Tamil Paraphrase Detection Using Encoder-Decoder Neural Networks

B. Senthil Kumar, D. Thenmozhi, S. Kayalvizhi

TAMIL PARAPHRASE DETECTION USING ENCODER-DECODER NEURAL NETWORKS

Senthil Kumar B, Thenmozhi D, Kayalvizhi S

Department of CSE, SSN College of Engineering, Chennai
{senthil,theni_d,kayalvizhis}@ssn.edu.in

Abstract. Detecting paraphrases in Indian languages require critical anlysis on the lexical, syntactic and semantic features. Since the structure of Indian languages differ from the other languages like English, the usage of lexico-syntactic features vary between the Indian languages and plays a critical role in determining the performance of the system. Instead of using various lexico-syntactic similarity features, we aim to apply a complete end-to-end system using deep learning networks with no lexico-syntactic features. In this paper we exploited the encoder-decoder model of deep neural network to analyze the paraphrase sentences in Tamil language and to classify. In this encoder-decoder model, LSTM, GRU units and gNMT are used as layers along with attention mechanism. Using this end-to-end model, there is an increase in f1-measure by 0.5% for the subtask-1 when compared to the state-of-the-art systems. The system was trained and evaluated on DPIL@FIRE2016 Shared Task dataset. To our knowledge, ours is the first deep learning model which validates the training instances of both the subtask-1 and subtask-2 dataset of DPIL shared task.

Keywords: Tamil Paraphrase Detection · Deep Learning · Encoder-Decoder Model · Sequence-to-sequence (Seq-2-Seq).

1 Introduction

Recent advances in deep neural network architecture has motivated many researchers in natural language processing to apply for different tasks such as PoS, language modeling, NER, relation extraction, paraphrase detection, semantic role labeling etc., Paraphrase detection is one of primary and important task for many NLP applications. The two sentences which convey the same meaning in a language are said to be semantically correct or paraphrases. Paraphrases can be detected, extracted and can be generated. Paraphrase detection is an important task in paraphrase generation and extraction system. In paraphrase generation, the paraphrase detection improves the quality by picking up the best semantic equivalent paraphrase from the list of generated sentences. In paraphrase extraction, the detection of paraphrase plays a vital role in validating the extracted sentences. Apart from this, paraphrase detection is also utilized in document summarization, plagiarism detection, question-answering

system etc.,

Paraphrases in sentences are detected in earlier systems using different traditional machine learning techniques. Recently paraphrase detection was explored in some regional Indian languages. One of the shared task which was first of its kind for Indian languages – Detecting Paraphrases in Indian languages – DPIL@FIRE2016[9][10] highlighted the performance of different systems using traditional machine learning techniques. The issues associated with these systems are: (1) they are dependent on the hand-crafted feature engineering which used pure lexical, syntactic and semantic features (2) the features are language-dependent (3) the morphological variations in Indian languages increase the complexity in detection and (4) the lack of resources such as WordNet, word2vec pre-trained embeddings for Indian languages deterred the systems from detecting the sentencial paraphrases. These limitations can be addressed by using the deep neural networks which require less or no features to build the model and is language independent.

Our main contributions in this paper are: (a) the use of encoder-decoder model to build paraphrase detection system for Tamil language which require no lexico-syntactic features. This is an end-to-end model. (b) the system is evaluated in a monolingual setting to detect the paraphrases in Tamil using DPIL-FIRE2016 shared task dataset. (c) we compared the performance of this system with the state-of-the-art paraphrase detection system for Tamil using deep neural networks.

2 Related work

The shared task on Detecting Paraphrases in Indian Languages DPIL@FIRE 2016[9][10] was a good effort towards creating a benchmark data for paraphrases in Indian Languages – Hindi, Tamil, Malayalam and Punjabi. Paraphrase detection for Indian languages are more challenging and complex due to its morphological variations and Tamil, Malayalam belongs to Dravidian language family which are more agglutinative in nature than Hindi and Punjabi from Indo-Aryan language family. Totally eleven teams submitted their results out of which only four teams evaluated the results for all the four Indian languages. In general most of teams used traditional machine learning techniques such as Logistic Regression, Random Forest, Gradient Tree Boosting, Multinomial Logistic Regression, Maximum Entropy and Probabilistic NN models. The commonly used lexical features are PoS, Lemma, Stopwords, word overlaps and the similarity features are Cosine, Jaccard coefficient,Levenshtein distance measures and the semantic feature like synonyms are used.

