RECURRENT NEURAL NETWORK BASED PARTS OF SPEECH TAGGER FOR KANNADA WORDS

Abstract - Each word, given its context and sense in a given sentence, can be assigned a POS tag. In natural language, a word may belong to more than one lexical category, so morphology of the word and its relationship with its neighboring words play an important role while assigning its proper tag. As a part of this study, Recurrent Neural Networks based Parts of Speech Taggers namely Bi-Directional LSTM, LSTM, RNN, GRU for Kannada words is proposed. The Enabling Minority Language Engineering Corpus containing 14,965 sentences with 23,044 manually tagged words is used to train the model. The accuracy achieved after applying the four proposed models namely, Bi-Directional LSTM, LSTM, RNN, GRU are 99. 23%, 98. 92%, 98. 19%, 99. 24% respectively.

Keywords - Parts of Speech Tagging; Deep Learning; Recurrent Neural Networks; Natural Language Processing.

INTRODUCTION

Parts of Speech (PoS) tagger or annotator, a tool which assigns the appropriate syntactic categories to the words in a given text. PoS tagger is applied for Natural Language Processing (NLP). The two important cutting-edge variants of the Recurrent Neural Network (RNN) which have enabled the training of real datasets are GRU and LSTM models. Although RNNs, capable of solving a variety of sequence problems, they still have problems such as exploding and vanishing gradients. Recurrent Neural Network architectures — GRU, BLSTM and LSTM solve the problems that RNN exhibits.
Kannada is a well-known Dravidian language spoken primarily in South India. Kannada is a classical and administrative language in Karnataka. Kannada is an inflectional, derivational, morphologically well-off and relatively free word order natural language. In most cases, the main verb is in terminating position and the remaining words of all other lexical categories and subcategories can occur in any orientation.

LITERATURE REVIEW

The authors have proposed a model for Tamil language words, using a tagged corpus of around four lakh words and their tagset is made of 35 PoS tags. They have used a corpus-based approach, and have obtained an overall F-measure of 96% [1]. The author has designed a PoS tagging model for Hindi by applying a neural network approach. They have used a dataset consisting of 2,33,941 words and a tagset of 25 tags. The multi-neuron based approach gave an accuracy of 92.19% [2]. The authors used a rule based model to tag Hindi language words, where the corpus was made of more than twenty five thousand words along with 30 different standard PoS tags and achieved an accuracy of 87.55% [3]. The authors have proposed a Maximum Entropy model to the PoS tag for Kannada. Their dataset consists of 51,267 words from the Enabling Minority Language Engineering (EMILLE) corpus. A tagset of 25 PoS tags are used. Accuracy of 81. 6% was obtained for this model. [4]

The authors have proposed a model for Marathi language which is built on a statistical approach. The dataset consists of 7000 sentences with 1,95,647 words and a tagset of twenty relations and 15 node level tags. Unigram, Bigram, Trigram and Hidden Markov Model (HMM) are the methods used, which give an accuracy of 77.38%, 90.30%, 91.46% and 93.82% respectively. [5]

The authors have proposed a RNN model for social media texts in Hindi, English, Bengali and Telugu data. Their dataset has a size of around 5233 sentences. They have used RNN models such as LSTM, Deep LSTM, and GRU and obtained F1-score values between the range 68%-78%. [6]. The authors have proposed a PoS tagging approach for Kannada Language using Morpheme-ruled orders. They have used the Dravidian tagset and the dataset has been derived from Enabling Minority Language Engineering corpus. The precision decreases, recall increases and F-measure shows small variation for the four groups of dataset the model is tested on. [7]

The authors have proposed a Kannada Parts-of-Speech tagger on a dataset which has been made publicly available by the IIIT Hyderabad. The dataset has a size of about 13.1 thousand sentences from three different domains and 168 thousand tokens. They have used two different approaches, the first one being Machine Learning approaches where CRF++ and SVM have been considered and in their second approach they have used Bi-Directional LSTM Model, LSTM Model and Plain RNNs. They obtained F1-Score, Recall and Precision in the range of 0. 91-0. 92 for Machine Learning approaches and a score in the range of 0.84-0.92. [8]

The authors have used a tagset that contains seventeen tags and designed a PoS tagger for nine hundred and odd Malayalam tweets and obtained an F1-score of 0.9254 [9]. The authors have designed a PoS Tagger for Konkani Language using a Hidden Markov Model and Viterbi algorithm approach on a pre-tagged Konkani corpus.[10]. The authors have proposed a PoS tagger for Gujarati language using approaches such as VITERBI and SVM. The corpus contains 1700 words. The Viterbi model gave a better performance than SVM with a precision of 85% [11]. The authors have proposed a PoS tagger for Maithili language using LSTM and GRU with and without CRF layer, and have achieved 91.53% accuracy[12].

