Cambridge Large Two (2020), 1–4

CAMBRIDGE
UNIVERSITY PRESS

ARTICLE

Neural Unsupervised paraphrasing

M Sathvika,[1] M Nandini,[2] and D Lakshmi Girija[3]

[1]2020101087, UG3, CSE - BTech
[2]2020101038, UG3, CSE - BTech
[3]2020101027, UG3, CSE - BTech

Abstract

This project presents an approach to Neural Unsupervised Paraphrasing for Indian Languages, a challenging task due to limited resources and inadequate embedding models. We utilize various models including GRU, LSTM, Attention to achieve their results. The implementation of basic RNN-derived models, machine translation and back translation are also explored. Through experimentation, we demonstrate the effectiveness of different approaches, which has the potential to improve language technologies for Indian languages and contribute to the field of natural language processing. This project highlights the importance of developing effective language models for languages with limited resources and provides insights into the methods and techniques that can be applied to achieve this goal.

Keywords: LSTM and GRU, Machine Translation and Back translation, Attention

We show how to get data for paraphrasing for Indian languages and publish the results of all the above methods

1. Introduction

Paraphrasing is a way in which we preserve the original meaning of the sentence but in a different manner. It involves understanding the original text, identifying its main points and ideas, and then expressing them in one's own words, while ensuring that the meaning remains unchanged. Since most of these systems were created for languages with high resource requirements, the focus must be on low-resource languages. The majority of studies have focused on the rephrasing of high-resource languages, with only a limited amount of research on low-resource languages. One barrier to solving this issue is the lack of corpora in languages with scarce resources. Thorough study of low-resource languages is more appealing due to the possible benefit it may have on the population of people who find high-resource languages complex. As a result, we attempt to create a paraphrase model for Indian languages in this study employing recurrent neural networks of the Long Short Term Memory (LSTM) and the Gated Recurrent Unit(GRU), machine translation using attention. We are mainly focused on Hindi and Telugu languages.

1.1 Literature Review

Previous works done in this direction for Indian languages were of Variational Auto Encoders and of LSTM, GRU with adaptive attention. They gave a bleu score of around 32.42 percent for trigrams and a meteor score of 37.68 percent. They have used MSCOCO dataset along with quora similar question answer pairs and have translated the dataset to hindi using google translate api. The data is limited and the dataset isn't as reliable as they are relying solely on google translate which may or may not provide accurate results as said by authors themselves.

There has been some work by Akshara et al for back translation for hindi and bengali languages and they got a bleu score of 33.72 for Hindi and 11.89 for bengali. They used encoder decoder models and have used transformers to implement it. They have used Samanantar parallel corpus for the data.

1.2 Dataset Extraction

1st dataset: There aren't any high-quality datasets that are suitable for our project because Hindi and Telugu are low-resource languages. As a result, we are using the MSCOCO(Microsoft Common Objects in Context) dataset, which contains 4,5 sentences, to describe the photos in English.By presuming that the translations produced by Google Translate API are accurate and adequate for ground truth for training our model for paraphrasing in Hindi and Telugu, we are translating these descriptions into Hindi and Telugu utilising that service. 2nd dataset At Kaggle, we are utilising the Telugu NLP dataset.We translate news headlines about various topics like weather, movies, and health into English and then into Telugu using the Google Translate API.The models are then trained using these two phrases, which are equivalent in both languages.This datasets also extracted same way as above method by doing two times

2. The Models

1. `LSTM`
2. `LSTM Encoder-Decoder`
3. `LSTM Attention`
4. `GRU Attention`
5. `MTBT - LSTM`
6. `MTBT - GRU Attention`

2.1 LSTM (Long short term memory)

An LSTM word to word model which takes input, encodes the required information in Hidden Layer. It takes as input the Hindi words from MSCOCO dataset and then takes the output as one of it's paraphrases. The model basically has an

input gate, output gate and forget gate to store information for long term dependencies.

