

SanTran

Sanskrit to English Translator

Web-page: shivanikohli.me/SanTrans

organization: github.com/shivanikohli/SanTrans

Shivani Kohli

MSCS, Marquette University
Milwaukee, WI
Shivani.kohli@marquette.edu

Charlie Morley

MSCS, Marquette University
Milwaukee, WI

Brandon Kupeczyk
MSCS, Marquette University

Shivam Thakrar
MSCS, Marquette University

Ben Walczak
MSCS, Marquette University
Milwaukee, WI

Abstract— Natural Language Processing (NLP) has come a long way from the days of Rule-Based Translation. Today’s leading models utilize Deep Neural Networks (DNN) and use Sequence to Sequence translation. This technique chunks sentences of one language, encodes them, runs them through a DNN, and decodes them into another language. Sanskrit is the root of many Indian languages and is still used today. However, many leading translation tools do not implement the ability to do English-Sanskrit translations. The goal of this project is to utilize Google’s TensorFlow Python module to create a Sequence to Sequence LSTM to translate to and from English and Sanskrit. The largest difficulty of the project is not necessarily the model but rather the data. Public English-Sanskrit translations are scarce. Acquiring accurate translations from the model using scarce data will be difficult. It will require minor changes to the model, as well as the acquisition of more English-Sanskrit translations.

Keywords—Machine Learning; Sanskrit; Translation; NLP

I. INTRODUCTION

Natural language processing is a popular machine learning technique these days for translating texts from one language to another. Currently, tools are available to translate most common or popular languages such as German, Hindi, Punjabi, and Spanish. However, a translation

tool does not exist for Sanskrit. Sanskrit is an ancient language that is one of the 22 official languages in the Eighth Schedule of the Indian Constitution. It is a very important language for the Hindu religion as it still is used in texts and chants to this day. Also, according to the 2001 census of India, Sanskrit has about five million speakers (as a first, second, or third language), so an effective, automated written translator of full sentences would be widely useful. Therefore, the aim of SanTran is to train an Sanskrit-English sentence translation model. Among the many different languages already translated, Sanskrit is one language that is still missing effective tools for full translation. There are numerous online dictionaries for Sanskrit-English word-by-word translation, but there is no full-fledged translator which takes the context and grammar of sentences into account.

Machine Translation is an active but well-developed field of data science; there are already a host of techniques and models that have been tried and tested on numerous languages for full translation. Applying some of these to Sanskrit specifically would be challenging and interesting.

Sanskrit has a non-Latin alphabet, but also an intermediate “Transliteration” language - called IAST - which is made up of Latin characters. This could be used to translate texts using existing models.

The model we will be using is commonly known as Encoder-Decoder LSTM, where LSTM refers to Long-Short Term Memory. LSTM is a specialized version of a recurrent neural network. It works by understanding each word in a sentence based on its understanding of previous words. The context of the sentence helps aid the understanding of what each word within the sentence should be. Recurrent neural networks use loops to allow information to persist in future occurrences. LSTM allows for learning of long-term and short-term dependencies. The model also consists of an encoder-decoder, two submodels. The encoder is responsible for generalizing and summarizing the semantics between many languages. Whereas the decoder is responsible for predicting an output sequence within the given language, one character per iteration of the recurrent neural network.

II. RELATED WORKS






Various authors [5] have discussed that Sanskrit is a viable language for Natural Language Processing as compared to other languages since its grammar and syntax is easier to understand. Words like “it is” and “I” do not have separate words in Sanskrit but can be easily decoded. Due to inflections, it reduces a four-letter English word to 2 letters in Sanskrit making it easier to decode and reducing storage space.

Upon research, two methods were found for the process of translating Sanskrit to English. The first comes from Young-Suk Lee who describes Statistical Machine Translation (SMT): an older but still effective method of translation. This method used Morphological Syntactic Features, also called ‘part-of-speech’ tags in conjunction with SMT which takes the language and uses their part-of-speech tags to create features for the statistical analysis of the text. The paper further explores prefix and suffix information in its mapping to make the model more powerful [2].