The literature review on PoS tagger clearly shows a lack of research towards a proper framework for Kannada language using Supervised learning methods. Taking this as a motivation, Neural Network models like LSTM, BiLSTM, RNN and GRU have been proposed for designing PoS tagger for Kannada Language.

PROPOSED WORK

The paper uses the Keras [13] framework for model architecture. Adam [14] is utilized for word embedding experiments with a batch size of 128. For all the recurrent network models, the loss function used is categorical cross-entropy [15] as it is a classification problem involving multiple classes. Metric chosen to determine the efficiency of the models is accuracy. The EMILLE corpus contains 14,965 sentences with 23,044 words that are manually tagged with their corresponding PoS tags. The longest sentence in the dataset has a length of 118 words. The PoS tagset for Kannada language is given in Fig. 1 and example is provided in Fig. 2.

Tag	Description	Example
NN	Noun	ಭಾಷೆ (bhaashe)
NNP	Proper Noun	ಕರ್ನಾಟಕ (karnataka)
PRP	Pronoun	ಅವನು (Avanu)
DEM	Demonstrative	ಈ (E)
VM	Verb Finite	ನಡೆದನು (naDedanu)
VAUX	Auxiliary Verb	ತಿನ್ನುತ್ತಾ (thinnuththaa)
JJ	Adjective	ಎತ್ತರವಾದ (Eththaravaada)
RB	Adverb	ನಿದಾನವಾಗಿ(nidhaanavaagi)
PSP	Postposition	ಜೊತೆ(jothe)
CC	Conjuncts	ಮತು³ (maththu)
WQ	Question Words	ಎĭÀ (Elli)
QC	Cardinal	ಎರಡು (Eradu)
QF	Quantifiers	ಬಹಳ (bahaLa)
QO	Ordinal	ಒಂದನೆಯ (ONdaneya)
INTF	Intensifier	ತುಂಬಾ (thumbaa)
INJ	Interjection	ಅಯ್ಯೋ (Ayyo)
NEG	Negation	ಬರೆದಿಲ್ಲ (baredilla)
SYM	Symbol	. , ( )
RDP	Reduplication	ದೊಡ್ಡ ದೊಡ್ಡ(dodda dodda)
UT	Quotative	ಎಂದು (Endu)
NUM	Numbers	77
ECH	Echo words	ಅಪ್ಪಿತಪ್ಪಿ (appitappi)
UNK	Unknown	ಸೆಲ್ಫಿ(selfie)
FOREIN	Foreign	Hello

Fig.1 – Tagset for Kannada PoS Tagger

```
['ಇವನು', 'ಭಾರತ', 'ಉಪಖಂಡ', 'ದ', 'ಬಹುಭಾಗವನ್ನು', 'ಒಗ್ಗೂಡಿಸುವದರಲ್ಲಿ', 'ಯಶಸ್ವಿಯಾದನು']
['PRP\r', 'NN\r', 'NN\r', 'NN\r', 'NN\r', 'NN\r', 'VM\r']
```

Fig.2 – PoS Tagging example

The dimensions of the input sentence for training data is 10812 x 100 while the output sentence has dimensions 10812 x 100 x 29. The dimensions of input sentence for validation data is 1908 x 100 whereas the dimension for output sentence is 1908 x 100 x 29. The input sentence of testing data has dimensions 2245 x 100 and the output sentence is 2245 x 100 x 29.