It was observed from the four teams which submitted their results for all the four languages that, there was a huge difference in the f1-score and accuracy across the category or family – Dravidian and Indo-Aryan – of languages. This

revealed that the morphological inflections and agglutinative characteristics of Dravidian language family deter the performance of the system. HIT2016 [7] which scored top result for all languages other than Hindi, submitted their results based on the Cosine similarity, jaccard coefficient, METEOR – a machine translation-based metric for the character n-grams. The accuracy of the system was 89% and 94% for Hindi and Punjabi respectively. For the Dravidian languages, the accuracy obtained was 81% and 79% for Malayalam and Tamil respectively. This clearly indicate that the features to be used in detecting the paraphrases are dependent on the language characteristics. Similarly the three teams JU_NLP [15], KS_JU[17] and NLP_NITMZ [17] which submitted their results for all the four languages had showed the difference in accuracy across the family of languages for their lexico-syntactic features. In general, the accuracy of Tamil and Malayalam are lesser when compared to the accuracy obtained by the Hindi and Punjabi languages for the similar lexico-syntactic features.

Another team Anuj [16] submitted his result only for Hindi language without using any similarity measures but by using only the lexical features such as lemma, stop word removal and synonym, yielded the better result than the team which used the similarity-based features. This clearly highlights that the similarity features, lexical or semantic features used to detect the phraphrases are dependent on the natural languge characteristics. Moreover there is a lack of resources such as WordNet, word2vec pre-trained embeddings for Indian languages. To overcome the shortcomings of traditonal machine learning techniques which depends on hand-crafted feature engineering, some attempts were made to develop models using deep neural networks for Indian languages. This greatly reduced the need of any lexical or syntactic resources for the regional languages. One such attempt for Tamil was by Mahalakshmi et al.,[13], Bhargava et al.,[2] and Senthil Kumar et al.,[18]. Mahalakshmi et al.,[13] used the recursive auto-encoders (RAE) to compute the word representations of the paraphrase sentences.

The model then used the euclidean distances to measure the similarity and then softmax classifier is used to detect the paraphrases. They evaluated the training dataset and obtained the accuracy of 50%. Bhargava et al., attempted to use three deep neural networks model for detecting paraphrases in four Indian languages – Tamil, Malayalam, Hindi, Punjabi – and English. They used CNN, CNN using WordNet and stacked LSTM models. It was found that all the systems performed well for English than the Indian languages. Among the Indian languages Hindi, Punjabi paraphrase sentences were detected better than Tamil and Malayalam. They reported the evaluation of subtask-1 dataset of all the Indian languages. For Tamil, the CNN model outperformed the other models and scored 74.3% of f1-measure. Senthil Kumar et al., used the deep Bi-LSTM-layered networks and evaluated the model with two attention

mechanisms. The model was evaluated for 5-fold cross validation of training dataset and obtained the accuracy of 65.2%.

3 Dataset Description

The dataset used to train and evaluate our system is the dataset released by the DPIL@FIRE 2016 shared task for four Indian languages. The shared task required participants to identify the sentential paraphrases in four Indian languages, namely Hindi, Tamil, Malayalam, and Punjabi. The corpora mainly contains the news articles from various web-based news sources. The shared task organized two subtasks: Subtask 1- to classify the given paraphrases into Paraphrase(P) or Non-Paraphrase(NP). Subtask 2- to classify the given paraphrases into Paraphrase(P), Semi-Paraphrase(SP) or Non-Paraphrase(NP). The corpora contains two subsets, one for each subtask. For subtask-1, the Tamil corpus was classified into one of binary class for the 2500 sentencial paraphrases and the 900 test pairs. For the subtask-2, the corpus contain 3500 sentential Tamil paraphrases and the 1400 test pairs. The analysis on the corpus showed that the average number of words used in the paraphrase sentences are less for the aggultinative languages. This is because of the more morphological inflections in the agglutinative languages. This lead to more unique words and hence larger vocabulary size. These morphological variations and agglutinative characteristics play vital role in determining the performance of the paraphrase detection system for agglutinative language like Tamil.

