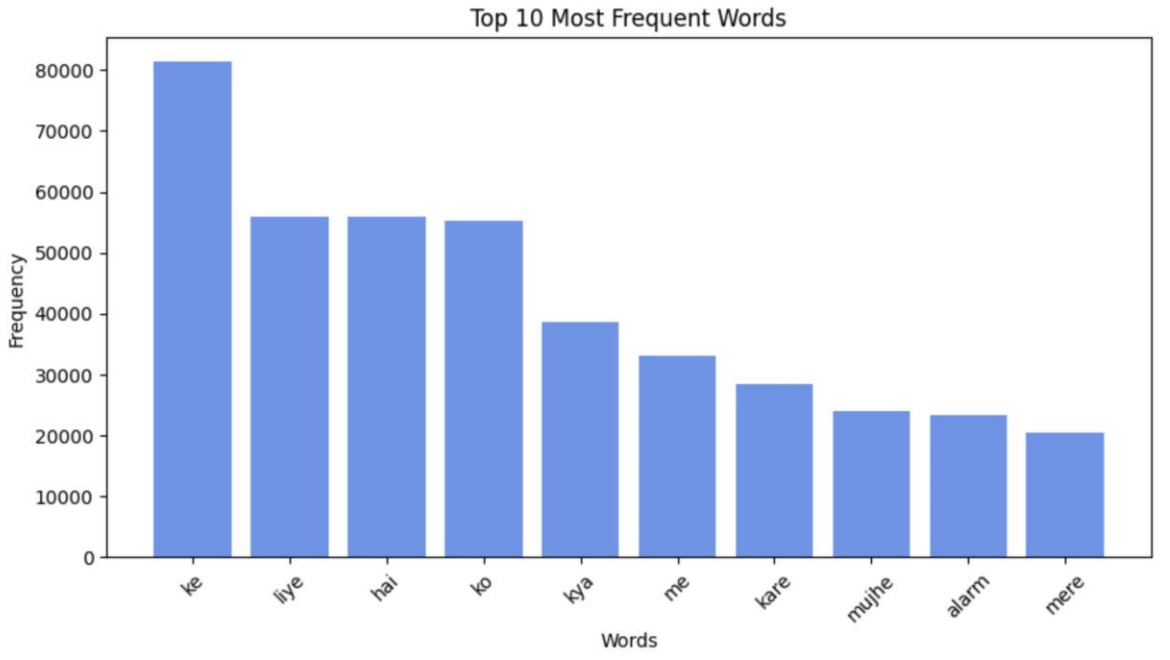
Methodology:

Data Analysis:

Data Analysis

I analyzed the Hinglish dataset to understand its structure and word distribution. Key steps included:

- **Preprocessing**: Cleaned text by removing URLs, special characters, and converting to lowercase.
- **Insights**: Generated a word cloud, identified the top 10 most common words, and calculated unique words to understand vocabulary size.

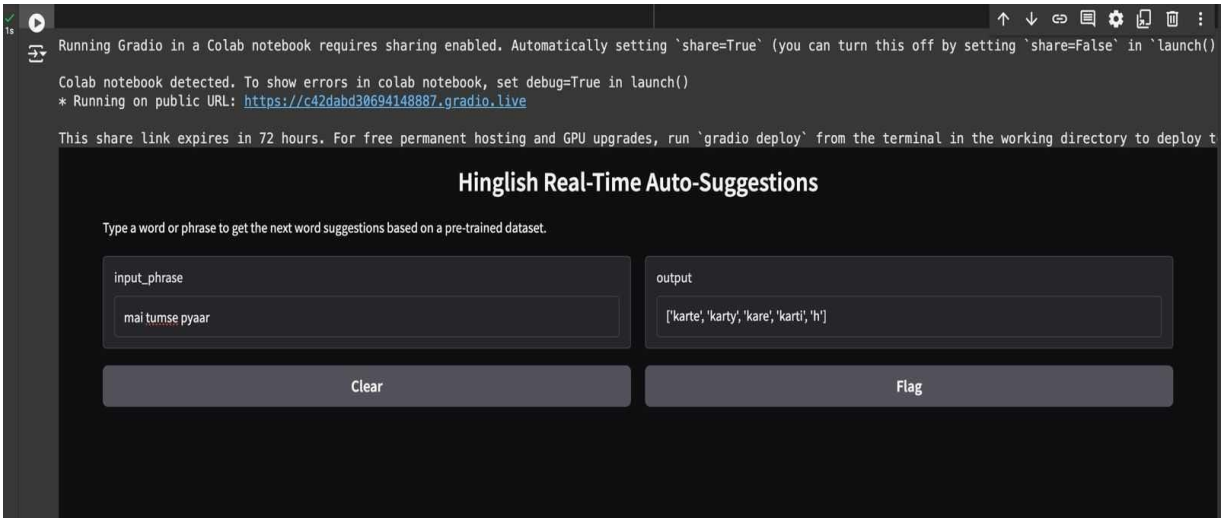


Bigram Model:

Bigram model

I implemented a **bigram-based approach and also used gradio for deployment** to predict the next word using pairs of consecutive words.