All the four models viz. , i) LSTM, ii) BiLSTM, iii) RNN and iv) GRU, basically have the same architecture, each model is of sequential type where a linear stack of layers are grouped into a keras model. The first layer of each model outputs a dimension which is the same as the embedding size and the dimension of the input is the same as that of the vocabulary size. The second layer is a model specific layer where an RNN model will have a Simple RNN layer of 64 RNN cells, an LSTM model will have an LSTM layer of 64 LSTM cells, an Bi-Directional LSTM model will have a Bidirectional layer of 64 LSTM cells, and a GRU model will have a GRU layer of 64 cells. The next layer in each of the four models is a Time Distributed which has a Dense layer and uses a softmax activation. The general architecture for the proposed PoS tagger for Kannada is depicted in Fig.3.

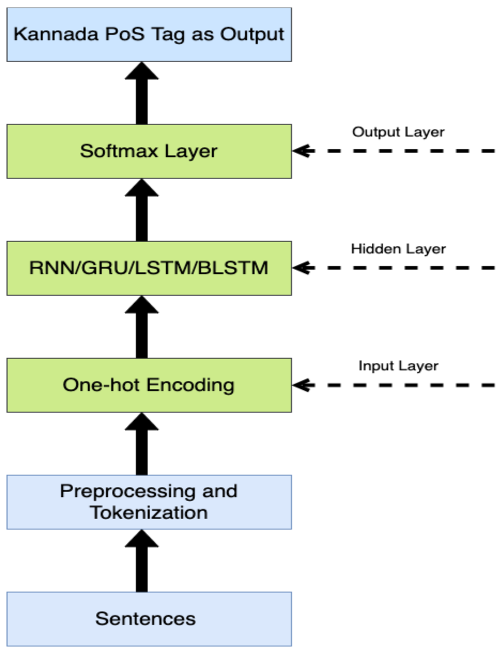

Fig.3 – Proposed Architecture for Kannada PoS Tagger

METHODOLOGY

Recurrent Neural Network [RNN]- The sequence of words is fed as input to the mode, and at every point in time an RNN unit is not only applied to the word but also to the output of the model from the previous time step. RNN handles the considered words in its consecutive form $x_1, x_2, x_x, \ldots, x_n$, where, the number of elements or the length of elements in a sequence is not uniform. These models use a method, where the output of a particular state is looped to the same state as input which is called the feedback mechanism. Aforementioned mechanism is seen as time passage and gives a memory of events that occurred up to that particular point. Previous state which is hidden and current state are used to determine hidden state. Hidden state calculation is done using the formula shown below,

$$H_t = f\,(x_t, h_t^{-1}) \quad (1)$$
$$H_t = f\,(Ax_t, Wh_{t-1}) \quad (2)$$

Where $\boldsymbol{f}$ is an activation function like the softmax activation function used in this proposed work.

Long-Short Term Memory [LSTM]- is used to trap dependencies which are lengthy. A decision is made internally if the information is required or not depending on the inputs to the memory cell, which are output from the previous state which is masked state and the current input. The general mathematics involved in measuring the masked state at a distinct time $\boldsymbol{t}$ is as seen in (2). There are 3 main gates, which are, input gate, output gate and forget gate. The estimation of the condition of the cell, along with the 3 gates, is used to enumerate the hidden state vector. The equations to calculate the hidden states are as shown below:

$$i_t = \sigma\,(A^{i}x_t + W^{i}\,h_{t-1}) \quad (3)$$
$$fg_t = \sigma\,(A^{fg}x_t + W^{fg}\,h_{t-1}) \quad (4)$$
$$o_t = \sigma\,(A^{o}x_t + W^{o}\,h_{t-1}) \quad (5)$$
$$\hat{c}_t = tanh\,(A^{c}x_t + W^{c}h_{t-1}) \quad (6)$$
$$c_t = \hat{c}_t O i_t + c_{t-1}\,O\,fg_t \quad (7)$$
$$h_t = o_t\,O tanh(c_t) \quad (8)$$

Here, g,i,o,f refer to the output gate, input gate, cell state, forget gate respectively and they have the dimension equivalent to hidden state vector[16].

Bi-directional Long Short Term Memory [BLSTM]- The information along forward and backward directions are captured with the help of a grid. BLSTM can seize the factors for higher times than its fellow variants. BLSTM has its applications in Natural Language Processing (NLP) where fields like modeling a language and sequence labeling come into picture.

