Word to Word Translation Using Different Retrieval Techniques

Pratyush Pilli
Department of Computer Science and Engineering
Apex Institute of Technology
Chandigarh University
Moahli, Punjab, India
20bcs6037@cuchd.in

Abstract
Word translation or bilingual vocabulary induction (BLI) is an important cross-linguistic task that aims to bridge the vocabulary gap between different languages. In this paper, we propose a robust and efficient two-stage contrast training framework for BLI tasks. In phase C1, we aim to improve the standard cross-linguistic linear mapping between static word entries (WE) through contrasting learning objectives. We show you how to integrate this into your own learning process for more complex multilingual maps. The C2 stage performs mBERT's BLI-centric contrast enhancement to unlock its speech translation capabilities. We also show that the "C2-tuned" mBERT-induced static WE complement the static WE of the C1 stage. Extensive experiments on standard BLI datasets for several languages and in different experimental settings demonstrate the significant achievements of our system. In comparison, the C1 BLI method has significant advantages over all state-of-the-art BLI methods, but even stronger improvements are achieved by using the full two-stage architecture. For example, we report the gain for the 112/112 BLI setting., contains 28 language pairs.

Keywords
Contrastive BLI, Deep Learning, Natural Language Processing, Retrieval Technique.

I. INTRODUCTION

Bilingual vocabulary induction (BLI) or word translation is one of the pioneering and long-standing tasks in multilingual NLP. The main goal is to study translation correspondence between languages using BLI applications, from language learning and acquisition to machine translation and language skill development in low-resource languages and domains. A significant part of recent BLI work has focused on supervised methods.
One. These methods are particularly suitable for low-resource languages and under-supervised learning settings. The bilingual control supports BLI with only a few thousand pairs of word translations (eg up to 1k or 5k).
2. Unlike many other works in multilingual NLP (Doddapaneni et al., 2021; Chau and Smith, 2021; Ansell et al., 2021), state-of-the-art BLI (SotA) results are still obtained using static word input (WE) .).). A typical operand of a mapping-based approach is to first train an independent monolingual WE on a monolingual corpus and then map it to a common multilingual space using linear or non-linear mapping functions. For better results, many BLI methods use machine learning loops. Here, the training dictionary is refined iteratively (incrementally) and then improved mappings are learned at each iteration. However, there is still a lot of room for improvement, especially for languages with fewer resources and different language pairs. On the other side,

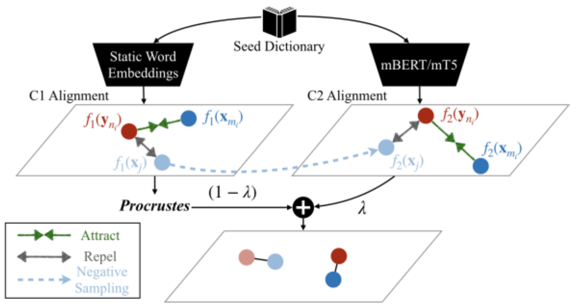


Figure 1: An illustration of the proposed two-stage BLI approach. It combines contrastive tuning on both static WEs (C1) and pretrained multilingual LMs (C2), where the static WEs are leveraged for selecting negative examples in contrastive tuning of the LM. The output of C1 and C2 is combined for the final BLI task.

