NAMED ENTITY RECOGNITION USING INDEPENDENTLY RECURRENT NEURAL NETWORKS

A project report

Submitted as a Part of Major Project-2

Master of Technology in Information Systems

BY

MANI WADHWA

2K17/ISY/09

Under the Guidance of:

Ms. Ritu Agarwal

(Assistant Professor- Information Technology)

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

May-2019

DECLARATION

I, Mani Wadhwa, Roll No. 2K17/ISY/09 of M.Tech. Information Systems, hereby declare that the project Dissertation titled “Named Entity Recognition using Independently Recurrent Neural Networks” which is submitted by me to the Department of Information technology, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Mani Wadhwa

2K17/ISY/09

CERTIFICATE

I hereby certify that the Major project report-2 titled "Named Entity Recognition using Independently Recurrent Neural Networks" which is submitted by Mani Wadhwa, 2K17/ISY/09 in partial fulfillment of the requirement for the award of degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or Diploma to this University or elsewhere.

Ms. Ritu Agarwal

Assistant Professor

Project Guide

Department of Information Technology

Delhi Technological University

ACKNOWLEDGEMENT

I am very thankful to **Ms. Ritu Agarwal** (Assistant Professor, Department of Information Technology) and all the faculty members of the Department of Information Technology of DTU. They all provided us with immense support and guidance for the project.

I would also like to express my gratitude to the university for providing us with the laboratories, infrastructure, testing facilities and environment which allowed us to work without any obstructions.

I would also like to appreciate the support provided to us by our lab assistants, seniors and our peer group who aided us with all the knowledge they had regarding various topics.

Mani Wadhwa

Roll No. 2K17/ISY/09

ABSTRACT

The task of Named Entity Recognition is one the Natural Language Processing applications. The popular models to address the problem of understanding sequential data are LSTM and GRU. These models not only improve upon the long term dependencies but provide a good understanding of the context to predict tokens and their tags. However, gradient vanishing over deeper and longer layers has been an issue with these state-of-the-art models as well. Hence, IndRNN, which has recently been proposed as an alternative for sequential data processing has been introduced in our approach. Its application on NER has still not been discovered. This thesis work deals with the effectiveness of independently recurrent neural network on the task of Named Entity Recognition. IndRNN provides independence within the layers which helps improve the understanding of the functioning of the neural network. Also, each RNN is connected to each RNN from another layer which provides the correlation between text that we need.

CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Named Entity Recognition (NER) refers to the extraction and classification of named entities into various predefined classes such as name of the individual, organization, places, dates etc. NER is particularly useful for when we wish to identify text related to certain entity. Some of its use cases include classifying the news content, searching and tagging the articles for keywords, assigning certain tasks to individuals according to place and time e.g. delivery of couriers. Short messages which are posted on social media poses a challenge in recognizing named entities as their language is informal and the data is most of the times noisy. Tweets, present on social media site, Twitter, are being used in this work. Tweets provide the information which is more up-to-date and freely expressible as compared to news articles. However, tweet classification for NER is a challenging task because of some reasons. One of them is the variety of named entities (places, movies, individuals, organizations etc.) which are updated often and hence lack the capability of being a source for good training data. Another shortcoming of tweets is the word limit of 140 characters on Twitter. This poses a challenge in determining type of an entity in lack of sufficient context. Hence various approaches have been proposed in this field. These can be divided into three categories as shown in Fig.1. First approach is classical one, for which NLTK package is available in python and can be used. It is a rule based approach. Next are machine learning approaches. These are also of two types: Classification based and Conditional Random Field (CRF) based. Classification approach treats the problem of NER as a multi-class classification problem. Different classification algorithms can be applied considering entities as labels. This method does not take into account the context of a sentence where entity occurs. CRF is a machine learning model. It is a statistics based model which can be used to label sequential data into various categories. CRF models can get the information in the form of features from previous and current labels, but not the upcoming or forward ones. CRF models are not very efficient for NER because of this shortcoming as well as feature engineering that has to be done prior. Finally, let’s introduce the deep learning approaches (state-of-the-art) for NER. Since NER is an application of Natural Language Processing (NLP), we are dealing with

sequential data and require context knowledge. For such an application, Recurrent Neural Networks (RNNs) are best suited. RNNs are further improvised to form Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). Bidirectional LSTM has been used extensively for NER. This is a two way LSTM where forward and backward context is taken into account. Another alternative to LSTM is Gated Recurrent Unit (GRU). This is relatively new but is more efficient than LSTM for certain tasks.