Gated Recurrent Unit [GRU]- A simple gating function is provided by a GRU [17]. It has the following gates: reset gate(r) and update gate(u). Current state and previous state which is hidden combined with value which is a sigmoid function over the previous input is fed into the reset gate. The equations given below represent the calculation of the update and reset gates.

$$u_t = \sigma(A^{u}x_t + W^{u}h_{t-1}) \quad (9)$$

$$r_t = \sigma\,(A^{r} + W^{r}h_{t-1}) \quad (10)$$

where,

$x_t, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots x_n$: n words considered as vectors corresponding to a corpus.

$W\,h_t$: The input word vector conditioning weight matrix

σ() : the nonlinearity function

The dataset is loaded and then split into sentences and consecutively into words which are encoded with UTF-8 standard. Tokenizer is instantiated and fits onto the data to encode the input sentences. The input and output sentences are verified for length discrepancies. The sentences and labels are then padded using a maximum sequence length of 100. The word2vec model is loaded and embedding size and vocabulary size is fed into the model. The size of embedding is 300 and the vocabulary size is the length of the output of word tokenizer on the dataset. Then the respective weights are assigned to the data. The labels are then one-hot encoded using the keras to categorical function. Then 70% of the considered dataset is included for training the models and 15% of the dataset for testing and the remaining 15% for validation. Then the four considered models are built and run on training, testing and validation data for 10 epochs each. The respective accuracy and categorical cross entropy losses are calculated on training and testing data.

RESULT ANALYSIS AND DISCUSSION

A. Result analysis:

This paper uses accuracy, which describes how the model performs across all the classes since the given problem is a multiclass classification task. Accuracy of the suggested methodology is computed using (11).

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}} \quad (11)$$

Categorical cross entropy is used to compute loss. When softmax activation and the loss function is used for a classification problem with multiple classes, during training probabilistic false negatives are not directly penalized. Whilst training a neural network for classification purposes, the function that calculates the cost, known as cost function, is the main character required to adjust the influence of the networks to create a model that does not underfit or overfit. When the neural network is in its forward propagation phase, it is tested on the dataset during training and the generated output in the case of a multiclass problem like ours indicates in possible labels the confidence or probability. The probabilities are then matched with the true labels of the word in question, and the loss function considered for the model, in our case the categorical cross entropy metric, then calculates how much deviation occurs between target and output tags generated by the neural network for the same word. During the next phase, which is backpropagation, a partial derivative of the categorical cross entropy for each weight that can be trained on the network is calculated.

The standard weighted categorical cross-entropy loss can be computed using the formula (12).

$$J = -\frac{1}{M} \Sigma_{k=1}^{K} \Sigma_{m=1}^{M} w_k \times y_m^k log(h_\theta (x_m, k)) \quad (12)$$

where,

M : aggregate of classes for training

K : aggregate of classes

w_k : class k's weight

y^k_m : target value for class k for training word m

x_m : input for training word m

h_θ : model with neural network weight as θ

Precision and Recall are calculated using (13) and (14) respectively. F1-score is the weighted average of Precision and Recall which is calculated using (15).

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}} \quad (13)$$

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}} \quad (14)$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (15)$$

Results obtained by the proposed four models with regard to accuracy, loss are given in Table 1. Graphs plotted on the accuracy obtained by models namely i) RNN, ii) LSTM, iii) BiLSTM and iv) GRU are given in Fig. 4, 5, 6 and 7 respectively.

Table 1 – Accuracy, Loss , Precision, Recall and F1 Score obtained by the proposed model.

SI No	Model	Accuracy	Loss	Precision	Recall	F1 Score
1	Recurrent Neural Network	0.9819	0.0677	0.9859	0.9780	0.9819
2	Long-Short Term Memory	0.9892	0.0339	0.9931	0.9865	0.9897
3	Bi-Directional Long Short Term Memory	0.9923	0.0271	0.9956	0.9890	0.9923
4	Gated Recurrent Unit	0.9924	0.0252	0.9932	0.9915	0.9924