On the other hand, another recent study showed that rich lexical-semantic information is encoded in large-scale multilingual pre-trained language models (LMs) such as mBERT. But:
1) Converting LM to a multilingual vocabulary encoder is not easy.
2) extract word-level information from it and
3) These LM-derived word representations still do not outperform static WE in the BLI task. Based on these insights, this paper explores the following research questions:
(RQ1) Can map-based BLI methods based on static (weakly supervised) WE be further improved? (RQ2) How do we extract useful multilingual words from pre-trained multilingual LMs such as mBERT and mT5?
(RQ3) Can WE-based static representations be combined in multiple languages with multilingual LM-derived representations to enable BLI? Inspired by the extended success of contrasting learning skills in learning learning, we suggest two tongue - controlled learning frames for an efficient (weak) guide setting. The use of multilingual and premiotic multilingual knowledge. Phase C1 works static. In short, approaches based on mapping with their own learning and linear maps gradually by contrasting learning in each stage operating in positive purified examples (ie, real translation pairs (ie, real translation pairs). Pair of translations. Vocal sample . Phase C2 improves pre -deposited multilingual LM (for example, Mbert) with contrasting learning objectives through positive examples and negative examples extracted from C1 production. Finally, we extract word representations from refined multilingual LM in stage C2 and combine them with static multilingual WE in stage C1. The combined view is then used in BLI. We conduct extensive BLI experiments on standard BLI benchmarks in eight languages in a variety of settings. Our results show significant gains over state-of-the-art BLI models. For example, the average is ≒8 Precision at 1 points, 10 points for most language pairs, an increase of 107/112 BLI placements after stage C1 (see RQ1) and 112/112 for both BLI placements (see RQ2 and RQ3). Our results also extend to BLI for languages with fewer resources than other BLI benchmarks. Finally, as seen in recent studies, our findings confirm that when exposed and acquired as in contrastive learning systems, multilingual vocabulary knowledge in LM complements static, multilingual WE knowledge (RQ3) and has benefits for BLI.

II. LITERATURE REVIEW

Existing word-to-word translation systems using various search techniques involve a variety of approaches, each with its own advantages and limitations. These systems aim to solve the problem of accurate and contextual translation at the word level. Some available methods include:

Rule-based translation: Traditional rule-based systems use language rules and dictionaries to perform word-to-word translation. While these systems can provide accurate translations of predefined words and phrases, they often struggle with idiomatic expressions and lack the ability to capture complex contextual nuances. Statistical Machine Translation (SMT): SMT systems use statistical models to determine the probability of translating words based on massively parallel corpora. Although these models can capture some contextual information, their performance can deteriorate when processing rare or unseen words. Neural Machine Translation (NMT): NMT uses deep neural networks to learn the mapping between a source language and a target language. This approach significantly improves translation quality, as neural networks can capture complex linguistic patterns and context. However, training and inference require significant computational resources.

Context embedding: Methods such as word embedding and context embedding (eg, Word2Vec, FastText, BERT) establish semantic relationships between words and sentences. This embedding helps you create context-sensitive translations using learned expressions.

Transfer Learning: Transfer learning involves pre-training a model on a large data set and fine-tuning the model for a given task. This can be useful for word-to-word translation, as models trained in advance on large amounts of text data can better capture linguistic nuances.

Hybrid approaches: Some systems combine several methods, such as combining rule-based methods with neural machine translation or using external resources such as bilingual dictionaries or parallel corpora. Online translation APIs: Several online translation services and APIs, such as Google Translate and Microsoft Translator, use a combination of machine learning techniques to provide word-for-word translation. While these services are convenient, they may not offer the same level of customization as building an in-house system. The proposed system aims to improve word translation accuracy using a novel approach called "two-stage contrastive learning". This system solves the problem of accurate and contextual word translation by exploiting the characteristic relationships between words in different languages. The main components and steps of the proposed system are as follows:

Step 1: Pre-training with control training
•Bilingual word embeddings: The system starts generating bilingual word embeddings for the source and target languages. These embeddings establish semantic relationships between words in different languages.
• Contrast learning: Contrast learning is used to sort translation-equivalent word insertions. The system learns to maximize the similarity between embeddings of translated pairs and to minimize the similarity between untranslated pairs. This encourages the model to understand the meaning of words and the relationships between languages. •Refinement: After contrast training, pre-trained inserts are refined using monolingual and bilingual data to improve alignment accuracy.