- **Strengths**: Simple and effective for short-term dependencies.
- **Limitations**: Struggles with capturing broader sentence context.



DistilBERT Model:

DistilBERT

To improve predictions, I fine-tuned **DistilBERT**, an efficient version of BERT, on the Hinglish dataset.

Benefits: Enabled context-aware suggestions through advanced transformer-based modeling.

Data Preparation

- **Tokenizer & Masking**: Used DistilBertTokenizer to preprocess sentences, applying a 15% masking probability to create training and validation datasets with a custom HinglishDataset class.
- **Data Loaders**: Batched datasets using DataLoader for efficient training.

Model Training

- Fine-tuned the **DistilBERT multilingual-cased model** using masked language modeling (MLM) on the Hinglish dataset.
- Configured **AdamW optimizer** (learning rate: 5e-5) with a linear scheduler and trained for three epochs, reducing loss from **1.77 (epoch 0)** to **0.232 (epoch 2)**.
- Utilized GPU when available for faster training.

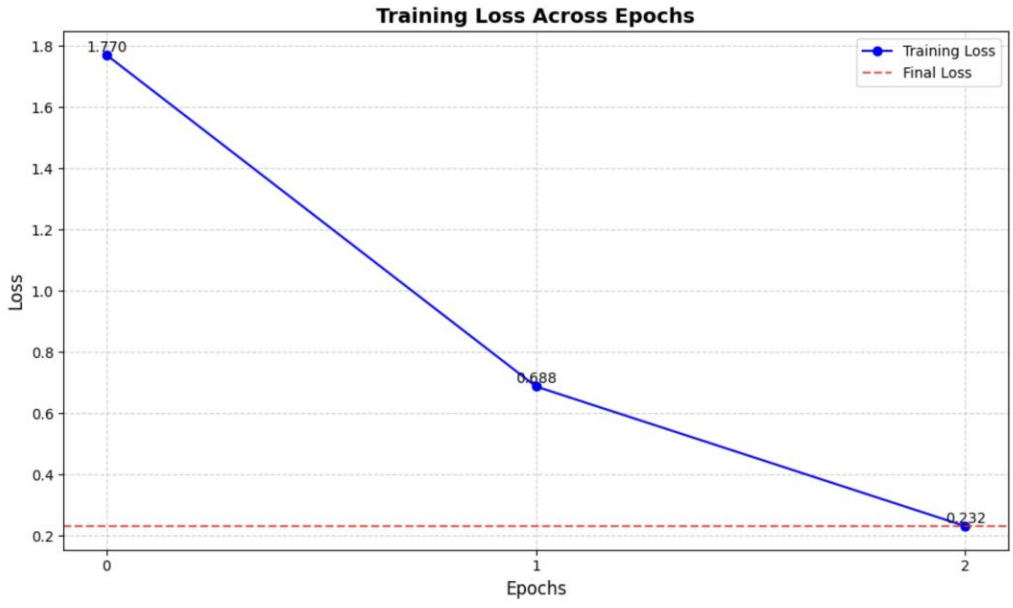
```
  0%|          | 0/11783 [00:00<?, ?it/s]We strongly recommend passing in an `attention_mask` since your input_ids may be padded. See https://huggingface.co/docs/transformers/troubleshooting#incorrect-output-when-padding-tokens-arent-masked.
Epoch 0: 100%|██████████| 11783/11783 [1:48:46<00:00,  1.81it/s, loss=1.77]
Epoch 1: 100%|██████████| 11783/11783 [1:48:56<00:00,  1.80it/s, loss=0.688]
Epoch 2: 100%|██████████| 11783/11783 [1:48:44<00:00,  1.81it/s, loss=0.232]
```

Evaluation and Prediction

- Created a **prediction pipeline** using the pipeline API for next-word suggestions. Users can input Hinglish text, and the model predicts the top five next words with confidence scores.

Visualization

- Plotted a graph showing the decline in training loss across epochs, indicating model convergence.



Experiments:

TinyBERT Experiment

- **TinyBERT**: Explored this lightweight model , but GPU constraints in Google Colab limited its full fine-tuning.
- **FuzzyWuzzy**: Integrated Fuzzy Matching to experiment with string similarity and alternative suggestions, particularly for handling typos or closely related Hinglish phrases.