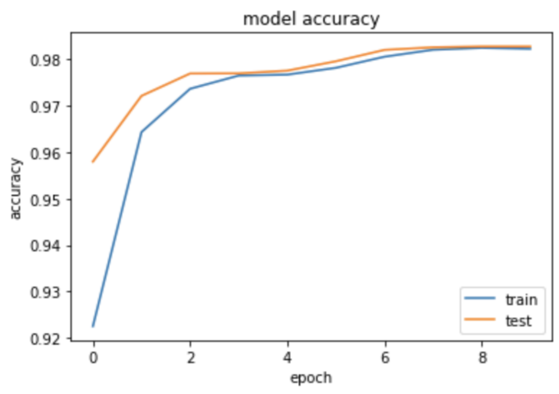

Fig 4 - Accuracy for RNN model

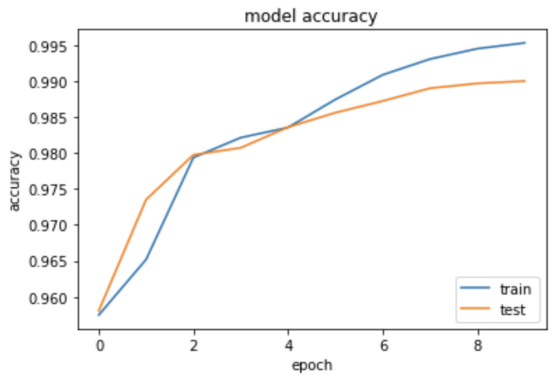

Fig.5 - Accuracy for LSTM model

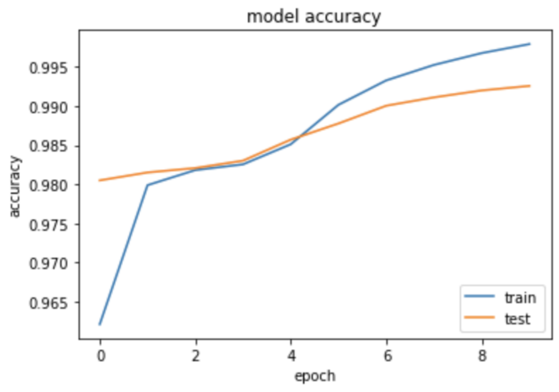

Fig.6 - Accuracy for BLSTM model

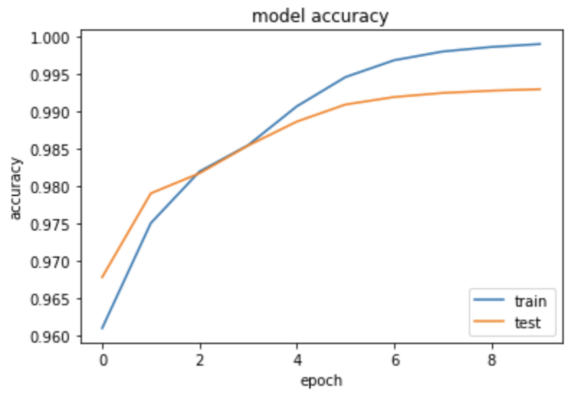

Fig.7 - Accuracy for GRU model

B. Discussion:

The RNN model performs well, having a healthy growth curve, having said that it also has the highest loss value compared to its other three variants. The LSTM model also provides some marginal improvement compared to the RNN model, but it still has a loss value which does imply that our predicted probabilities and the actual labels are not good enough a match. Using bidirectional LSTMs, it is seen that the time taken to train is twice as there is an increase in the count of parameters in the network, which shows a commutation between performance, time taken to train. GRU model clearly has a greater accuracy at 0. 9924 and a minimal loss of 0.0252 indicating that our model has a great probability percentage. This shows that GRU can be used for similar datasets with over 10,000 sentences and gives a better performance than the Vanilla RNN Neural networks and LSTM models. Models having good accuracy rates but high loss indicates a possibility that there are one or a few outliers that are classified extremely badly and these are causing the loss to explode, but at the same time the model is still learning on the rest of the dataset.

Table 2 – Comparing the results of proposed work with existing work.