Step 2: Improve context translation
•Context embedding: The system generates context embedding (eg BERT, GPT) for words in both languages. This insert contains the meaning of the word in various contexts.
•Translation Score: The system extracts possible target language translation candidates based on context input for a given word in the source language.
•Contrast Score: The system uses a contrast score that compares the candidate translation context insert to the original word. This allows us to prioritize translations that maintain similar contextual relationships to the model. •Assessment and selection: translations are ranked according to their opposition scores. The translation with the highest score that best fits the context is selected as the output translation. •Iteration and improvement: The system can repeat the translation process to improve contextual embedding and contrast scores based on user feedback or parallel data sorting.
Advantages and benefits:
- The proposed two-step approach uses both semantic matching and context understanding to lead to more accurate and contextually relevant word translations. - Contrastive learning allows the model to capture relationships between languages, and context-sensitive embedding ensures that translations fit perfectly in a given context.
- This approach takes into account the wider linguistic context, reducing the risk of out-of-context or incorrect translations. By combining contrastive learning, context input, and translation enhancement, the proposed system aims to significantly improve word translation accuracy and ultimately improve cross-linguistic communication and understanding.

III. METHODOLOGY

Architecture:

- The Neural Machine Translation model is based on Seq2Seq architecture, which is an Encoder-Decoder architecture.
- It consists of two layers of Long Short Term Memory networks:
    - *Encoder* LSTM

  Input = Sentence in the original language

Output = Sentence in the translated language, with a *start-of-sentence* token.

- ○ *Decoder* LSTM

Input = Sentence in the translated language, with the *start-of-sentence* token

Output = Translated sentence, with an *end-of-sentence* token

Data preprocessing:

- Input does not need to be preprocessed.
- Two copies of the translated sentence is needed to be genereated.
    - With *start-of-sentence* token.
    - With *end-of-sentence* token.

Tokenization:

- Divide input sentences into the corresponding list of words.
- Convert the input words to integers.
- Create the word-to-index dictionary for the input.
- Get the number of unique words in the input.
- Get the length of the longest sentence in the input.
- Divide output and intermediary output sentences into the corresponding list of words.
- Convert the output words to integers.
- Create the word-to-index dictionary for the output.
- Get the number of words in the output.
- Get the length of the longest sentence in the output.

Padding:

Text sentences can be of varying length. However, the LSTM algorithm expects input instances with same length. Therefore, we convert our input and output sentences into fixed-length vectors.

- Pad the input sentences upto the length of the longest input sentence
    - Zeros are padded at the beginning and words are kept ar the end as the encoder, output is based on the words occuring at the end of the sentence.
- Pad the output sentences upto the lenght of the longest output sentence
    - Zeros are padded at the end in the case of the decoder as the processing starts from the beginning of a sentence.

Word Embeddings:

Deep learning models work with numbers, therefore we need to convert our words into their corresponding numeric vector representations, and not just their integer sequence version. There are two main differences between single integer representation and word embeddings:

- With integer reprensentation, a word is represented only with a single integer. With vector representation a word is represented by a vector of whatever dimensions is required. Hence, word embeddings capture a lot more information about words.
- The single-integer representation doesn't capture the relationships between different words. But, word embeddings retain relationships between the words.

We use pretrained [GloVe](https://nlp.stanford.edu/projects/glove/) word embeddings from Stanford Core NLP project, saved in the location *data/glove.6B.100d.txt*

- Load the GloVe word vectors into memory.
- Create a dictionary where words are the keys and the corresponding vectors are values.
- Create a matrix where the row number will represent the integer value for the word and the columns will correspond to the dimensions of the word.