It follows the following architectures:

$$l_t = \sigma(W_l(h_{t-1}, x_t) + b_l) \tag{1}$$

$$\tilde{c_t} = \sigma(W(h_{t-1}, x_t) + b) \tag{2}$$

$$c_t = i_t\tilde{c_t} + f_t c_{t-1} \tag{3}$$

$$h_t = o_t * tanh(c_t) \tag{4}$$

Here l can be f,i,o based on the forget, input and output gates.

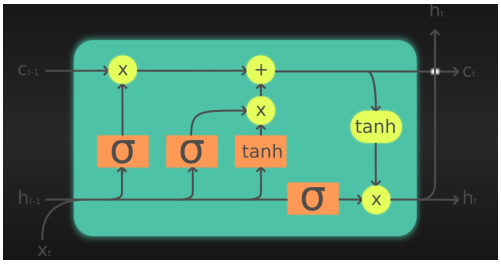

Figure 1. The LSTM model

2.2 Encoder Decoder - LSTM

We use encoder – decoder sequence to sequence model here. In the encoder model we obtain the context of the original sentence through the final hidden state of the LSTM. The context vector is then passed to the decoder. It takes the context and previously generated output to give us the current output. We do this on MSCOCO dataset for Hindi. This model generates context and retains it over the course of obtaining the sequence over the decoder.

The LSTM is a function of the context vector and the previous output and the previous hidden state.

$$h_t = g(h_{t-1}, y_{t-1}, c) \tag{5}$$

$$y_t = softmax(W * h_t) \tag{6}$$

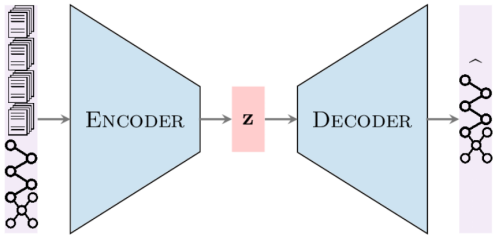

Figure 2. The Encoder Decoder model

2.3 LSTM Attention

Here we have a encoder decoder sequence to sequence model with attention. In the previous model the context vector c became the bottle neck as a single vector stores all the information regarding the context. So we introduce attention and all the vectors in the final hidden layers are given a weight α_{ij}.

The weights are calculated by the similarity between the hidden decoder vector and the given hidden state of the encoder. Then we apply softmax over our output. These weights will then be multiplied by the hidden state and added to get our new context.

$$\alpha_{ij} = \frac{e^{(he_i.hd_{j-1})}}{\Sigma_i e^{(he_i.hd_{j-1})}} \tag{7}$$

$$c_j = \Sigma_i(\alpha_{ij} * he_i) \tag{8}$$

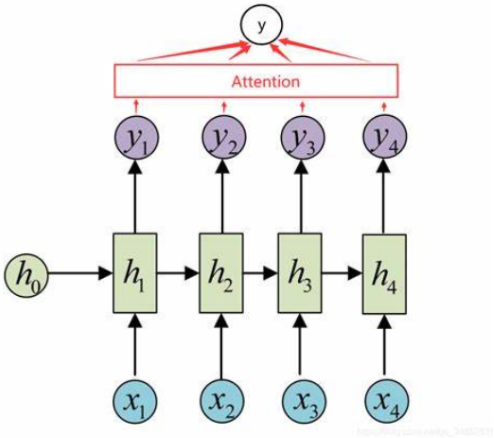

Figure 3. Encoder with Attention

2.4 GRU Attention

In this model instead of LSTMs at the encoder and decoder, we use the gated recurrent unit. This model uses an update gate instead of input gate and forget gate. It also uses attention.

2.5 MTBT - LSTM

Here we implemented a Machine translation – back translation model. This involves translating telugu data using Bi-directional LSTM to english on MT model and the back translating it to telugu from english on a BT model. This model is a combination of two models – MT and BT. The models output is the translated text. Once we send the translated text from MT to BT, we get a twice-translated text. The twice translated text is a paraphrase of the original text.

2.6 MTBT - GRU Attention

Here we implement MTBT using GRU Attention for translating on both MT and BT models. This ensures we get a contextualised representation from both translation models and we get an accurate translated text.

