Enhancing Word Meaning Retrieval Using FastText and NLP Techniques

Sankalp Devanand, Prateek Agasimani, V. S. Shamith, Rohith Neeraje

***Abstract*—**Machine translation has witnessed significant advancements in recent years, but the translation of languages with distinct linguistic characteristics, such as English and Sanskrit, remains a challenging task. This research presents the development of a dedicated English to Sanskrit machine translation model, aiming to bridge the linguistic and cultural gap between these two languages. Using a variety of natural language processing (NLP) approaches including FastText embeddings, this research proposes a thorough method to improve word meaning retrieval. Data preparation, part-of-speech tagging, dictionary searches, and transliteration are all included in the methodology. The study also addresses the implementation of an interpreter pattern and uses a word similarity task to assess the quality of word embeddings. The experimental outcomes show how the suggested approach may be used to enhance word meaning retrieval tasks with greater efficacy, accuracy, and adaptability. Evaluation of the model's performance is conducted through rigorous testing, comparing its output against existing machine translation systems. The assessment includes quantitative metrics such as BLEU scores, METEOR scores, Jaccard Similarity etc.

***Keywords*—**Machine translation, English to Sanskrit, natural language processing, word meaning retrieval, FastText embeddings.

I. Introduction

LANGUAGE has always been a powerful vehicle for human communication, bridging gaps, and fostering understanding between individuals and cultures. As our world becomes increasingly interconnected through globalization and digital technology, the importance of effective cross-linguistic communication has grown exponentially. In this context, machine translation has emerged as a pivotal tool in breaking down language barriers and facilitating communication between speakers of different languages.

While machine translation systems have made remarkable progress in translating languages with similar linguistic structures and widespread usage, the translation of languages with distinct characteristics presents a unique set of challenges. English and Sanskrit, two languages with rich historical and cultural significance, exemplify this challenge. Sanskrit, one of the world's oldest languages, holds a prominent place in religious, philosophical, and classical literature. Its unique grammatical structure, vast lexicon, and intricate morphology pose significant obstacles for machine translation systems designed primarily for modern, widely spoken languages like English [1].

This research endeavors to address the intricate task of English to Sanskrit machine translation, aiming to bridge the gap between these two languages and cultures. By developing a dedicated machine translation model, we seek to unlock the potential for cross-cultural communication, knowledge dissemination, and preservation of Sanskrit's linguistic and cultural heritage.

The main purpose of current work is to investigate English to Sanskrit translation using dictionary-based system and analyze on the grounds of different BLEU, METEOR, Jaccard scores to improve the quality of the existing MT output. The rest of the paper is structured as follows: Section II, concisely outlines related works. Section III, elaborates dataset. Section IV, gives a brief description of different methodologies used. Section V, details result of automatic translators and comparative analysis of results acquired from various systems. Section VI, conclusion and future work. Lastly, Section VII, ends the paper with references.

II. Related Works

Dictionary based translation approach has been used English to Sanskrit to overcome limitations of neural machine translation such as accuracy which google is lacking. For an effective translation model, the datasets are the most important aspect [2]. For this purpose, literature survey studies the effects of various types of datasets with their pros and cons. Dataset is collected and preprocessed like replacing characters in words, replacing characters in sentences, tokenizing, replacing NaN values with <nan> tokens, performing pos tagging etc. Then training a FastText model with the preprocessed dataset and loading the model. The model performs get_most_relevant_sentence() function to get the most relevant word for the given input word and different validation methods are used to validate the model automatically [3].

In this context, our research aims to build upon these related works by applying state-of-the-art data preprocessing, training, and validation techniques to the challenging task of English-Sanskrit machine translation. By addressing the unique linguistic characteristics and resource limitations of Sanskrit, we strive to contribute to the advancement of machine translation technology for underrepresented languages.

III. Dataset

For an effective translation model, the dataset is the most vital factor. For those purposes, this literature survey studies the effects of various sorts of datasets with their execs and cons, and concludes that the dictionary dataset is the maximum most

Rohith Neeraje is with the PES University, India (corresponding author, e-mail: rohithneeraje12@gmail.com).

advantageous for a word- to-word model and a graphical based dataset is favored for a sentence- to-sentence translation.