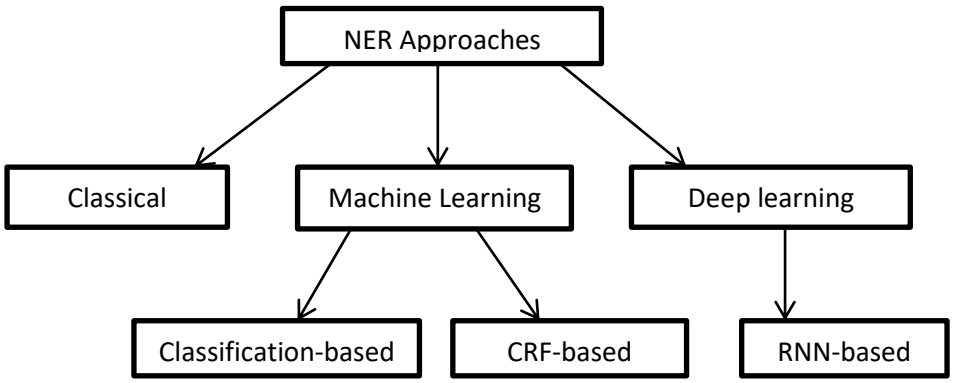


Fig.1. NER approaches bifurcation

Applications of NER

Nowadays, a variety of articles dealing with diverse range of topics are being published every day. So, keeping track of each article and the subject that it deals with becomes quite imperative. NER can actually be used to scan the complete article and find the major people, places and organizations. When we know the important and frequent tags within an article, it helps to categorize the article and eventually help us with better content organization and discovery. After the tags have been derived using NER, searching also becomes easier in a huge amount of content, dissolved down to certain important tags. One of the most effective usages of NER is automating the recommendation process. Recommendation systems play an important role in today's world, whether it is in entertainment, education, news or products. NER makes sure that the viewer is being recommended the product or show with similar tags. It is also highly useful in customer support systems. For example, name of the product and place where the complaint is lodged may help redirect it to the right service center. Publications and journals hold thousands of research papers. There can be many similar papers with slight modification and hence to

determine such papers, NER can be used. Clustering the papers on the basis of entities and finding the relevant information is a task that can be solved using NER.

CHAPTER 2. RELATED WORK

The method of segmentation of tweets has been used for NER. In this work, tweets are segmented into meaningful segments, where the preservation of semantics and context are being done. NER can be done after this step using two ways, Random walk and Part Of Speech (POS) tagger. Random walk method relies on co-occurence of named entities multiple times. Keeping this observation, a segment graph is built. An edge signifies that two entities co-occur in tweets. Jaccard coefficient has been used to calculate the weight of the edges. Finally the random-walk model is applied to this formed segment graph. This method however is not very effective as the length of tweets are small and co-occurrence rarely happens. Second method is POS tagger which uses noun phrases as entities instead of individual words. So this noun phrase has a greater probability to appear in different tweets. So, POS tags of a noun phrase determines their likelihood.[1] Tweets are available in various languages. Hence, different techniques have been used for different languages. One of the works in vietnamese suggested applying a preprocessing operation of normalisation. This process handles spelling mistakes and noisy data like emotion symbols. Then word segmentation takes place over clean data. POS tagging is then performed over it[2] Another method of segmentation involves dividing the tweets into fragments with the help of 2-gram algorithm. These fragments are then classified into universal context ones and confined context ones. Then the POS tagger is used to identify named entities. Each tag is used to identify words being present in tweets. All such words are used to identify the named entities.[3] Most of the methods employed for NER are supervised learning methods. In such methods, large amount of data is required for training an efficient classifier. One of the popular techniques for NER is CRF, which again is a supervised learning method. These methods are very well suited for well formed sentences, but twitter is a social networking site and sentences are somewhat more natural. To overcome such limitations, a semi-supervised method has been proposed, which is an extension of CRF. In this work, CRF is combined with a semi-supervised method which relies on co-occurrence coefficient of words which are surrounded by proper noun. Since lack of context information has been a challenge in the case of tweets, cosine