Results:

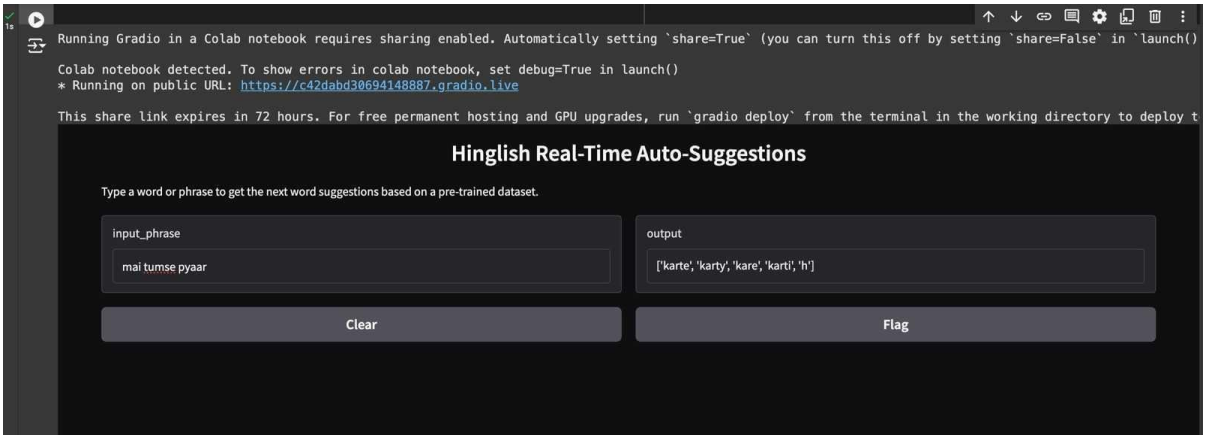
```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the `Pipeline` object. Model will be on CPU.
Hinglish Next-Word Prediction
Type a word or sentence to predict the next word.
Type 'exit' to quit.
Enter a word or sentence:  bahut cute

Predicted next words:
1. hai (score: 0.8065)
2. ho (score: 0.0552)
3. tha (score: 0.0168)
4. karo (score: 0.0147)
5. lagi (score: 0.0101)

Enter a word or sentence:  i want to go home

Predicted next words:
1. now (score: 0.3014)
2. soon (score: 0.2406)
3. late (score: 0.1275)
4. later (score: 0.0945)
5. today (score: 0.0877)
```

```
Enter a word or sentence:  ghar

Predicted next words:
1. tak (score: 0.2730)
2. traffic (score: 0.1983)
3. se (score: 0.0765)
4. station (score: 0.0648)
5. par (score: 0.0321)

Enter a word or sentence:  kya hua

Predicted next words:
1. hai (score: 0.6999)
2. he (score: 0.1484)
3. tha (score: 0.0729)
4. traffic (score: 0.0094)
5. ye (score: 0.0062)

Enter a word or sentence:  you seem so

Predicted next words:
1. well (score: 0.4585)
2. happy (score: 0.0450)
3. good (score: 0.0408)
4. great (score: 0.0330)
5. close (score: 0.0263)

Enter a word or sentence:  wapas

Predicted next words:
1. on (score: 0.1216)
2. play (score: 0.1191)
3. music (score: 0.0781)
4. song (score: 0.0747)
5. message (score: 0.0724)

Enter a word or sentence:  ek baar to meri

Predicted next words:
1. ma (score: 0.3560)
2. dil (score: 0.0163)
3. sir (score: 0.0123)
4. run (score: 0.0123)
5. maa (score: 0.0108)

Enter a word or sentence:  call

Predicted next words:
1. karo (score: 0.2822)
2. up (score: 0.2076)
3. me (score: 0.0941)
4. off (score: 0.0864)
5. out (score: 0.0671)

Enter a word or sentence:  exit
Exiting. Goodbye!
```

Analysis & Conclusion:

In this project, I developed a Hinglish next-word suggestion system using several NLP models. I began with a simple **Bigram model**, which was useful for short-term predictions but struggled with capturing longer dependencies. To improve accuracy, I fine-tuned **DistilBERT** on the Hinglish dataset. This significantly enhanced the model's ability to provide context-aware predictions. I trained the model using **masked language modeling**, where **15%** of the words in the sentence were randomly masked. I found that masking **15%** of the tokens worked well, as **higher masking rates led to increased computational costs** without much gain in performance.

Demo:



The **Bigram model** was integrated with Gradio, allowing real-time Hinglish next-word predictions.

In contrast, **DistilBERT** could take inputs directly from the user through the command line, as shown below.

```
Hardware accelerator e.g. GPU is available in the environment, but no `device` argument is passed to the
`Pipeline` object. Model will be on CPU.
Hinglish Next-Word Prediction
Type a word or sentence to predict the next word.
Type 'exit' to quit.
Enter a word or sentence:  bahut cute

Predicted next words:
1. hai (score: 0.8065)
2. ho (score: 0.0552)
3. tha (score: 0.0168)
4. karo (score: 0.0147)
5. lagi (score: 0.0101)

Enter a word or sentence:  i want to go home

Predicted next words:
1. now (score: 0.3014)
2. soon (score: 0.2406)
3. late (score: 0.1275)
4. later (score: 0.0945)
5. today (score: 0.0877)
```

Research Papers & Resources:

[BERT Research Paper]

[Are Multilingual Models Effective in Code-Switching? Research Paper]

[Transformer models and BERT model: Overview (Youtube Video)]

Acknowledgements:

Thanks to online resources, YouTube(Google tutorials), ChatGPT, and research papers for their valuable support in this project.

Special Thanks
I would like to express my gratitude to **Nidhi Mam fo**r her invaluable guidance and support throughout this project.