RESEARCH PAPER on AUTO SUGGESTION in HINGLISH

ABSTRACT INTRODUCTION:

Our project focuses on **Auto-Suggestion in Hinglish**, aiming to predict the next word based on the given input using advanced **Large Language Models (LLMs)** and **NLP** techniques. We primarily leverage **BERT models**, known for their encoder-only architecture, which excels in understanding contextual nuances.

PRIOR RELATED WORK:

1. ONLINE COURSES:

(29) Whitepaper Companion Podcast - Foundational LLMs & Text Generation - YouTube

This is the course I signed up for which basically tells you about the LLM.it really helped me gain so much knowledge on LLM.

2.From YOU TUBE:

(29) Finetuning LLM - YouTube

The above link is playlist where they teach us about how to fine tuning.

3.RESEARCH PAPER:

fine tuning

This research paper helped a lot for fine tuning. Like how many epochs and learning rate. And why and how to do the fine tuning.

4.KAGGLE NOTEBOOK

IndicBard - FineTuning

This is the source code I found from Kaggle. This has been really helpful when I don't how to proceed with fine tuning.

5.Sarcasm Detection System for Hinglish Language:

https://aclanthology.org/2021.ranlp srw.2.pdf 2. Analysis of a Hinglish ASR

6.System's Performance for Fraud Detection:
https://dl.acm.org/doi/abs/10.1007/978-3-031-48312-7_4

7. HinglishNLP: Fine-tuned Language Models for Hinglish Sentiment Detection: https://arxiv.org/abs/2008.09820

8. Cooking Is Creating Emotion: A Study on Hinglish Sentiments of Youtube Cookery Channels Using Semi-Supervised Approach: https://www.mdpi.com/2504-2289/3/3/37

9. Research paper on BERT:
(PDF) BERT: A Review of Applications in Natural Language Processing and Understanding

DATASET:

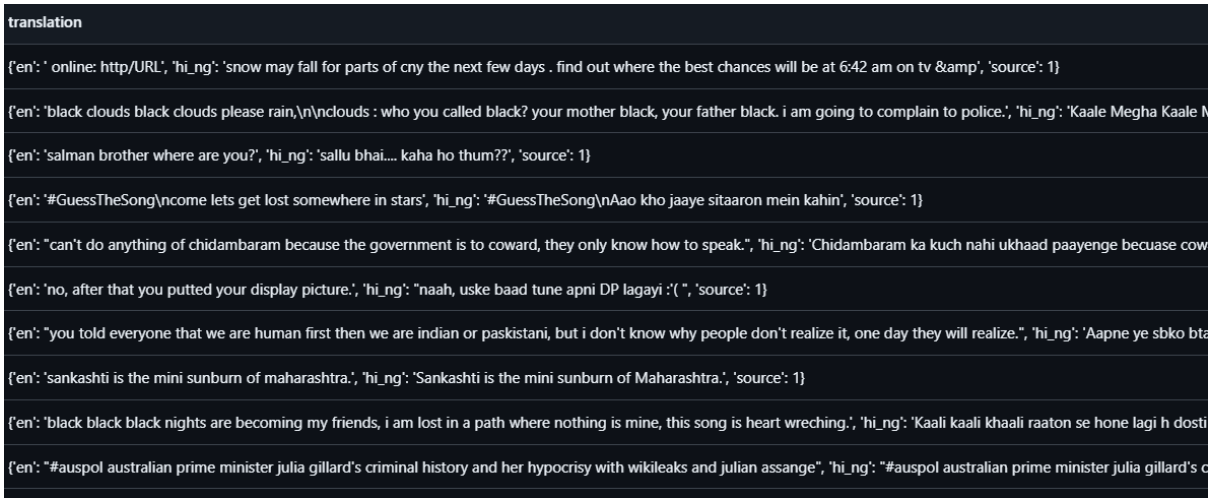
Training dataset:

Team16_Hinglish-Auto-suggestions/train.csv at main · harshithamadarapu/Team16_Hinglish-Auto-suggestions

Validation dataset:

Team16_Hinglish-Auto-suggestions/validation.csv at main · harshithamadarapu/Team16_Hinglish-Auto-suggestions

In the form of:



Methodology:

1. Exploratory Data Analysis (EDA): EDA revealed the structure and distribution of Hinglish phrases: ---Word Frequencies: Visualization by using CountVectorizer and word clouds. ---Text Lengths: The phrase length distribution is examined for consistency. ---n-Grams: Bigrams and trigrams are looked at for the occurrences to identify common contextual patterns.