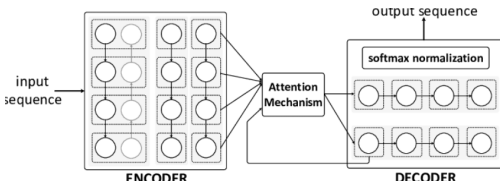

Figure 4. Seq2Seq Machine translation

3. Evaluation Methods

1. METEOR METEOR stands for Metric for Evaluation of Translation with Explicit ORdering, which utilizes precision, recall, and alignment in combination to measure the similarity between two text passages.METEOR computes a score based on the weighted harmonic mean of unigram precision and recall, with recall being weighted more heavily than precision. Additionally, it takes into account the stems, synonyms, and paraphrases of words to improve the accuracy of the metric.
2. BLeU BLEU (Bilingual Evaluation Understudy) is a metric used to evaluate the quality of machine-generated translations by comparing them to a set of reference translations. It is a common metric used in natural language processing tasks such as machine translation, text summarization, and paraphrasing.To calculate the BLEU score, we first calculate the precision of the candidate paraphrase for each n-gram (a contiguous sequence of n words) in the reference paraphrase. We then take the geometric mean of the n-gram precisions, giving more weight to higher n-grams. Finally, we apply a brevity penalty to adjust for cases where the candidate paraphrase is shorter than the reference paraphrase.

4. Results

The results of models trained on hindi dataset 1.

Table 1. Quantitative analysis

Model	Meteor score	bleu score
LSTM(attention	22.6997256894	20.2354589612
GRU Attention	25.5789655677	21.3548795245
Encoder-decoder	0.5602536969	0.2056894513
LSTM	0.01524784575	0.00654721212
Translation	15.235811235	6.9865473286

Table note
a METEOR Score
b Bleu Score

The following are the epoch loss vs number of epochs for the training data.

5. Hyperparameters

`Optimizer : Adam`

`Loss function : CrossEntropyLoss`

1. `LSTM`

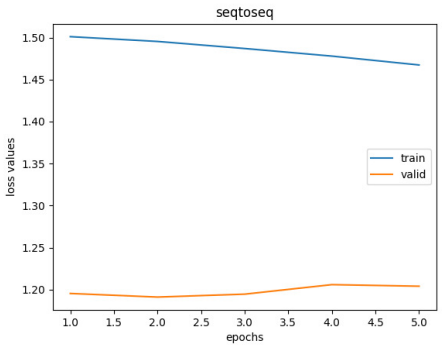

Figure 5. Encoder Decoder for LSTM

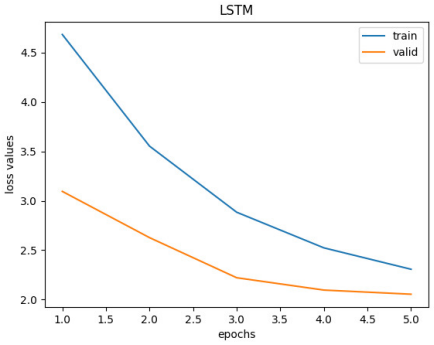

Figure 6. LSTM

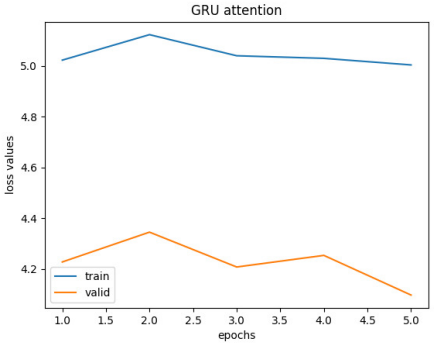

Figure 7. GRU Attention

Epochs = 5
Learning Rate = 0.01
Batch Size = 128
Hidden Size = 200
Embedding Size = 300

2. `LSTM Encoder Decoder`
   Epochs = 5
   Learning Rate = 0.01
   Batch Size = 128
   Embedding dim = 256
   Hidden dim = 256
   Num layers = 1
3. `LSTM Attention`