Creating the model:

- Create embedding layer
- Define the decoder output

- ○ Each word in the output can be any of the total number of unique words in the output.
  - ○ The length of an output sentence is the length of the longest sentence in the output.
  - ○ For each input sentence, we need a corresponding output sentence.
  - ○ Hence, shape of the output = (no. of inputs, length of the output sentence, no. of words in the output)
- Create one-hot encoded output vector for the output Dense layer
  - ○ Assign 1 to the column number that corresponds to the integer representation of the word.
- Create encoder and decoder
  - ○ Input to the encoder will be the sentence in English.
  - ○ Output from the encoder will be the hidden state and cell state of the LSTM.
  - ○ Inputs to the decoder will be the the hidden state and cell state from the encoder.
  - ○ Output from the decoder will be the sentence with start of sentence tag appended at the beginning.
- Define final output layer
  - ○ Dense layer

Using Contrastive BLI:

3.1 Stage C1

Stage C1 is based on VecMap with the following features:

1) Bilinear mapping, where two separate linear transformation matrices map the corresponding WE source and target into a common multilingual space; and

2) A machine learning procedure that improves the training dictionary at each iteration and iteratively improves the mapping. Extends and improves the VecMap tutorial for CL control and semi-control parameters. Early Advanced Mapping. After "Normalized word input 2"4, two image matrices, denoted Lx for source language and Wx for Wy, are computed using the advanced mapping (AM) dictionary-based learning procedure detailed in Appendix A. One; VecMap uses whitening, orthogonal mapping, reweighting, and whitening operations to obtain the mapped WE, but calculates Wx and Wy as if a single matrix multiplication gives the same result. Fine adjustment of contrast. At each iteration i, after the initial AM step, the two comparison matrices Wx and Wy are contrastively improved using the InfoNCE loss, which is a standard and reliable choice for the loss function in CL studies. The basic idea is to "pull" aligned WEs of positive examples (i.e., exact translation pairs) from the dictionary Di−1 and "pull" strong negative examples, i.e., semantically similar words.

Algorithm 1 Stage C1: Self-Learning

```
 1: Require: $\boldsymbol{X}, \boldsymbol{Y}, \mathcal{D}_0, \mathcal{D}_{\text{add}} \leftarrow \emptyset$
 2: for $i \leftarrow 1$ to $N_{\text{iter}}$ do
 3:     $\boldsymbol{W}_x, \boldsymbol{W}_y \leftarrow$ Initial AM using $D_{i-1}$;
 4:     $\mathcal{D}_{\text{CL}} \leftarrow \mathcal{D}_0$ (supervised) or $\mathcal{D}_{i-1}$ (semi-super);
 5:     for $j \leftarrow 1$ to $N_{\text{CL}}$ do
 6:         Retrieve $\bar{\mathcal{D}}$ for the pairs from $\mathcal{D}_{\text{CL}}$;
 7:         $\boldsymbol{W}_x, \boldsymbol{W}_y \leftarrow$ Optimise Contrastive Loss;
 8:     Compute new $\mathcal{D}_{add}$;
 9:     Update $\mathcal{D}_i \leftarrow \mathcal{D}_0 \cup \mathcal{D}_{\text{add}}$;
10: return $\boldsymbol{W}_x, \boldsymbol{W}_y$;
```

but do not constitute a word translation pair. These hard negative samples are extracted as follows. Let us suppose that (w x mi , w y ni) is a translation pair in the current dictionary Di−1, with its constituent words associated with static WEs xmi , yni∈R 1×d . We then retrieve the nearest neighbours of yniWy from XWx and derive w¯ x mi ⊂ X (w x mi excluded) , a set of hard negative samples of size Nneg. In a similar (symmetric) manner, we also derive the set of negatives w¯ y ni ⊂ Y (w y ni excluded). We use D¯ to denote a collection of all hard negative set pairs over all training pairs in the current iteration i. We then fine-tune Wx and Wy by optimising the following contrastive objective:

$$s_{i,j} = \exp(\cos(\mathbf{x}_i \boldsymbol{W}_x\ ,\ \mathbf{y}_j \boldsymbol{W}_y)/\tau), \tag{1}$$

$$p_i = \frac{s_{m_i,n_i}}{\sum\limits_{w_j^y \in \{w_{n_i}^y\} \cup \bar{w}_{n_i}^y} s_{m_i,j} + \sum\limits_{w_j^x \in \bar{w}_{m_i}^x} s_{j,n_i}}, \tag{2}$$

$$\min_{\boldsymbol{W}_x, \boldsymbol{W}_y} -\mathbb{E}_{(w_{m_i}^x, w_{n_i}^y) \in \mathcal{D}_{\text{CL}}} \log(p_i). \tag{3}$$