2. Choice of Pre-trained Model:

---Multilingual BERT (mBERT): Fine-tuned for Hinglish text as it can adapt to varied multilingual data.

---DistilBERT: Selected based on a lightweight architecture allowing for both faster training and inference times with still satisfactory performance.

--- LSTM + DISTIL BERT :- as LSTM is very at finding the semantic meaning of the sentence and where as BERT is good at finding the contextual meaning of the sentence. So thought it would be good if we merge.

3. Evaluation Metrics: To evaluate model performance to predict the next word --- Cosine Similarity: Measures semantic similarity between embeddings. ---BLEU Score: Compares predictions to reference text using n-gram precision. ---Jaccard Similarity: Evaluates overlap between predicted and actual words. ---Training Loss ---Validation Loss

Cleaned text by removing URLs, special characters, and converting to lowercase.

Experiments:

1. LSTM+DISTIL BERT :
   Epochs : 20, learning rate : 3e - 5
   Perplexity: 5.8154
   Training Loss : 0.975

   Validation Loss :1.7605

   - TOKENIZATION with DistilBert Tokenizar

   - Trained the data with LSTM+DISTILBERT

INSIGHTS GAINED :-

1. **LSTM and DistilBERT Integration**: I've learned how to combine DistilBERT with LSTM to improve the model's ability to understand and generate text sequences for Hinglish auto-suggestion.
2. **Freezing Pre-trained Models**: Freezing DistilBERT's weights speeds up training and focuses on fine-tuning the LSTM layer.
3. **Tokenization**: I've used DistilBERT's tokenizer for efficient text preprocessing, converting raw text into model-friendly tokens.
4. **Data Sampling**: Experimenting with smaller datasets speeds up training but affects model performance.
5. **Mixed Precision Training**: Implementing mixed precision reduces memory usage and speeds up training.

   OUTPUT :
   Gives the output but there are errors and sometimes it is not predicting any more just giving the input text.

nlp_autosuggestion

this link leads to the code I have done.i have done this on KAGGLE NOTEBOOK.

2.DISTIL BERT (1st time):

Epochs: 3

Learning_rate=2e-5,

Training loss :0.007263169806896317

Validation loss': 2.798308429419194e-08

- TOKENIZATION with Distil Bert Tokenizer
- Training with Distil Bert Tokenizer

INSIGHTS :

2. **Model Training**: The model trained well, with a significant decrease in both training and validation loss. The evaluation loss is very low, indicating good model performance on the validation data.
3. **Prediction Issue**: The model predicted [PAD] instead of an expected word, which suggests potential issues with tokenization, input formatting, or the model's understanding of the data.
4. **Data Cleaning**: The cleaning function effectively removed unnecessary characters and alphanumeric words, ensuring that only relevant text was used for training.
5. **Next Word Prediction**: The next word prediction mechanism needs improvement, as the model is not correctly predicting the intended word. This could be due to tokenization issues or insufficient fine-tuning.

   Team16_Hinglish-Auto-suggestions/distill bert(train) .ipynb at main · harshithamadarapu/Team16_Hinglish-Auto-suggestions

FINE TUNING :

1. INDIC BERT:
   Learning_rate=2e-5
   Epochs=2
   training_loss= 18.448992131347655"

   Insights :

   Here are straightforward insights from the code:

1. **Loss Stagnation**: The loss remains constant across epochs, suggesting that the model isn't improving. This could be due to:
   - Too high or too low learning rate.
   - Insufficient or unrepresentative training data.
2. **Hyperparameter Tuning Needed**:
   - **Epochs**: Try increasing epochs (more than 2) for better results.
   - **Batch Size**: Adjust batch size for stable gradients.
3. **Model Choice**: IndicBERT may not perform well on Hinglish due to its formal language focus.

Team16_Hinglish-Auto-suggestions/FineTuneing(indic bert).ipynb at main · harshithamadarapu/Team16_Hinglish-Auto-suggestions

4.**Fine-Tuning mBERT** ---Fine-tuned on Hinglish dataset based on masked language modelling. ---Evaluated model performance by optimizing hyper-parameters for multitask adaptability with validation set.