similarity has been used to cluster tweets which belong to some similar group.[4] Bi-directional LSTM has been used in this work. It takes into account the forward and backward information and hence proves to an efficient approach for NER. Experiments have been done in this approach with preprocessing part. Word embeddings have been used alone. Then along with neighbour word embeddings and finally with POS Tag.[5] The classifier for NER has been improved upon in this work. NER is performed on a set of labeled data. It inculcates a combination of active learning and self learning methods. Then finally the CRF model is applied in this approach as well.[6] The dataset file is trained with CRF model of machine learning. CRF can be implemented using CRFsuite in python. Some natural language features such as part of speech, words before and following it, capitalisation were also considered while training in this work.[7] NER approaches have a drawback of high dependency over hand crafted features as well as domain knowledge. So, to overcome such shortcoming, methods have been proposed which do not rely on hand crafted features. The technique can be used for both image and text.[8] Convolution Neural Network (CNN) has been used along with LSTM to improve the efficiency in the task of NER. First of all, every word is vectorised. The process of vectorisation is done by concatenating word embeddings along with the character-level features that are being extracted by CNN. Then finally the vector is fed to LSTM which is combined with CRF that does the task of assignment.[9]

CHAPTER 3.RECURRENT NEURAL NETWORKS AND ITS VARIANTS

Recurrent Neural Networks are one of the most powerful neural networks to deal with the sequential data. They have got the internal memory which helps them remember facts from past. This is utterly important for NLP tasks. This is because when we try to learn or generate words or a sentence, we take our previous knowledge into account rather than starting afresh everytime. This is the case with RNNs, they take previous context into account. RNNs were actually introduced earlier as other machine learning algorithms. But their effectiveness have now reached its peak with a lot of computation power and resources available. Because of its internal memory, it is able to precisely predict what is about to follow based on its previously read context. That is why they are the preferable kind of algorithm for sequential data like speech,

text, audio, video etc. because they are able to form a level of understanding which other algorithms fail to form.

CHAPTER 3.1. RECURRENT NEURAL NETWORK

Before we explain the working of RNNs, it is important to know the feed-forward networks. In feed-forward networks, the movement of information only takes place in one direction, starting from input layer, moving through hidden layers and towards the output layer. The information once passed is not being touched by same node again. Since feed-forward networks have no memory of their own, they are not good at predicting what is going to follow. Only the current input is considered. Except training, feed forward networks remember nothing. In RNN, information goes through a loop. Hence, while predicting, RNNs makes use of information read so far as well as current information. Fig.2. demonstrates the difference between feed-forward and recurrent neural networks.