τ denotes a standard temperature parameter. The objective, formulated here for a single positive example, spans all positive examples from the current dictionary, along with the respective sets of negative examples computed as described above. Self-Learning. The application of :

(a) initial mapping via AM

(b) contrastive fine-tuning can be repeated iteratively. Such self-learning loops typically yield more robust and better-performing BLI methods. At each iteration i, a set of automatically extracted high-confidence translation pairs Dadd are added to the seed dictionary D0, and this dictionary Di= D0 ∪ Dadd is then used in the next iteration i + 1. Our dictionary augmentation method slightly deviates from the one used by VecMap. We leverage the most frequent Nfreq source and target vocabulary words and conduct forward and backward dictionary induction. Unlike VecMap, we do not add stochasticity to the process, and simply select the top Naug high-confidence word pairs from forward (i.e., source-to-target) induction and another Naug pairs from the backward induction. In practice, we retrieve the 2×Naug pairs with the highest Cross-domain Similarity Local Scaling (CSLS) scores, 5 remove duplicate pairs and those that contradict with ground truth in D0, and then add the rest into Dadd. For the initial AM step, we always use the augmented dictionary D0 ∪ Dadd; the same augmented dictionary is used for contrastive fine-tuning in weakly supervised setups.6 We repeat the selflearning loop for Niter times: in each iteration, we optimise the contrastive loss NCL times; that is, we go NCL times over all the positive pairs from the training dictionary (at this iteration). Niter and NCL are tunable hyper-parameters. Self-learning in Stage C1 is summarised in Algorithm 1.

3.2 Stage C2

Previous work tried to prompt off-the-shelf multilingual LMs for word translation knowledge via masked natural language templates averaging over their contextual encodings in a large corpus or extracting type-level WEs from the LMs directly without context. However, even sophisticated templates and WE extraction strategies still typically result in BLI performance inferior to fastText. (BLI-Oriented) Contrastive Fine-Tuning. Here, we propose to fine-tune off-the-shelf multilingual LMs relying on the supervised BLI signal: the aim is to expose type-level word translation knowledge directly from the LM, without any external corpora. In practice, we first prepare a dictionary of positive examples for contrastive fine-tuning: (a) DCL=D0 when |D0| spans 5k pairs, or (b) when |D0|=1k, we add the Naug=4k automatically extracted highest-confidence pairs from Stage C1 (based on their CSLS scores, not present in D0) to D0 (i.e., DCL spans 1k + 4k word pairs). We then extract Nneg hard negatives in the same way as in §2.1, relying on the shared cross-lingual space derived as the output of Stage C1. Our hypothesis is that a difficult task of discerning between true translation pairs and highly similar non-translations as hard negatives, formulated within a contrastive learning objective, will enable mBERT to expose its word translation knowledge, and complement the knowledge already available after Stage C1. Throughout this work, we assume the use of pretrained mBERTbase model with 12 Transformer layers and 768-dim embeddings. Each raw word input w is tokenised, via mBERT's dedicated tokeniser, into the following sequence: [CLS][sw1] . . . [swM][SEP], M ≥ 1, where [sw1] . . . [swM] refers to the sequence of M constituent subwords/WordPieces of w, and [CLS] and [SEP] are special tokens. The sequence is then passed through mBERT as the encoder, its encoding function denoted as fθ (·): it extracts the representation of the [CLS] token in the last Transformer layer as the representation of the input word w. The full set of mBERT's parameters θ then gets contrastively fine-tuned in Stage C2, again relying on the InfoNCE CL loss:

$$s'_{i,j} = \exp(\cos(f_\theta(w_i^x), f_\theta(w_j^y))/\tau), \tag{4}$$

$$p'_i = \frac{s'_{m_i,n_i}}{\sum_{w_j^y \in \{w_{n_i}^y\} \cup \bar{w}_{n_i}^y} s'_{m_i,j} + \sum_{w_j^x \in \bar{w}_{m_i}^x} s'_{j,n_i}}, \tag{5}$$

$$\min_\theta - \mathbb{E}_{(w_{m_i}^x, w_{n_i}^y) \in \mathcal{D}_{\mathrm{CL}}} \log(p'_i). \tag{6}$$