5.Training DistilBERT (2nd time)---Trained DistilBERT for masked language modelling with consideration of computational cost. ---Also, dynamic masking during training were used to simulate realistic text prediction use cases.

The training loss was gradually reduced from 1.5188 to 1.2805, and the validation loss was reduced from 3.9386 to 3.5305, which shows effective learning and generalization.

6.Loss Monitoring ---Both models are trained and validated with a concern for reduced training and validation losses. ---Loss trends have been analyzed for proper generalization and not overfitting, which was needed at times. Early stopping has been used.

7.mBERT: The training loss reduced from 1.5359 to 1.1049. However, there were NaN values in the validation loss at Epochs 1 and 3, indicating issues with validation evaluation. Similarity Metrics: Jaccard Similarity: 0.1891 Average Cosine Similarity: 0.0329

INSIGHTS :

The Jaccard Similarity and Cosine Similarity scores are low, meaning that there is limited lexical and semantic overlap between the predicted and actual words. This implies that both models are weak in capturing contextual relevance for Hinglish phrases. DistilBERT had stable learning, whereas mBERT had validation issues. Further improvements are needed in both models to capture semantic meaning and context in Hinglish auto-suggestions.

8. Training Configuration

- Loss Function: Categorical cross-entropy to handle the multi-class output.
- Optimizer: Adam optimizer with a learning rate of 0.001.

The training process included hyperparameter tuning to achieve the best possible results on the validation dataset.

Training Configuration

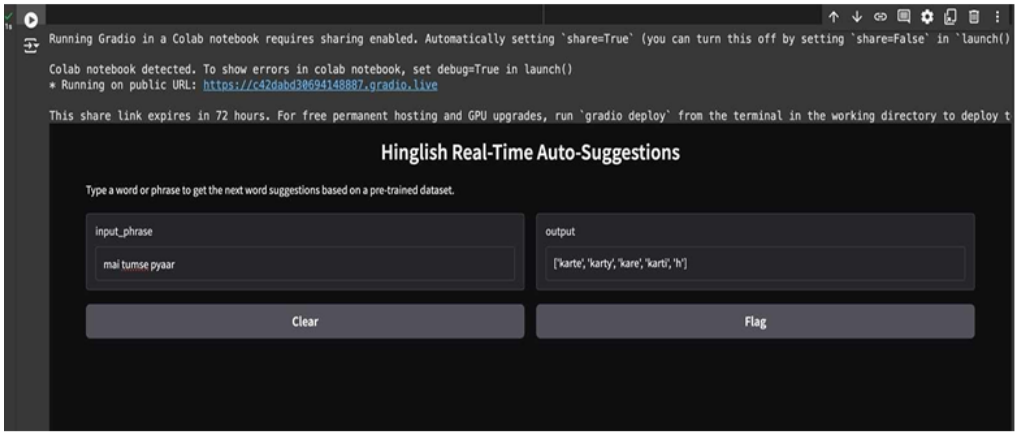
- Loss Function: Categorical cross-entropy to handle the multi-class output.
- Optimizer: Adam optimizer with a learning rate of 0.001.
- Batch Size: 32
- Epochs: 20

The training process included hyperparameter tuning to achieve the best possible results on the validation dataset.