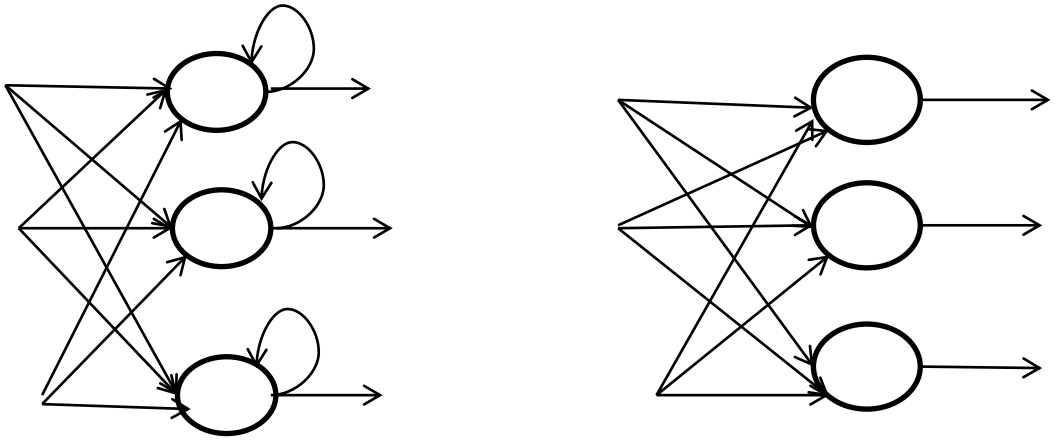


Fig.2. Demonstrating difference between RNN and normal feed-forward network[10]

Usually RNNs have short-term memory. This limitation is solved by LSTM and GRU which will be discussed later. So, for example if we have a word ‘hello’, a normal feed-forward network, on reaching ‘l’ will forget about ‘h’ and hence it becomes impossible for it to predict the next letter or word. On the other hand, an RNN is able to remember that much and hence is useful for predicting in the case of sequential data. It does this with the help of its internal memory. It first produces the output, stores the output and feeds it back into the network. It has the ability to add

the immediate seen text into present text. Therefore, precisely said RNN has two inputs, present and past seen recently. This is an important property as a context always contains the information of what is going to follow, this is the main reason why RNN can do things that other algorithms cannot. A feed forward network assigns weights to the inputs in order to produce the output. But RNN assigns weights to previous and current input. Also feed forward network network can map each input to single output, whereas RNN has the capability to map inputs to outputs in many ways i.e. one to many, many to many, many to one.

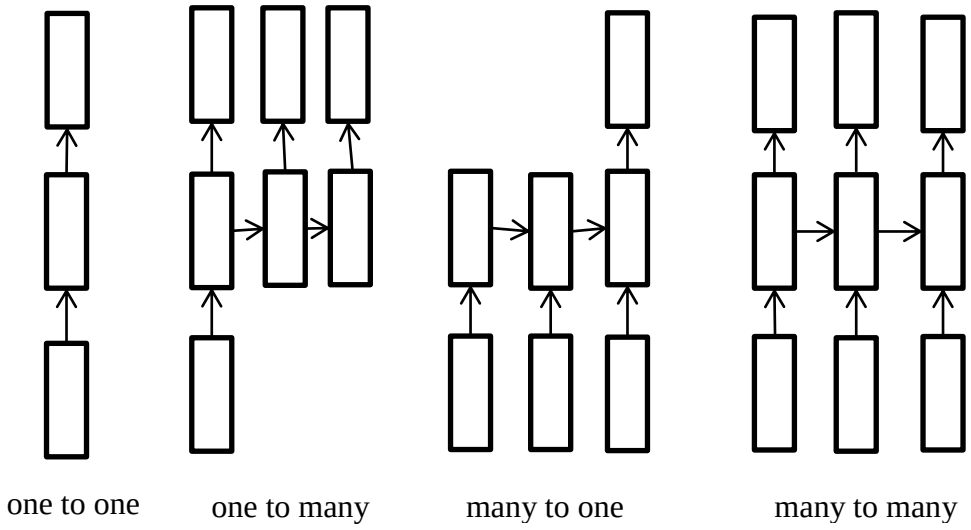


Fig.3. Many forms of RNN

Following are the sequence of equations for a RNN.

Given a sequence of input x= $(x_1, x_2, x_3 \ldots\ldots\ldots x_t)$,now the updation of the hidden state h_t is given by:

$$h_t = 0, \qquad t = 0 \qquad (3.1.1)$$

$$= \Phi(h_{t-1,} x_t), \quad otherwise$$

where Φ is a nonlinear function which can be a sigmoid function. Also, there is an output produced which may or may not be used for further computation, which is y = (y_1, y, y_3 …… y_t), which could be of variable length as well.