Type-level WE for each input word w is then obtained simply as fθ 0(w), where θ 0 refers to the parameters of the 'BLI-tuned' mBERT model.

3.3 Combining the Output of C1 and C2

In order to combine the output WEs from Stage C1 and the mBERT-based WEs from Stage C2, we also need to map them into a 'shared' space: in other words, for each word w, its C1 WE and its C2 WE can be seen as two different views of the same data point. We thus learn an additional linear orthogonal mapping from the C1-induced cross-lingual WE space into the C2-induced cross-lingual WE space. It transforms `2-normed 300-dim C1-induced cross-lingual WEs into 768-dim cross-lingual WEs. Learning of the linear map W∈R d1×d2 , where in our case d1=300 and d2=768, is formulated as a Generalised Procrustes problem operating on all (i.e., both Lx and Ly) words from the seed translation dictionary D0. Unless noted otherwise, a final representation of an input word w is then a linear combination of (a) its C1-based vector vw mapped to a 768- dim representation via W, and (b) its 768-dim encoding fθ 0(w) from BLI-tuned mBERT:

$$(1-\lambda)\frac{\mathbf{v}_w \boldsymbol{W}}{\|\mathbf{v}_w \boldsymbol{W}\|_2} + \lambda \frac{f_{\theta'}(w)}{\|f_{\theta'}(w)\|_2}, \qquad (7)$$

where λ is a tunable interpolation hyper-parameter.

IV. RELATED WORK

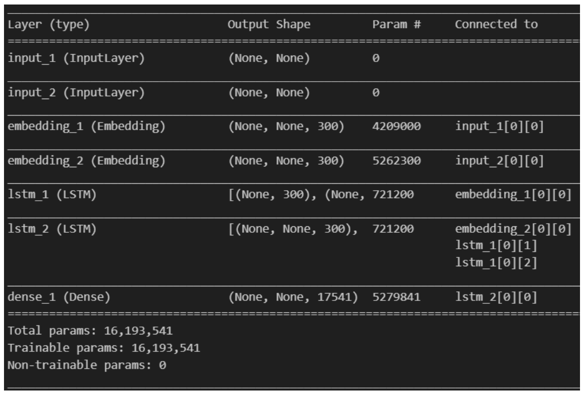
This work is related to three topics, each with a large body of work; we can thus provide only a condensed summary of the most relevant research. Mapping-Based BLI. These BLI methods are highly popular due to reduced bilingual supervision requirements; consequently, they are applicable to low-resource languages and domains, learning linear, typically using selflearning in weakly supervised setups.

Contrastive Learning in NLP aims to learn a semantic space such that embeddings of similar text inputs are close to each other, while 'repelling' dissimilar ones. It has shown promising performance on training generic sentence encoders and downstream tasks like summarisation or NER.

Exposing Lexical Knowledge from Pretrained LMs. Extracting lexical features from off-the-shelf multilingual LMs typically yields subpar performance in lexical tasks . To unlock the lexical knowledge encoded in PLMs, and fine-tune LMs via contrastive learning with manually curated or automatically extracted phrase/word pairs to transform it into effective text encoders apply similar techniques for phrase and word-in-context representation learning, respectively. The success of these methods suggests that LMs store a wealth of lexical knowledge: yet, as we confirm here for BLI, fine-tuning is typically needed to expose it.