The other method comes from P Bahadur, A Jain, and D.S.Chauhan. They explain a general algorithm of translating between Sanskrit and English. A method of top-down processing can be used which starts with the Sanskrit text, then breaks it up into tokens and phrases, continues to parse them, takes that output and parses it into English grammar, and finally convert that phrase into English. This exact process can be reversed in order to create English-Sanskrit translation, but additional tokens need to be added with the English. [3].

III. DATASETS AND FEATURES

The first dataset we are using is from sanskritbible.in. It contains a full verse by verse translation of the New Testament into Sanskrit. We wrote a Python file called scrapeit.py to take the data from the website and query the text pairs (Sanskrit Verse, English Verse) as JSON objects. Once we had this query, we converted the individual languages and verses into their own CSV files, so we can compare both languages side by side.

<p>योलिनखितः सुसंवादः प्रथमोऽध्यायः (John Ch. 001)</p> <p>Script   </p>		<p>Parallel Translations of John Ch. 001</p> <p>Close Parallel  Toggle Inline Bibles </p>	
#	Verses	#	Verses
१:१	आदी वादः आनीतु स च वाद ईशुसः सांप्रतान्तरां स वादः सावर्णीकः स वादः	१:१-१	In the beginning was the Word, and the Word was with God, and the Word was God.
१:२	स आदानीशुसः सावर्णीकः	१:२-१	The same was in the beginning with God.
१:३	तेन सर्वं वस्तु सृज्यते साव्नी सृष्टमस्तु विभित्य विभित्य त्वेन सृज्यते नदित्ये।	१:३-१	All things were made by him; and without him was not any thing made that was made.
१:४	स जीवन्मत्सः, सत्वा जीवन्मत्सः ज्योतिः।	१:४-१	In him was life; and the life was the light of men.
१:५	आज्योतिर्मत्सः प्रमात्सः ज्योतिर्मत्सः प्रमात्सः ज्योतिर्मत्सः प्रमात्सः	१:५-१	And the light shineth in darkness; and the darkness comprehended it not.
१:६	इत्युत सत्वा जीवन्मत्सः ज्योतिर्मत्सः प्रमात्सः	१:६-१	There was a man sent from God, whose name was John.
१:७	सत्वा जीवन्मत्सः जीवन्मत्सः प्रमात्सः ज्योतिर्मत्सः प्रमात्सः	१:७-१	The same came for a witness, to bear witness of the Light, that all men through him might believe.

The data that comes from the Sanskrit query originally came in as unicode format, which we change to escape sequences (e.g., “\u028f”) using UTF-8 encoding. Each individual character is encoded this way and full words are combinations of various encodings. From here, we began the step of preprocessing the data. Our idea for preprocessing was to take the written text and convert each sentence of the text into an array of items. For example, the sentence: “The chicken crossed the street.” becomes [“The”, “chicken”, “crossed”, “the”, “street”, “.”].

We also had to go through and separate all the punctuation in the text since written language usually places words adjacent to ‘,’, ‘,’’, ‘!’’, and

other such symbol. Once we had preprocessed this data in the English text, we applied the same preprocessing onto the unicode-escaped Sanskrit file using regex.

The second data set we looked at is from holy-bhagavad-gita.org which contains a verse by verse translation of the Bhagavad Gita. We are going to be using this dataset to see if our translator properly encodes the English text into the proper Sanskrit characters and translate it back into English. To grab this data, we are looking at the individual HTML tables to get the translations and characters and query it in a similar way as we did the first data set.