Table 1. Data population for Subtask-1

Sub task - 1	**Training**	**Testing**	**Total**
Paraphrase	1000	400	1400
Non-paraphrase	1500	500	2000
Total	2500	900	3400

Table 2. Data population for Subtask-2

Sub task - 2	**Training**	**Testing**	**Total**
Paraphrase	1000	400	1400
Non-paraphrase	1500	500	2000
Semi-paraphrase	1000	500	1500
Total	3500	1400	4900

<Paraphrase pID="TAM0071">
<Sentence1>
ஜி.எஸ்.டி. மசோதாவை நிறைவேற்ற எதிர்க்கட்சிகளின் ஆதரவை பெற முயற்சிப்போம் என்றார் வெங்கையா நாயுடு.
</Sentence1>
<Sentence2>
ஜி.எஸ்.டி. மசோதாவை நிறைவேற்ற எதிர்க்கட்சிகளின் ஆதரவைப் பெற திட்டமிட்டுள்ளதாக வெங்கையா நாயுடு தெரிவித்துள்ளார்.
</Sentence2>
<Class>**P**</Class>
</Paraphrase>

Fig. 1. A pair of Tamil paraphrase sentences in the dataset

Figure 1 shows an example of paraphrase sentences from the DPIL@FIRE2016 dataset which is tagged as P category. The dataset was annotated with one of the paraphrase category (P, SP, NP) and in the XML format. Each paraphrase was allocated with a unique ID and the pair of sentences are separated and annotated with the corresponding paraphrase tag.

4 Preprocessing

The pair of sentences from the dataset are extracted and presented to the network as shown in Figure 2. The pair of sentences are appended together. The two sentences are delimited with the tag <eol> which acted as separator between the pair of sentences and <s>, </s> are used to denote the boundary of the pair of input sentences. These sentences are given as input to the encoder-decoder model as input sequences.

<s> ஜி.எஸ்.டி. மசோதாவை நிறைவேற்ற எதிர்க்கட்சிகளின் ஆதரவை பெற முயற்சிப்போம் என்றார் வெங்கையா நாயுடு . <eol> ஜி.எஸ்.டி. மசோதாவை நிறைவேற்ற எதிர்க்கட்சிகளின் ஆதரவைப் பெற திட்டமிட்டுள்ளதாக வெங்கையா நாயுடு தெரிவித்துள்ளார் . **</s>**

Fig. 2. A preprocessed input sentences w_1, ... w_n

5 Encoder-Decoder Model

To detect the Tamil paraphrases using encoder-decoder model we have adopted the Neural Machine Translation architecture [11] which was based on the sequence-to-sequence learning [3][19]. The sequence-to-sequence(seq-2-seq) models were pioneered by Ilya Sutskever[15], applied that model for machine translation (MT) task. This is the most promising model in natural language processing (NLP) tasks such as sequence labeling [4][20], machine translation [11], syntactic parsing [21][8], semantic parsing [5]. In our approach we treated the

encoder-decoder model as the classifier to classify the source sequence of words to the corresponding paraphrase label as the target. This encoder-decoder model or sequence to sequence (seq-2-seq) model is the most popular model in learning the target sequence conditioned on the source sequence. This model typically uses the encoder to encode the given input sequence into a fixed vector representation which is then fed into the decoder. The decoder then predicts the next word given the source sequence representation, the previous output and the current hidden state. The decoder predicts the next word conditioned on the previous input sequences. We treated the paraphrase detection as the classification problem. Given the pair of paraphrases as input, the model has to predict the class label. The encoder and decoder uses a multilayered RNN variant units to map the input sequence to a vector of a fixed dimensionality, and then decode that to the target label from that vector.