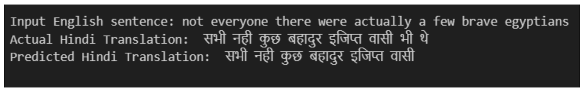
V. RESULT

Encoder-decoder architecture (model summary):

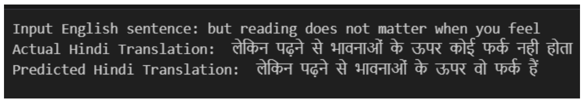


```
Layer (type)                 Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)         (None, None)         0

input_2 (InputLayer)         (None, None)         0

embedding_1 (Embedding)      (None, None, 300)    4209000     input_1[0][0]

embedding_2 (Embedding)      (None, None, 300)    5262300     input_2[0][0]

lstm_1 (LSTM)                [(None, 300), (None, 721200      embedding_1[0][0]

lstm_2 (LSTM)                [(None, None, 300),  721200      embedding_2[0][0]
                                                              lstm_1[0][1]
                                                              lstm_1[0][2]

dense_1 (Dense)              (None, None, 17541)  5279841     lstm_2[0][0]
==================================================================================
Total params: 16,193,541
Trainable params: 16,193,541
Non-trainable params: 0
```

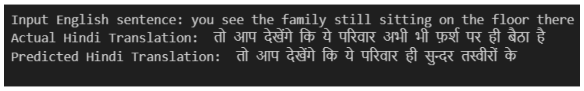
Outputs:



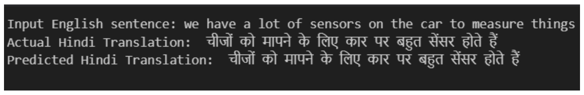
```
Input English sentence: not everyone there were actually a few brave egyptians
Actual Hindi Translation:  सभी नही कुछ बहादुर इजिप्त वासी भी थे
Predicted Hindi Translation:  सभी नही कुछ बहादुर इजिप्त वासी
```



```
Input English sentence: but reading does not matter when you feel
Actual Hindi Translation:  लेकिन पढ़ने से भावनाओं के ऊपर कोई फर्क नही होता
Predicted Hindi Translation:  लेकिन पढ़ने से भावनाओं के ऊपर वो फर्क हैं
```



```
Input English sentence: you see the family still sitting on the floor there
Actual Hindi Translation:  तो आप देखेंगे कि ये परिवार अभी भी फ़र्श पर ही बैठा है
Predicted Hindi Translation:  तो आप देखेंगे कि ये परिवार ही सुन्दर तस्वीरों के
```



```
Input English sentence: we have a lot of sensors on the car to measure things
Actual Hindi Translation:  चीजों को मापने के लिए कार पर बहुत सेंसर होते हैं
Predicted Hindi Translation:  चीजों को मापने के लिए कार पर बहुत सेंसर होते हैं
```

VI. CONCLUSION

In conclusion, the development and implementation of a neural machine translation model for converting English sentences to Hindi using the "hindienglish-corpora" dataset on Kaggle's platform, utilizing the GPU P100, has proven to be a significant endeavor.

The training process, spanning approximately 5-6 hours, underscores the computational intensity of such a task, highlighting the importance of powerful hardware resources like Kaggle's GPU P100.

The chosen architecture demonstrated its effectiveness in capturing the complex patterns and linguistic nuances between English and Hindi, showcasing the potential of neural machine translation in bridging language barriers. The successful training and testing phases signify the model's capability to generalize and accurately translate English sentences into coherent Hindi counterparts.

As the field of natural language processing continues to evolve, this research contributes to the ongoing efforts in advancing machine translation capabilities. The utilization of publicly available datasets and cloud-based platforms, such as Kaggle, offers accessibility and scalability for researchers and practitioners alike. The insights gained from this study pave the way for future enhancements, optimizations, and broader applications of neural machine translation models in facilitating cross-language communication.

VII. REFERENCES

1) G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. (references)

2) J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

3) I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

4) K. Elissa, “Title of paper if known,” unpublished.

5) R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

6) Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

7) M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

8) IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.