The dictionary-based model is selected for the word translation is due to the emphasis on translating character words in place of sentence. As such the model does no longer require the semantics of the language as an entire for sentences [4]. This also covers a huge range of vocabulary, that is essential for the accuracy of the model. It also allows the system to learn the precise translation of each phrase in its supply language. This approach has validated to be the simplest for the word-to-word translation of maximum languages.

Inside the context of a sentence-to-sentence translation model, a parallel corpora dataset also can be used [5]. A parallel corpus [6] includes bilingual texts which encompass both sentences and phrases in each the languages English and Sanskrit. Those datasets are generally constructed from a collection of various books and web sites. There are numerous benefits to the usage of this method, which are the parallel corpus helps the model to accurately capture the sentence level context of the languages, and allows within the fluency and grammar of the target language. But there are numerous disadvantages to use a parallel corpus that are there is no available parallel corpus for the language pair, and making sure accurate alignment between the English and Sanskrit sentences is complex and errors prone [7]. Hence the authors concluded that the usage of a parallel corpus for a sentence translation version is extraordinarily complicated.

For the sentence-to-sentence model, a graph dataset [8] has been chosen. This is as it the graph dataset is beneficial for sentence level translation where the point of interest is on translating entire sentences in place of person phrases. It lets in the model to study the relation among the phrases within the sentence, and this offers the context and semantic for the sentence and allows in information how they're used to bring that means. It additionally helps in disambiguating phrase meanings and resolving polysemy effectively. This method works well when translating complicated sentences and paragraphs that require a deep understanding of the context [9].

IV. Methodology

Several methodologies have been used in areas like data preprocessing, system training, testing. The same are described as follows.

A. Data Preprocessing

Corpus of data has been collected from various sources. The key function of preprocessing step is to tokenization of word and their meanings and creating a dictionary, which indexes the words present in the training process. The preprocessing steps replaces the several characters in meaning column with the blank space. Then each entry in the meaning column will be tokenized and NaN values among these tokens will be removed, then the pos tagging operation will be performed and filtered out the rows with nan pos tags [10].

B. Part-of-Speech Tagging

Using NLTK library, part of speech tagging is carried out on tokenized phrases. Part of speech tagging is a crucial natural language processing task. It involves assigning grammatical categories (such as nouns, verbs, adjectives, etc.) to words in a text. This process aids in understanding sentence structure and meaning, enabling more accurate language analysis and information retrieval.

C. FastText Embedding

In order to record the semantic links between words, FastText embeddings are used. The dataset is used to train a skip-gram model [11], which creates vector representations of words and sentences. The most pertinent sentence for a given input word is retrieved using the cosine similarity of the model. FastText have ability to capture subword information, allowing it to represent out of vocabulary words effectively. FastText’s efficiency and versatility make it a popular choice in natural language processing applications [12].

D. Dictionary Lookup

The use of a dictionary lookup mechanism increases the accuracy of word meaning retrieval [13]. With the use of input sentences, the dataset is filtered to enable the recovery of pertinent word meanings. It involves mapping words or phrases from one language to their equivalent in another language using a prebuilt bilingual dictionary. It helps in handling common and direct translations.

E. Word Similarity Evaluation

A word similarity task is used to assess the quality of FastText embeddings. The cosine similarity between word pairs is calculated using a pre-trained FastText model. The model's success in capturing semantic links is measured using the Spearman correlation coefficient. It calculates the most relevant word from a collection based on the similarity between an input word. It utilizes vector representations of words, employing cosine similarity to measure their likeness. The word with highest similarity to the input word is returned.

F. Transliteration

Transliteration is also used in the technique for terms with non-Latin characters. The Google Input Tools API is used to transliterate the incoming words, turning non-Latin characters into their matching Latin characters. It helps in converting text from one script or writing system into another while maintaining the phonetic or character representation, rather than translating the content. Transliteration modules are incorporated to increase the overall translation accuracy and usability.

G. System Training

In the system training phase, we employed the FastText model. Our choice of model configuration included the use of the skipgram algorithm, a powerful unsupervised learning approach. With a focus on achieving optimal results, we carefully tuned the training parameters, setting the number of epochs to 21, the learning rate to 0.2, and the word vector dimensionality to 300. These parameter choices were guided by previous research and extensive experimentation, ensuring that the model would effectively capture semantic relationships within the input text data. Leveraging this well-configured FastText model, we embarked on training with our dataset, equipping the system with the ability to create meaningful word embeddings and representations that would serve as a foundation for downstream tasks, such as text classification and clustering.