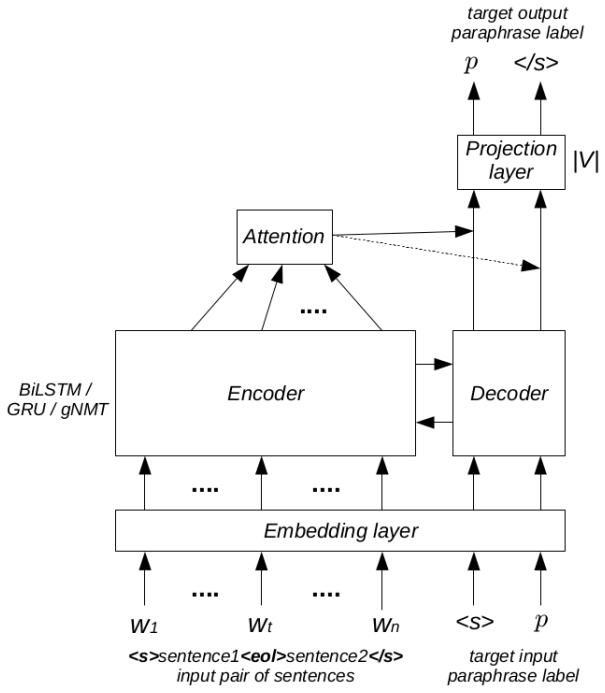

Fig. 3. Encoder-Decoder model for paraphrase detection

The model has an embedding layer, an encoder, a decoder and a projection layer. The encoder and the decoder receives the following input: first the source sentence, then a boundary marker which indicates the transition from the encoding to the decoding mode, then the target input and target output class label. Here the source sentence is the input sequence which is a pair

of paraphrase sentences with a delimiter and the target is the corresponding class label. The system is feed with those sentences in tensors, which are in time-major format. Once the source sentences and target label are given, the model first look up its word embeddings to retrieve the corresponding word representations. The vector representation for each word is derived by choosing a fixed vocabulary of size V for source and target sequences. The embedding weights are usually learned during training. Since the input is time major, the encoder receives the input till the boundary marker. At each time step t, the hidden state $h_{<t>}$ of the encoder is updated by, $h_{<t>} = f(h_{<t-1>}, wt)$, where f is a non-linear activation function, wt is the source sentences. Here in our case wt is the word sequences from the pair of input sentences.

Once the source sentences are encoded, the last hidden state of the encoder is used to initialize the decoder. Once the boundary marker reached, it marked the beginning of the decoder. The hidden state of the encoder is a vector of fixed dimensionality $\mathbf{s}$ – which is a summary of the source sentences. Given the hidden state $\mathbf{h}_{<t>}$, the decoder predicts the next word $\mathbf{v}_t$. However $\mathbf{h}_{<t>}$ and vt are conditioned on the previous output $\mathbf{v}_{t-1}$ and on the summary s of the input sequence. Hence the hidden state of the decoder at time t is computed by, $\mathbf{h}_{<t>} = f(\mathbf{h}_{<t-1>}, \mathbf{v}_{t-1}, \mathbf{s})$, and the conditional distribution of the next symbol is, $P(v_t|\mathbf{v}_{t-1}, ..., \mathbf{v}_1, \mathbf{s}) = g(\mathbf{h}_{<t>}, \mathbf{v}_{t-1}, \mathbf{s})$, for given activation functions f and g. Here in our case, the previous output v_{t-1} is the boundary marker of the paraphrase sentence and the v_t is the paraphrase label to be predicted by the model. Hence the Encoder-Decoder are jointly trained to maximize the conditional log-likelihood, max_θ 1/N $log\ p_\theta\ (\mathbf{v}_n|\mathbf{w}_n)$ where $(\mathbf{w}_n, \mathbf{v}_n)$ is the pair of paraphrase sentences and target label from the training set, θ is the varying model hyper parameters.