9 Bigram model : 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.



10. DistilBERT: To improve predictions, 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).

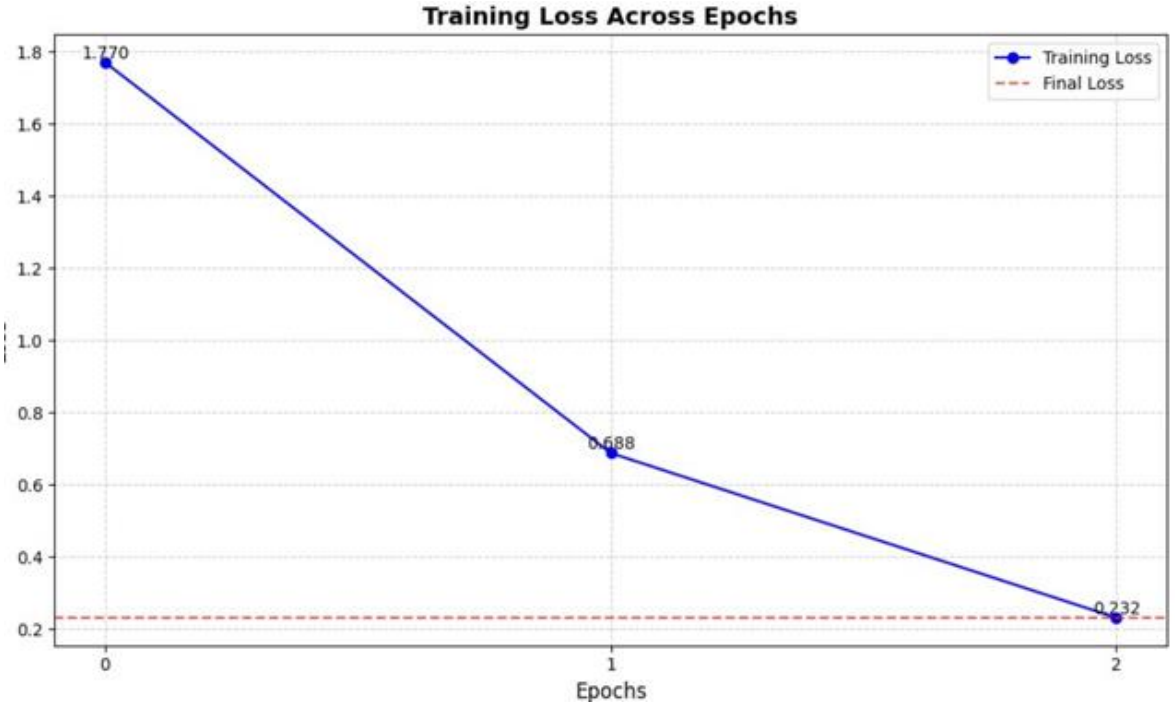
• Utilized GPU when available for faster training.

```
  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-w
hen-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.



11. **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 :

For DISTIL BERT MODEL

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)
```

Using LSTM + DISTIL BERT MODEL:

```
Input Text: ki
Predicted Text: ki previous
```

61]:
```
input_text = "baje"
predicted_text = predict_next_word_with_sampling(model, tokenizer, input_text, max_length=5, temperature=0.8)
print(f"Input Text: {input_text}")
print(f"Predicted Text: {predicted_text}")
```

```
Input Text: baje
Predicted Text: baje районов
```

62]:
```
input_text = "raha"
predicted_text = predict_next_word_with_sampling(model, tokenizer, input_text, max_length=5, temperature=0.8)
print(f"Input Text: {input_text}")
print(f"Predicted Text: {predicted_text}")
```

```
Input Text: raha
Predicted Text: raha
```

63]:
```
input_text = "yaad"
predicted_text = predict_next_word_with_sampling(model, tokenizer, input_text, max_length=5, temperature=0.8)
print(f"Input Text: {input_text}")
print(f"Predicted Text: {predicted_text}")
```

```
Input Text: yaad
Predicted Text: yaad
```

2. DistilBERTTraining_Finetuning&Evaluate.ipynb

Epoch	Training Loss	Validation Loss
0	1.518800	3.938618
1	1.310600	3.686892
2	1.280500	3.530458

```
Enter a sentence: main tumse pyaar
Predicted next 5 words: hih main ki k
```

3. FineTuningMultiLingualBERTwithWORD2VEC.ipynb

Epoch	Training Loss	Validation Loss
1	1.535900	nan
2	1.238800	3.850853
3	1.104900	nan

```
Enter a sentence: uske
Predicted next 5 words: haine logo meinya
```

Analysis:

1.Bigram model: useful for short-term predictions but struggled with capturing longer dependencies.
2.DISTIL BERT: this is faster in all GPU models. That's why we chose this.and the results we got are also satisfactory as accuracy increased using this.
3. LSTM + BERT : as LSTM is good with semantic knowledge and Bert is encode - only model. But the results aren't satisfactory. LSTM is left to right and BERT is bidirectional.
4.MultiLingual BERT: mBERT showed a steady reduction in training loss, but encountered NaN values in validation loss during some epochs, indicating potential issues with model stability in handling Hinglish data for auto-suggestions.

CONCLUSION:

Eventhough we tried different kinds of models we got the best results from DISTIL BERT is more stable than the other Bert models in terms of training and validation loss and performance consistency.although m-bert gives better results in terms of bleu score for higher accuracy applications.

ACKNOWLEDGEMENT :

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

Special Thanks:

we would like to express my gratitude to Nidhi Mam for her invaluable guidance and support throughout this project.