The update that happens in RNN hidden sate in eq.3.1.1. is practically implemented as:

$h_t = g(Wx_t + Uh_{t-1})$, (3.1.2)

where g is a bounded function which can be tangent function of sigmoid function

Issues with RNN:

Exploding Gradient and Vanishing Gradient

The problem of exploding gradient occurs when an algorithm assigns high value to weights during training. This causes the RNN to become unstable for learning from training data. At an extreme case, values can become so large that it could either overflow or give NaN values. This occurs essentially because of the reason that there is an exponential growth if we multiply gradients by values greater than 1. Vanishing gradient has exactly the opposite case. The algorithm assigns very low values to weights and hence the gradient becomes very small that it tends to vanish. At an extreme case, the model might stop training as there is no further updating of weights visible because of the vanishing gradient. This happens when the gradient becomes less than 1 and keeps on decreasing further.

To solve these issues LSTM and GRU were being introduced. These two have been used practically in many applications.

CHAPTER 3.2.LSTM (LONG SHORT TERM MEMORY)

One of the major advantages of RNN is that they can relate previous information to the present one, such as previous context in sentence might be able to recognize what word is going to follow. RNNs are supposed to do exactly this. But there is a catch in this. When there is a condition of predicting the word on the basis of just read context, then RNNs can easily be

applied. On the contrary, when there is a huge gap between the context and the word to be predicted, the RNNs loses its effectiveness. For example, if we have a sentence such as Jack went to the office. There were clouds in the sky and so on… and later we have to predict 'he/she'. Then, RNN is unable to predict in the case of such long gap between context and prediction. Long-term dependency is the problem with RNN. To overcome this shortcoming, LSTMs were introduced[11] To remember things for a long period of time is a functionality of LSTM, it is not something it has to work hard to achieve. All RNNs have a series of repeating neural networks where each neural network is a simple structure, like a tanh layer.

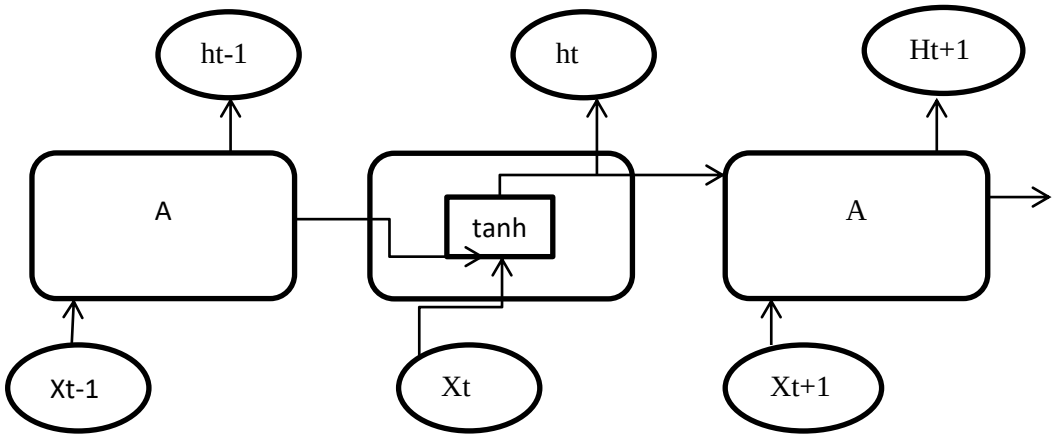


Fig.4. The repeating module in a standard RNN contains a single layer.

LSTMs also have a series of neural networks, but it differs in the structure of the single neural network that it has a complicated structure. It has four interacting layers in each repeating module.

LSTMs have the ability to add or remove previous information to pass to next layers, by a structure called 'gates'. Gates are a medium which optionally let the previous information flow. It usually has two components, sigmoid neural network layer and a pointwise multiplication operator. The role of sigmoid functions is to decide which information to pass by generating the value between 0 and 1. A '0' value means let nothing pass and a '1' value lets everything pass of a certain component. The core of LSTM comprises of cell state and it various gates. The cell state is the component that acts as a highway and carries the information down to the sequences of neural networks. This is actually the 'memory component' of the network. The state of t

he cell determines the information being carried. As it moves, information gets added or is being removed to the state of the cell with the help of gates. These gates decide the part of information that is to be kept and removed.