The projection layer is feed with the tensors of the target output words, these are decoder inputs shifted to the left by one step with a boundary marker appended on the right. Based on the previous source hidden state and target input, for each timestep on the decoder side, the layer outputs a maximum logit value. The projection layer is the dense matrix to turn the top hidden states of decoder, to logit vectors of dimension V. Given the logit values, the training loss can be easily computed by using standard SGD/Adam optimizer by varying the learning rate.

The model was also trained with the attention mechanism, which computes the attention weight by comparing the current decoder hidden state with all encoder states. This attention weight helps in identifying the sequence of source input words that helps in predicting the target label. Instead of discarding all the source hidden states, attention mechanism provides direct connection between source sentences and target label by paying attention to the relevant source sequences. We have used both scaled-luong [12] and normed-Bahdanau[1] attention mechanisms. There are several key differences between the attention

model proposed by Bahdanau et al.,[1] and Luong et al.,[12]. The primary difference is that the Luong attention simply uses hidden states at the top LSTM layers in both the encoder and decoder, where as the Bahdanau attention use the concatenation of the forward and backward source hidden states in the bi-directional encoder and target hidden states in their decoder. For M:N sequence mapping tasks such as sequence labelling, machine translation task, Luong attention mechanism performs better as the word alignment is required during the translation by the decoder from the encoder. Since our task is to map the n word sequences to the paraphrase label which is N:1 mapping, Bahdanau attention produces better result as only the encoder attention weight is required to predict the paraphrase label.

The encoder-decoder model basically was built by using RNN units in the stacked layers. It can also be built based on the variants of RNN units such as LSTM, Bi-LSTM, GRU and gNMT units[11]. As shown in Figure 3, we used GRU and gNMT units in the encoder-decoder layers to detect the Tamil paraphrases.

6 Experiment

There were earlier attempts using LSTM, Bi-LSTM as stacked layers in the encoder-decoder models to predict the Tamil paraphrase sentences[2][17]. For some of the tasks such as machine translation[11], GRU and gNMT model performed better than the RNN variants – LSTM, Bi-LSTM. This motivated us to use the other RNN-variant GRU and gNMT in the stacked layers of encoder-decoder model. On fine tuning the hyper parameters of the model, we found that 2-layered GRU and 5-layered gNMT – 1 Bi-LSTM with 3 residual layers – at both encoder-decoder performs better than the other for the subtask-1. These two models were trained and evaluated on the DPIL@FIRE 2016 dataset for the Tamil language. Both the systems were trained and validated for the training data of both subtask-1 and subtask-2. The subtask-1 contains 2500 pairs of training sentences and 900 pairs of testing sentences. Out of which only 40% are paraphrase pairs, the remaining 60% pairs of sentences are non-paraphrase pairs. The subtask-2 contains 3500 paraphrase pairs and 1400 testing instances. For the subtask-2, we experimented with all the variations in encoder-decoder models using Bi-LSTM, GRU and gNMT model. The models are fine-tuned with various options and hyperparameters. The gNMT model for the subtask-1 performed better with the SGD as optimizer, 128 LSTM units, learning rate 1.0, initial weight was set to 0.1 with forget_bias 1.0 to guard against the vanishing gradient, batch size 128, dropout is 0.2, with the normed-bahdanau attention mechanism. The model was trained and tested on using single GPU.

7 Results

The system was evaluated for the training instances of the subtask-1 and subtask-2. The existing deep learning systems for Tamil paraphrase detec-

tion[13][11][18] reported their performance only for the subtask-1 training instances. From the conventional state-of-the-art systems using DPIL dataset, the HIT2016 [7] scored high across all the languages which used Cosine similarity as one of its feature, the Indo-Aryan family languages obtained an average accuracy of 91.5% whereas the Dravidian languages obtained 80% only. The cosine similarity feature used for Indo-Aryan family could not perform well for the Dravidian languages, since it is agglutinative. Hence there is a need for system which identifies the feature or model specific to Dravidian languages.