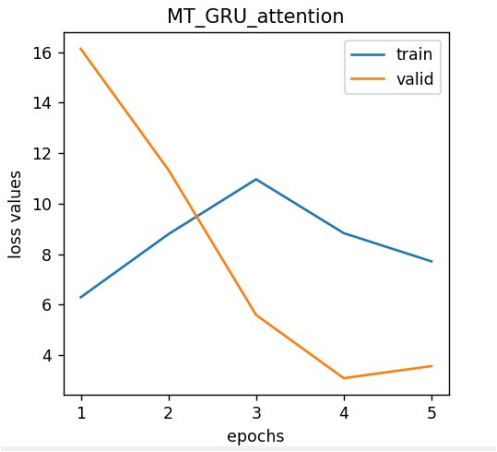

Figure 8. MT$_{G}RU_{A}ttention$

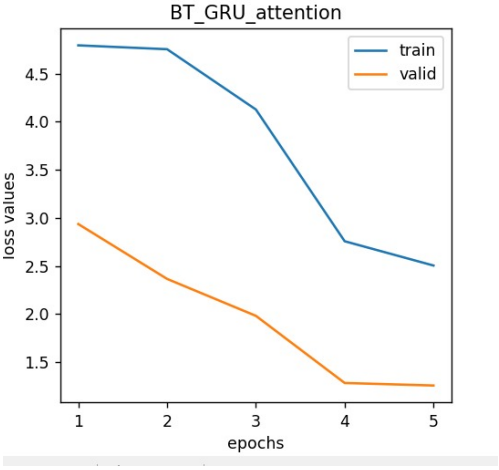

Figure 9. BT$_{G}RU_{A}ttention$

Epochs = 5
Learning Rate = 0.01
Batch Size = 128
Embedding dim = 256
Hidden dim = 256
Num layers = 1 - Bidirectional with LSTM Attention

4. `GRU Attention`
   Epochs = 5
   Learning Rate = 0.01
   Batch Size = 128
   Embedding dim = 256
   Hidden dim = 256
   Num layers = 1 - Bidirectional with GRU Attention
5. `Machine Translation`
   Epochs = 5
   Learning Rate = 0.01
   Batch Size = 128
   Embedding dim = 256
   Hidden dim = 256
   Num layers = 1 - Bidirectional with GRU Attention

6. References

https://arxiv.org/pdf/1709.05074.pdf
https://kalaharijournals.com/resources/DEC$_{5}$44.*pdf*
https://www.kaggle.com/datasets/sudalairajkumar/telugu-nlp
https://cocodataset.org/download

7. Conclusion

Hence for the Hindi dataset, GRU Attention shows the best result followed by LSTM Attention. LSTM and encoder-decoder models do not show much of a progress as it lacks attention to the context.In general, GRU attention is computationally faster than LSTM attention because the GRU has fewer parameters than LSTM.In LSTM, there are three gates (input, forget, and output) and a memory cell, whereas in GRU, there are only two gates (reset and update) and no separate memory cell. This means that GRU has fewer parameters to learn than LSTM, making it computationally faster and requiring less training time.Classical seqtoseq encoder-decoder model performs poor when the input sequence lengths large ,because whole sentence information is stored in context vector and because of longer sequence useful information from the starting is not captured properly.

8. Qualitative Analysis

MT:
['ఈ', 'కాలము', 'భారతదేశము', 'ఓ', 'పెద్ద', 'సాంఘిక', 'మార్పు', 'చవిచూసినది.']
['<s>', 'shiya', 'fasting,Knowledge', 'workstations,', 'demand', 'teginchi', 'fasting,Knowledge', 'demand', 'demand', 'demand', 'demand']
BT
['<s>', 'shiya', 'fasting,Knowledge', 'workstations,', 'demand', 'teginchi', 'fasting,Knowledge', 'demand', 'demand', 'demand', 'demand']
['బాగుంది, '<pad>', 'ఓ', 'కాలం', '<pad>', 'ఓ', 'సంత', '<pad>', 'రైలు', 'ఇంకా, 'ఓ']