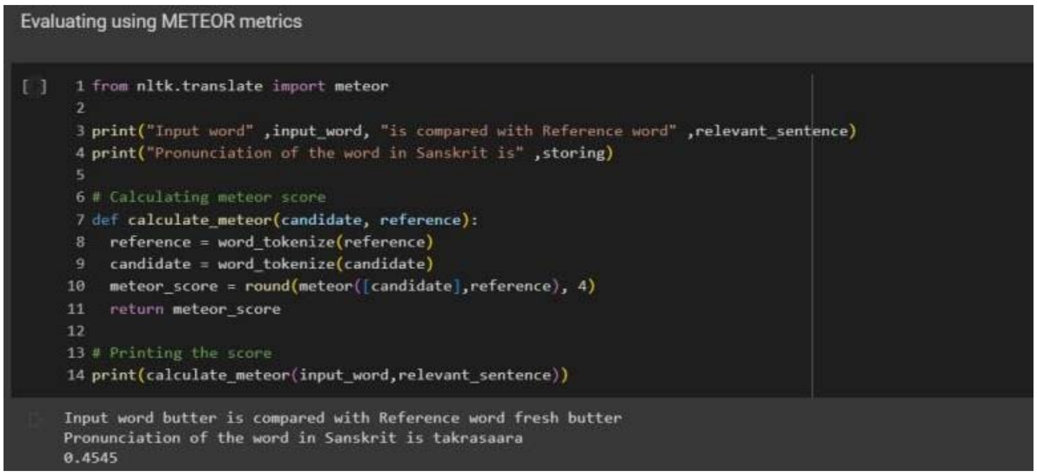
V. Results and Analysis

System training is followed by the system testing where several validation methods are implemented to check the system accuracy automatically. System takes an input word from the user and then system generates a most relevant word from the dictionary with the cosine vectors. After generating the most relevant word, model translates the most relevant word to target language Sanskrit. These input word and most relevant words are inputted into several validation models like BLEU, METEOR, Jaccard similarity, Precision and recall. These methods give the accuracy of words match based on various criteria. The same is explained in detail in Section V.

A. Meteor

Using the NLTK library to calculate the METEOR score, a machine translation evaluation metric, for a candidate word compared to a reference sentence. It tokenizes both the candidate and reference, then computes and prints the METEOR score, which quantifies the quality of the candidate word translation against the reference sentence. For the input word “BUTTER”, the most relevant word generated is “FRESS BUTTER”, and the METEOR score found is 0.4545 out of 0.5, which is approximately 90.9%.

Meteor is a widely used automatic machine translation evaluation metric known for its comprehensive approach to assessing translation quality. It takes into account precision, recall, stemming, synonymy, and word order, providing a versatile and nuanced evaluation of translated text. Researchers and practitioners rely on METEOR to fine- tune machine translation models, compare translation systems, and track improvements in translation quality assessment. Its flexibility makes it valuable for evaluating translations across different languages and domains, making it an essential tool in the field of machine translation.



Evaluating using METEOR metrics

```
from nltk.translate import meteor

print("Input word" ,input_word, "is compared with Reference word" ,relevant_sentence)
print("Pronunciation of the word in Sanskrit is" ,storing)

# Calculating meteor score
def calculate_meteor(candidate, reference):
  reference = word_tokenize(reference)
  candidate = word_tokenize(candidate)
  meteor_score = round(meteor([candidate],reference), 4)
  return meteor_score

# Printing the score
print(calculate_meteor(input_word,relevant_sentence))
```

```
Input word butter is compared with Reference word fresh butter
Pronunciation of the word in Sanskrit is takrasaara
0.4545
```