Bhagavad Gita 2.1
Sanjay said: Seeing Arjun overwhelmed with pity, his mind grief-stricken, and his eyes full of tears, Shree Krishna spoke the following words.
Bhagavad Gita 2.2
The Supreme Lord said: My dear Arjun, how has this delusion overcome you in this hour of peril? It is not befitting an honorable person. It leads not to the higher abodes, but to disgrace.
Bhagavad Gita 2.3
O Parth, it does not befit you to yield to this unmanliness. Give up such petty weakness of heart and arise, O vanquisher of enemies.
Bhagavad Gita: Chapter 2, Verse 1
<p>सञ्जय उवाच । तं तथा कृपयाविष्टमश्रुपूष्णकुलेक्षणम् । विषीदन्तमिदं वाक्यमुवाच मधुसूदनः ॥ १॥</p> <p>saijaya uvācha tam tathā kṛpayāviṣṭamaśru pūṣṇakulekṣaṇam viśhidantamidaṁ vākyaṁ uvācha madhusūdanaḥ</p> <p>saijayah uvācha— Sanjay said ; tam— to him (Arjun) ; tathā— thus ; kṛpayā— with pity ; āviṣṭam— overwhelmed ; āshru-pūṣṇa— full of tears ; ākula— distressed ; īkṣhaṇam— eyes ; viśhidantam— grief-stricken ; idam— these ; vākyaṁ— words ; uvācha— said ; madhusūdanaḥ— Shree Krishn, slayer of the Madhu demon</p>
Translation
BG 2.1: Sanjay said: Seeing Arjun overwhelmed with pity, his mind grief-stricken, and his eyes full of tears, Shree Krishna spoke the following words.

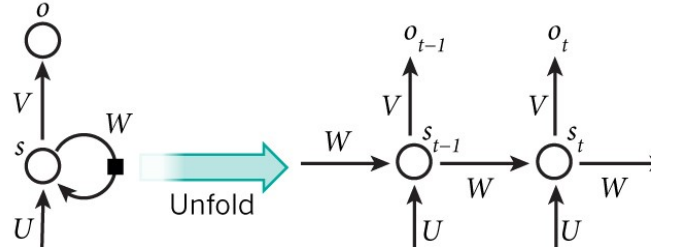
We plan to continue to diligently look for more datasets. Most of our data comes from religious texts, and these kinds of texts tend to have a specific style, grammar, and vocabulary which has some differences with common, modern vernacular (which we consider to be an important use-case of a translator). On one hand, the religious texts are a good dataset to train on since Sanskrit is mainly used to interpret these religious texts, but nevertheless, we consider expanding the scope of our data to be important.

IV. METHODS

For the method we have implemented a recurrent neural network with an LSTM layer.

A. Background

Recurrent Neural Networks are artificial neural networks that form cyclic connections between different units and are predominantly used for processing sequential knowledge [7]. In traditional neural networks, it is assumed that the input and output are independent of each other. However, this leads to complications as for instance to predict the next letter in the alphabet, the previous one has to be known. RNNs are recurrent as they perform the same task for every element in the sequence, i.e., the output is dependent on previous computation [8].



Long short-term memory (LSTM) is a type of RNN which are used to learn long term dependencies. LSTMs help maintain the error that can be backpropagated² through time and layers [14]. They were developed to solve the issue of vanishing and exploding gradients in regular RNNs, allowing for training on longer sequences of inputs. Additionally, they can decide what they keep and remove from their cell state with the help of gates which are generally composed of the sigmoid function.

The equations for an LSTM are as follows from [9]:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma(W_c x_t + U_c h_{t-1})$$

We make use of the “seq2seq” pattern for RNNs. This pattern describes a specific way of executing the neural network. While some RNN execution patterns will accept input and give output with each iteration, the seq2seq pattern splits execution into first a set of iterations for taking input, then followed by a set of iterations for generating output.

Finally, our model also takes advantage of an “attention” mechanism. With each iteration of the decoding operation, the current LSTM-layer internal vector is compared with those generated in each iteration of the encoding operation. This vector of comparisons, called the “context vector”, is combined with the current LSTM-layer state to produce an “attention vector”, which is used to produce the network output, and is given to the next iteration of the decoder [6].

V. RESULTS

Initial look at your dataset:

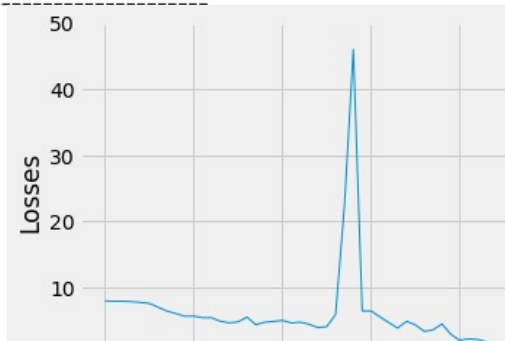
```
#inspecting data
print('Sentence in Sanskrit - encoded:', X[0])
print('Sentence in English - encoded:', Y[0])
print('Decoded:\n-----')

for i in range(len(X[1])):
    print(convert_sanskrit(en_idx2word[X[1][i]]), end = " ")

print('\n')

for i in range(len(Y[1])):
    print(de_idx2word[Y[1][i]], end = " ")
```

```
Sentence in Sanskrit - encoded: [6411]
Sentence in English - encoded: [3443]
Decoded:
```