To our knowledge, ours is the first end-to-end deep learning model which validates the training instances of both the subtask-1 and subtask-2 dataset. We developed two variations with respect to attention techniques on the deep neural network – system using scaled-luong (SL) , normed bahdanau (NB) as attention mechanisms. We performed 5-fold cross validation on the training instances of both the subtask-1 and subtask-2 datasets. In subtask-1, for the given 2500 instances, each fold the data is split into 2000 pairs of sentences as the training instances and 500 pairs of sentences as testing instances. Our split for the subtask-1 is similar to Mahalakshmi et al., [13]. For subtask-2, each fold contains 3000 instance pairs as the training and 500 pairs of sentences as the testing instances. Our model for subtask-1 was treated as the binary classifier which predicts P/NP class. For the subtask-2 our model was treated as the multi-class classifier which predicts P/SP/NP class. The performance of the models for both the subtask-1 and subtask-2 was measured by precision, recall and f1-measure metrics. We compared our different models – GRU, gNMT variants – with the existing deep tamil paraphrase detection systems. For the subtask-1, gNMT-layered encoder-decoder model obtained 0.5% improvement than the state-of-the-art systems.

Table 3. Cross-validation performance of subtask-1

Subtask-1 Models	**Precision**	**Recall**	**Accuracy**	**F1-score**
GRU-NB	74.46	52.24	67.16	61.40
GRU-SL	72.78	54.96	67.2	62.3
gNMT-SL (1 Bi-LSTM +1 residual)	75.84	61.76	71.04	68.08
gNMT-NB (1 Bi-LSTM +1 residual)	72.98	67.84	71.36	70.32
gNMT-NB (1 Bi-LSTM +3 residual)	**78.14**	**71.76**	**75.04**	**74.81**
Mahalakshmi et al., [13]	-	-	50.0	-
Bhargava et al., [2]	-	-	-	74.3
Senthil Kumar et al., [18]	-	-	65.2	-

Mahalakshmi et al., [13] reported the overall accuracy of the system detecting the Tamil paraphrases by 50% and Senthil Kumar et al., [18] used

Bi-LSTM-layered encoder-decoder model to detect the Tamil paraphrases and performance in accuracy was reported as 65.2%. Bhargava et al., [2] used the three models and the best CNN-based deep neural network model was measured in f-measure of 74.3%. The following Table 3, shows the comparison of subtask-1 dataset evaluation of the existing deep learning models with our encoder-decoder models.

For the subtask-2, we experimented with all the variants of RNN units – BiLSTM, GRU and gNMT units in the encoder-decoder layers with scaled-luong and normed-bahdanau attention mechanisms. 2-layered BiLSTM with scaled-luong attention has higher accuracy than the other encoder-decoder models for subtask-2 as shown in Table 4. When comparing with the subtask-1 models which is binary classification, gNMT models are not performing well for the multi-class classification.

Table 4. Cross-validation performance of subtask-2

Subtask-2 Models	**Precision**	**Recall**	**Accuracy**	**F1-score**
2 BiLSTM-SL	**64.52**	**62.74**	**75.42**	61.76
2 BiLSTM-NB	61.98	61.06	74.45	60.58
GRU-SL	63.56	62.54	75.05	**61.81**
GRU-NB	63.91	62.37	74.66	61.71
gNMT-SL (1 Bi-LSTM +1 residual)	60.55	58.98	72.66	58.11
gNMT-NB (1 Bi-LSTM +1 residual)	61.39	60.16	73.33	59.48
gNMT-NB (1 Bi-LSTM +3 residual)	60.55	60.5	73.39	60.49

For the test data, HIT2016 - a traditional learning system scored top result in the shared task obtained 82.11% for the subtask-1, our gNMT-based deep learning system obtained 55.6%. For the subtask-2, HIT2016 obtained 75.5% whereas our GRU-based obtained 49.2% which is low compared with the traditional machine learning model. Data sparsity is one of the main reason behind the low performance of deep neural networks models.