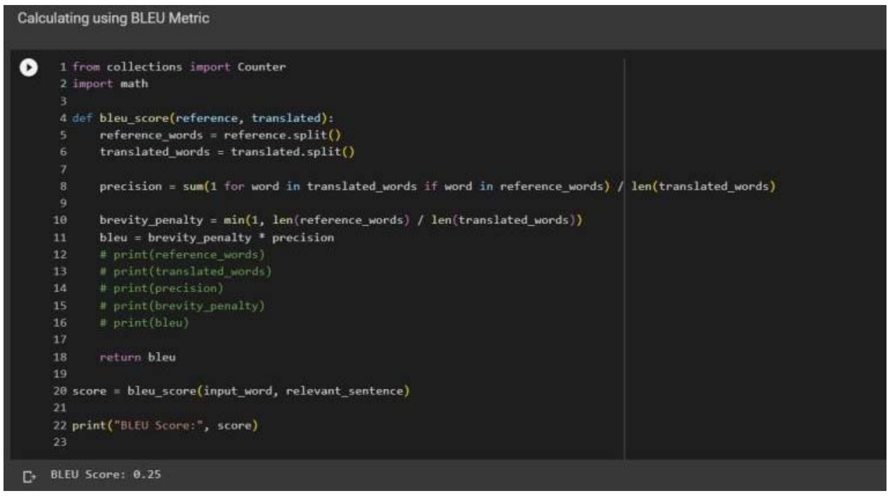
Fig. 1 Evaluating using meteor metrics

B. BLEU

BLEU (Bilingual Evaluation Understudy) is a precision-based machine translation evaluation metric. It computes a score by comparing n-grams (contiguous sequences of words) in the candidate translation to those in the reference translations. BLEU is widely used to measure the quality of machine-generated translations, with higher scores indicating better translation quality. For the input word “BUTTER”, the most relevant word generated is “FRESS BUTTER”, and the BLEU score found is 0.25 out of 1, which is approximately 25%, the reason for this low accuracy is, BLEU method is effective for n-grams. Here we are using unigram for word-to-word translation, for sentence-to-sentence translation, using n-grams will increase the accuracy which is expecting near to METEOR scores.

C. Jaccard Similarity

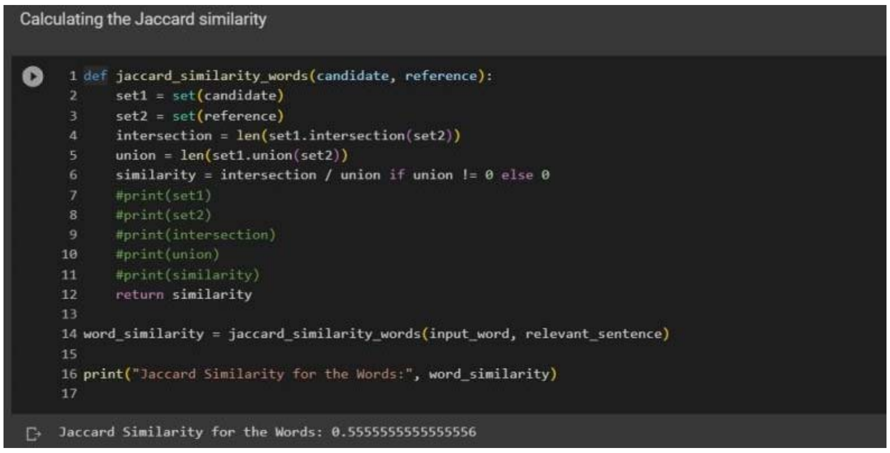
To calculate the Jaccard similarity between two sets of words, candidate and reference. It computes the intersection and union of the word sets and then determines their similarity, which is a measure of word overlap. The code then computes and prints the Jaccard Similarity for the given input word and a relevant sentence. For the input word “BUTTER”, the most relevant word generated is “FRESS BUTTER”, and the Jaccard similarity score found is 0.55 out of 1.0, which is approximately 55%. The accuracy is little lower because, Jaccard similarity uses exact matching of letters in case of w2w model, but for sentence-to-sentence model, words are matched in place of letters, so it makes sense of using this method for sentence-to-sentence model.