Initially our losses rapidly fall and then there is a spike in our losses after which loss begins to fall again.

[illegible]

The translation we receive for our current model. We notice that it initially translates correctly however, it unnecessarily repeats words for an extended period.

VI. DISCUSSION

The project still requires a fair amount of work. Although our model works for now, there are still a few flaws or features we can add to the current model. Currently, the model needs to be retrained every time the code is compiled and ran. Ideally, we would save the state of the most up to date model and reload it when needed for additional training and testing. Training the model with new and more diverse datasets may also strengthen the model. Diverse training sets could include a wider variety of language translations, in addition varying sources of data. Hopefully, this may help our model handle future Sanskrit translations.

In addition to fortifying the model, we also need to implement techniques to evaluate our model. Now, we are evaluating the model solely using our knowledge of English and other languages. This may not be sufficient when attempting to translate more complex material or languages we may not understand perfectly. To evaluate the model, we plan to use the BLEU Score as our main metric. [10] The BLEU Score or Bilingual Evaluation Understudy is a way of measuring how well one group of text matches to a reference or translated text. It uses every word in the first text and tries to pair it with a word from the translated text regardless of word positioning. The score may not be perfect, but it is efficient and describes a reasonable score or metric of our model. A perfect BLEU Score would be 1.0; whereas the worst score would be a 0. However, perfect scores do not exist even with human translators. Therefore, the success of a score is only relative to previous scores.

VII. CONCLUSION

We have some initial results. However, there is significant room for improvement. We would ideally increase our dataset and train our model for a longer period. Therefore, for the next couple of

weeks, we will expend our energy in improving the current model.

VIII. APPENDIX

Shivani

- Create the model
- Wrote: Introduction, Model, Result, Conclusion

Charlie

- Create the model, Scrape data
- Wrote: Model, dataset

Shivam

- Scrape data
- Wrote: Introduction, Related Methods, Data sets, slides

Ben

- Researched potential models and sources of code to us
- Wrote: Abstract, Introduction, Discussion, Slides

Brandon

- Researched potential models and sources of code to use, Scraped dataset
- Wrote: Abstract, Slides

REFERENCES

- [1] Amrith Krishna, Pavankumar Satuluri, and Pawan Goya, *A Dataset for Sanskrit Word Segmentation*
- [2] Lee, Young-Suk, Morphological Analysis for Statistical Machine Translation (IBM)
- [3] P Bahadur, A Jain, D.S.Chauhan, English to Sanskrit Machine Translation
- [4] Priscila Aleixo, Thiago Alexandre Salgueiro Pardo, Finding Related Sentences in Multiple Documents for Multidocument Discourse Parsing of Brazilian Portuguese Texts
- [5] Rashmi Jha, Aman Jha, Deeptanshu Jha, and Sonika Jha, Is Sanskrit the most suitable language for Natural Language Processing?
- [6] Tensorflow documentation, <https://www.tensorflow.org/tutorials/seq2seq>
- [7] Ian Goodfellow and Yoshua Bengio and Aaron Courville. "*Deep Learning*." MIT press (2016)
- [8] Britz, Denny. "*Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*." WildML. N.p., 08 July 2016. Web. 30 July 2017.
- [9] Hochreiter, Sepp, and Jürgen Schmidhuber. "*Long Short Term Memory*." Neural
- [10] Papineni, Kishore, et al. "Bleu: a Method for Automatic Evaluation of Machine Translation." *Achweb.org*, IBM T. J. Watson Research Center, July 2002, www.aclweb.org/anthology/P02-1040.pdf. Computation, 1997. Web.