8 Error Analysis

From Table 1, the number of non-paraphrase sentence pairs are 60% whereas the paraphrase sentence pairs are only 40%. Table 5 shows the class-wise prediction of the sentence pairs by the different models. There is a huge difference in predicting the paraphrase sentences by the different models from 65.3-89.7%. Where as the non-paraphrase sentence pairs prediction by all the models are approximately between 65 – 68%. Eventhough the number of non-paraphrase sentences are higher, the system performance depends only on predicting the

paraphrase sentence pairs. There is a drastic increase around 24% in detecting the paraphrase sentence pairs across the models when compared with the non-paraphrase sentence pairs detection with only around 3%. From the Table 2,

Table 5. Class-wise prediction (in %) for subtask-1

Subtask-1 Models	**P**	**NP**
GRU-NB	65.3	**68.40**
GRU-SL	68.7	66.2
gNMT-SL(1Bi-LSTM + 1 residual)	77.2	66.93
gNMT-NB(1Bi-LSTM + 1 residual)	84.8	62.40
gNMT-NB(1Bi-LSTM + 3 residual)	**89.7**	65.27

it is clear that 43% of sentences are non-paraphrase sentences where as around 28.5% of each are paraphrase and semi-paraphrase sentence pairs in the training dataset. Table 6 shows the class-wise f1-score across all the models. To calculate the class-wise performance, the multiclass data will be treated as if binarized under a one-vs-rest transformation [6]. In this subtask-2 also, even though the number of non-paraphrase sentence pairs are more when compared to the paraphrase and semi-paraphrase pairs, the performance of the system highly depends on the detection of paraphrase and semi-paraphrase sentence pairs. The average detection of P and SP sentence pairs are 63.39% and 63.82% f1-score respectively, which are more when compared to the average detection of NP sentence pairs with f1-score of 54.65% only, which is almost 9% less when compared to the P and SP detection. Hence the above analysis indicates that the performance

Table 6. Class-wise prediction (in f-score) for subtask-2

Subtask-2 Model	**P**	**NP**	**SP**	**F1-score**
2 BiLSTM-SL	64.43	**56.70**	64.16	61.76
2 BiLSTM-Nb	**65.45**	53.56	62.71	60.58
GRU-SL	64.47	54.75	**66.20**	**61.80**
GRU-NB	63.51	55.68	65.93	61.71
gNMT-SL(1Bi-LSTM + 1 residual)	60.53	52.19	61.58	58.11
gNMT-NB(1Bi-LSTM + 1 residual)	62.34	52.99	63.10	59.48
gNMT-NB(1Bi-LSTM + 3 residual)	63.01	56.69	63.62	60.49
Average	63.39	54.65	63.82	

of the system depends only on the P, P & SP sentence pairs in subtask-1 and subtask-2 dataset resepectively, but not on the NP sentence pairs. Eventhough the NP sentences are more in the dataset when compared with the other class(es)

of sentences in both the subtasks, the performance of the system does not depends on the NP sentence pairs. The lack in detecting the NP sentence pairs may be due to the encoder-decoder model characteristic, since the model learn word representations in continuous space, it tends to map the source sentence to target label that occur frequently in the context [14].

9 Conclusions

We used the encoder-decoder deep neural network model to detect the Tamil paraphrases from the DPIL@ FIRE2016 corpus. We evaluated our systems for 5 folds on the both the subtasks. We developed two variations with respect to attention techniques on the deep neural network – system using scaled-luong (SL), normed bahdanau (NB) as attention mechanisms. We compared our different models – GRU, gNMT variants – with the existing deep tamil paraphrase detection systems. For the subtask-1, gNMT-layered encoder-decoder model obtained 0.5% improvement in f1-score than the state-of-the-art systems. For the subtask-2, GRU-SL encoder-decoder model obtained overall f1-score of 61.81%. To our knowledge, ours is the first end-to-end deep learning model which validated the training instances of both the subtask-1 and subtask-2 dataset of DPIL@ FIRE2016. Eventhough the state-of-the-art systems for Tamil paraphrase detection have used different neural networks using CNN, MLP and Bi-LSTM, ours is the complete end-to-end encoder-decoder model with less or no preprocessing and lexico-syntactic features. The performance of this system can be improved further with more amount of training instances. Since the deep neural network system require more data to be trained, the DPIL training instances for Tamil paraphrase – 2500 and 3500 – was not enough for the deep neural network model to capture and learn the syntactic feature of the language from the given instances. Another way to improve the performance of the systems is to apply the recent deep neural network architectures that is suitable for the paraphrase detection.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Bhargava, R., Sharma, G., Sharma, Y.: Deep paraphrase detection in indian languages. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017. pp. 1152–1159. ACM (2017)
3. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
4. Daza, A., Frank, A.: A sequence-to-sequence model for semantic role labeling. arXiv preprint arXiv:1807.03006 (2018)
5. Dong, L., Lapata, M.: Language to logical form with neural attention. arXiv preprint arXiv:1601.01280 (2016)