Calculating using BLEU Metric

```python
from collections import Counter
import math

def bleu_score(reference, translated):
    reference_words = reference.split()
    translated_words = translated.split()

    precision = sum(1 for word in translated_words if word in reference_words) / len(translated_words)

    brevity_penalty = min(1, len(reference_words) / len(translated_words))
    bleu = brevity_penalty * precision
    # print(reference_words)
    # print(translated_words)
    # print(precision)
    # print(brevity_penalty)
    # print(bleu)

    return bleu

score = bleu_score(input_word, relevant_sentence)

print("BLEU Score:", score)
```

```
BLEU Score: 0.25
```

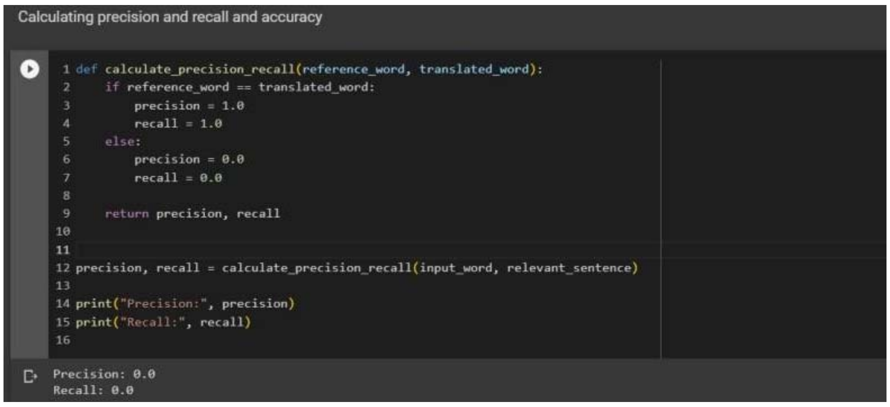
Fig. 2 Evaluating using BLEU metrics


Calculating the Jaccard similarity

```python
def jaccard_similarity_words(candidate, reference):
    set1 = set(candidate)
    set2 = set(reference)
    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))
    similarity = intersection / union if union != 0 else 0
    #print(set1)
    #print(set2)
    #print(intersection)
    #print(union)
    #print(similarity)
    return similarity

word_similarity = jaccard_similarity_words(input_word, relevant_sentence)

print("Jaccard Similarity for the Words:", word_similarity)
```

```
Jaccard Similarity for the Words: 0.5555555555555556
```

Fig. 3 Calculating the Jaccard similarity


Calculating precision and recall and accuracy

```python
def calculate_precision_recall(reference_word, translated_word):
    if reference_word == translated_word:
        precision = 1.0
        recall = 1.0
    else:
        precision = 0.0
        recall = 0.0

    return precision, recall


precision, recall = calculate_precision_recall(input_word, relevant_sentence)

print("Precision:", precision)
print("Recall:", recall)
```

```
Precision: 0.0
Recall: 0.0
```

Fig. 4 Calculating the precision and recall scores

D. Precision and Recall

To calculate precision and recall for a reference word compared to a translated word. If the two words match, precision and recall are both set to 1; otherwise, they are both set to O. The code then computes and prints the precision and recall values for the given input word and relevant sentence. Our score for input word “BUTTER” and generated most relevant word “FRESH BUTTER” is 0, since precision and recall are giving only 2 outputs (1 and 0), if exact match is found, 1 is returned, else 0 is returned.

VI. Conclusion and Future Work

The model developed effectively translates the given input word in English to output word in Sanskrit. Data is collected from various sources and it is preprocessed and loaded into a separate .csv file. We have used FastText embeddings to train the model with the preprocessed data. We have also implemented different functions to perform different operations on the dataset, such as POS tagging, word similarity evaluation, transliteration, dictionary lookup etc. We have implemented different valuation methods for checking the accuracy of generated output. METEOR is giving the good results, while BLEU is performing a little bad as we have considered only unigrams, but for further sentence to sentence model, it is expected to give the better result as we will be using n-gram technique.

For the current model, there are a few limitations. These include, the model is unable to form any meaningful relations between the word and its meaning, and as such, going forward it would be ideal to have or create a dataset which highlights the relations between words and their meanings. The methodologies for analysis of such word-to-word models are not well standardized, and in the future, these analytics, i.e., Meteor score, BLEU score, Jaccard similarity and the precision and recall should be in a more standardized form.

Currently, the implementation of word-to-word translation model is complete. For future works, it starts with the implementation of sentence-to-sentence model. For the sentence-to-sentence model, a graph dataset should be chosen so as to have the optimal results. This is because it the graph dataset is useful for sentence level translation where the focus is on translating entire sentences rather than individual words. It allows the model to learn the relation between the words in the sentence, and this gives the context and semantic for the sentence and helps in understanding how they are used to convey meaning. It also helps in disambiguating word meanings and resolving polysemy effectively. This approach works well when translating complex sentences and paragraphs that require a deep understanding of the context. And so, future works for this framework will help in the dictionary-based approach for any language which has limited support for the dataset, and can be used for almost any language, provided the required dataset has been created and pre-processed in a good way.