References	Models Proposed	Accuracy	Loss	Precision	Recall	F1 Score
[6]	LSTM,Deep LSTM,GRU	NA	NA	NA	NA	0.78
[8]	CRF++,SVM,Bi-LSTM,LSTM,RNN	NA	NA	0.92	0.92	0.92
[12]	LSTM,GRU,LSTM+CRF,GRU+CRF	0.91	NA	NA	NA	NA
Proposed Work	RNN,LSTM,Bi-LSTM,GRU	0.99	0.02	0.99	0.99	0.99

CONCLUSION AND FUTURE WORK

A suggestion for building a PoS tagger for words in Kannada language using Deep Learning approaches is shared in this paper. Sequential model algorithms like BLSTM, LSTM, RNN and GRU are used in building the proposed models. The Enabling Minority Language Engineering(EMILLE) Corpus containing 14,965 sentences with 23,044 manually tagged words is used as the training and testing dataset in the proposed models. The accuracy obtained by the LSTM, BLSTM, RNN and GRU models is 98.9%, 99.2%, 98.19% and 99.24% respectively. It is observed clearly the GRU model outperforms other models. GRU model has an evident overfitting problem, which can be attempted to be resolved using dropouts. In future work, the proposed work can be implemented on larger datasets and also for other morphologically rich languages such as Tamil and Hindi.

ACKNOWLEDGMENT

This paper is a revised and expanded version of an article entitled, "RECURRENT NEURAL NETWORK BASED PARTS OF SPEECH TAGGER FOR KANNADA WORDS" presented in 'ICRTST-2021', held at ' ATME College of Engineering, Mysore,' India during 8th and 9th of July, 2021.

REFERENCES

[1] S. Pandian & Geetha, T. V. “Morpheme based Language Model for Tamil Part-of-Speech Tagging” Polibits.38. 19-25. 10. 17562/PB-38-2, 2008

[2] Parikh, Ankur. “Part-Of-Speech Tagging using Neural Network.”, 2010.

[3] Navneet Garg, Vishal Goyal, Suman Preet. Rule based Hindi Part of Speech Tagger.”, 2012

[4] Shambhavi. b. R, Ramakanth Kumar P and Revanth G. Article: “A Maximum Entropy Approach to Kannada Part Of Speech Tagging”.*International Journal of Computer Applications* 41(13):9-12, March 2012.

[5] Singh, Jyoti & Joshi, Nisheeth & Mathur, Iti. “Development of Marathi Part of Speech Tagger Using Statistical Approach”10. 1109/ICACCI. 2013. 6637411, 2013

[6] Patel, Raj & Pimpale, Prakash & Mukundan, Sasikumar. “Recurrent Neural Network based Part-of-Speech Tagger for Code-Mixed Social Media Text”, 2016

[7] R J, Prathibha & M C, Padma. “Morpheme Based Parts of Speech Tagger For Kannada Language”, 2016

[8] Todi, Ketan & Mishra, Pruthwik & Sharma, Dipti. “Building a Kannada POS Tagger Using MachineLearning and Neural Network Models”, 2018

[9] Kumar, S. , Kumar, M. Anand and Soman, K. P. "Deep Learning Based Part-of-Speech Tagging for Malayalam Twitter Data (Special Issue: Deep Learning Techniques for Natural Language Processing)" *Journal of Intelligent Systems*, vol. 28, no. 3, 2019, pp. 423-435.

[10] Diksha N. Prabhu Khorjuvenkar, Megha Ainapurkar, Sufola Chagas. “Parts of Speech Tagging For Konkani Language”, 2018

[11] Manisha Prajapati, Archit Yajnik.“POS Tagging of Gujarati Text using VITERBI and SVM”, 2019

[12] Ankur Priyadarshi and Sujan Kumar Saha. 2023. A Study on the Performance of Recurrent Neural Network based Models in Maithili Part of Speech Tagging. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 22, 2, Article 32 (February 2023), 16 pages.

[13] Gulli A, Pal S. “ Deep learning with Keras” Packt Publishing Ltd; 2017.

[14] Diederik P. Kingma, Jimmy Ba “Adam: A Method for Stochastic Optimization.” Submitted on 22 Dec 2014 (v1), last revised 30 Jan 2017 (this version, v9).

[15] Y. Ho and S. Wookey, "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling," in *IEEE Access*, vol. 8, pp. 4806-4813, doi: 10. 1109/ACCESS. 2019. 2962617, 2020.

[16] Sepp Hochreiter, Jürgen Schmidhuber; “Long Short-Term Memory.” *Neural Comput* ; 9 (8): 1735–1780, 1997.

[17] J Chung, C. Gilcehre, K. Cho and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling”, arXiv preprint arXiv:1412. 3555, 2014.