6. Haghighi, S., Jasemi, M., Hessabi, S., Zolanvari, A.: Pycm: Multiclass confusion matrix library in python. Journal of Open Source Software **3**(25), 729 (2018). https://doi.org/10.21105/joss.00729, https://doi.org/10.21105/joss.00729
7. Kong, L., Chen, K., Tian, L., Hao, Z., Han, Z., Qi, H.: Hit2016@ dpil-fire2016: Detecting paraphrases in indian languages based on gradient tree boosting. In: FIRE (Working Notes). pp. 260–265 (2016)
8. Konstas, I., Iyer, S., Yatskar, M., Choi, Y., Zettlemoyer, L.: Neural amr: Sequence-to-sequence models for parsing and generation. arXiv preprint arXiv:1704.08381 (2017)
9. Kumar, M.A., Singh, S., Kavirajan, B., Soman, K.: Dpil@ fire 2016: Overview of shared task on detecting paraphrases in indian languages (dpil). vol **1737**, 233–238 (2016)
10. Kumar, M.A., Singh, S., Kavirajan, B., Soman, K.: Shared task on detecting paraphrases in indian languages (dpil): An overview. In: Forum for Information Retrieval Evaluation. pp. 128–140. Springer (2016)
11. Luong, M.T., Brevdo, E., Zhao, R.: Neural machine translation (seq2seq) tutorial. https://github. com/tensorflow/nmt (2017)
12. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
13. Mahalakshmi, S., Anand Kumar, M., Soman, K.: Paraphrase detection for tamil language using deep learning algorithm. Int. J. Appl. Eng. Res **10**(17), 13929–13934 (2015)
14. Nguyen, T.Q., Chiang, D.: Improving lexical choice in neural machine translation. arXiv preprint arXiv:1710.01329 (2017)
15. Saikh, T., Naskar, S.K., Bandyopadhyay, S.: Ju_nlp@ dpil-fire2016: Paraphrase detection in indian languages-a machine learning approach. In: FIRE (Working Notes). pp. 275–278 (2016)
16. Saini, A., Verma, A.: Anuj@ dpil-fire2016: a novel paraphrase detection method in hindi language using machine learning. In: Forum for Information Retrieval Evaluation. pp. 141–152. Springer (2016)
17. Sarkar, K.: Ks_ju@ dpil-fire2016: detecting paraphrases in indian languages using multinomial logistic regression model. arXiv preprint arXiv:1612.08171 (2016)
18. Senthil Kumar B, Thenmozhi D, C.A., S, K.: Tamil paraphrase detection using long-short term memory networks. In: Proceedings of Tamil Internet Conference – TIC2019, Chennai, India, ISSN 2313-4887. pp. 4–10
19. Sutskever, I., Vinyals, O., Le, Q.: Sequence to sequence learning with neural networks. Advances in NIPS (2014)
20. Thenmozhi, D., Kumar, S., Aravindan, C.: Ssn_nlp@ iecsil-fire-2018: Deep learning approach to named entity recognition and relation extraction for conversational systems in indian languages. In: FIRE (Working Notes). pp. 187–201 (2018)
21. Vinyals, O., Kaiser, Ł., Koo, T., Petrov, S., Sutskever, I., Hinton, G.: Grammar as a foreign language. In: Advances in neural information processing systems. pp. 2773–2781 (2015)