References

[1] Fan, Angela, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines et al. "Beyond english-centric multilingual machine translation." The Journal of Machine Learning Research 22, no. 1 (2021): 4839-4886.

[2] Nair, Jayashree, K. Amrutha Krishnan, and R. Deetha. "An efficient English to Hindi machine translation system using hybrid mechanism." In 2016 international conference on advances in computing, communications and informatics (ICACCI), pp. 2109-2113. IEEE, 2016.

[3] Khan, Nadeem Jadoon, Waqas Anwar, and Nadir Durrani. "Machine translation approaches and survey for Indian languages." arXiv preprint arXiv:1701.04290 (2017).

[4] Raulji, Jaideepsinh K., Jatinderkumar R. Saini, Kaushika Pal, and Ketan Kotecha. "A novel framework for Sanskrit-Gujarati symbolic machine translation system." International Journal of Advanced Computer Science and Applications 13, no. 4 (2022).

[5] Krishna, Amrith, Pavankumar Satuluri, and Pawan Goyal. "A dataset for Sanskrit word segmentation." In Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, pp. 105-114. 2017.

[6] Kunchukuttan, Anoop, Pratik Mehta, and Pushpak Bhattacharyya. "The iit bombay english-hindi parallel corpus." arXiv preprint arXiv:1710.02855 (2017).

[7] Schlichtkrull, Michael Sejr, Nicola De Cao, and Ivan Titov. "Interpreting graph neural networks for nlp with differentiable edge masking." arXiv preprint arXiv:2010.00577 (2020).

[8] Zhang, Yuhao, Peng Qi, and Christopher D. Manning. "Graph convolution over pruned dependency trees improves relation extraction." arXiv preprint arXiv:1809.10185 (2018).

[9] Khanuja, Simran, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam et al. "Muril: Multilingual representations for indian languages." arXiv preprint arXiv:2103.10730 (2021).

[10] Laskar, Sahinur Rahman, Abdullah Faiz Ur Rahman Khilji, Darsh Kaushik, Partha Pakray, and Sivaji Bandyopadhyay. "Improved English to Hindi multimodal neural machine translation." In Proceedings of the 8th Workshop on Asian Translation (WAT2021), pp. 155-160. 2021.

[11] Chakravarthi, Bharathi Raja, Mihael Arcan, and John Philip McCrae. "WordNet gloss translation for under-resourced languages using multilingual neural machine translation." In Proceedings of the second workshop on multilingualism at the intersection of knowledge bases and machine translation, pp. 1-7. 2019.

[12] Chakravarthi, Bharathi Raja, Mihael Arcan, and John P. McCrae. "Comparison of different orthographies for machine translation of under-resourced dravidian languages." In 2nd Conference on Language, Data and Knowledge (LDK 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[13] Nair, Jayashree, K. Amrutha Krishnan, and R. Deetha. "An efficient English to Hindi machine translation system using hybrid mechanism." In 2016 international conference on advances in computing, communications and informatics (ICACCI), pp. 2109-2113. IEEE, 2016.

[14] Khan, Nadeem Jadoon, Waqas Anwar, and Nadir Durrani. "Machine translation approaches and survey for Indian languages." arXiv preprint arXiv:1701.04290 (2017).

[15] Fan, Angela, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines et al. "Beyond english-centric multilingual machine translation." The Journal of Machine Learning Research 22, no. 1 (2021): 4839-4886.

[16] Bahadur, Promila, Ajai Jain, and Durg Singh Chauhan. "Architecture of English to Sanskrit machine translation." In 2015 SAI Intelligent Systems Conference (IntelliSys), pp. 616-624. IEEE, 2015.

[17] Punia, Ravneet, Aditya Sharma, Sarthak Pruthi, and Minni Jain. "Improving Neural Machine Translation for Sanskrit-English." In Proceedings of the 17th International Conference on Natural Language Processing (ICON), pp. 234-238. 2020