Formally, the formulas to update an LSTM unit at time ‘t’ are:

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \quad (3.2.1)$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \quad (3.2.2)$$

$$\bar{c}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c) \quad (3.2.3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \bar{c}_t \quad (3.2.4)$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \quad (3.2.5)$$

$$h_t = o_t \odot \tanh(c_t) \quad (3.2.6)$$

where,

x_t: input vector (at time t)

σ: sigmoid-function

$\odot$: element-wise product

h_t: stateful hidden state vector (at time t)

b_*: bias vectors

U_*: weight matrices of different gates of input x_t

W_*: weight matrices for hidden state h_t

Chapter 3.3. Bi-LSTM

Since we have observed, that LSTM performs well when the dependencies are in the forward direction. However, for the sequence labeling tasks, which require knowledge of both past and future context, LSTMs might not give good results. Hence, bidirectional LSTM network for NER has been introduced[12]. In vanilla LSTM, the information is only being retrieved from past but in bi-LSTM the information not only comes from backward direction, but also from forward direction. Then these two states are combined to form the final output[13]. Some modifications can also be done to the model by setting the state to 0 while we begin a new sentence and at the end of it.

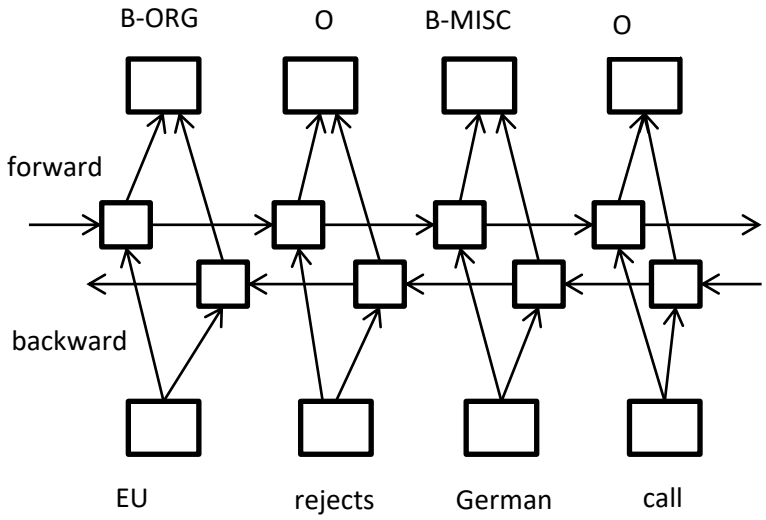


Figure 5: Bi-directional LSTM network[14]

Chapter 3.4. GRU (Gated Recurrent Unit)[15]

The architecture of GRU is defined by given equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \qquad (3.4.1)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \qquad (3.4.2)$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h(h_{t-1} r_t) + b_h), \qquad (3.4.3)$$

$$h_t = \sigma(z_t h_{t-1} + (1 - z_t)\tilde{h}_t) \qquad (3.4.4)$$

z_t and r_t are the vectors which corresponds to update gate and reset gate. For the current time t, h_t represents the current state vector. Both of these gates have sigmoid activation function, σ(.). z_t as well as $r_t$take the value between 0 and 1. $\tilde{h}_t$ is the candidate state, and processed with tanh function. x_t is the input vector. U_z, U_r, U_h are the recurrent weights and W_z, W_r, W_h are the feed-forward weight of connections. The biases which are b_z, b_r, b_h have also been added.

The similarities between GRU and LSTM are pretty noticeable in the equations itself. The addition when the update occurs from t to t+1 is a feature which is common in both LSTM and GRU. This was not found in RNN, the contents of a particular unit is updated with the present computations. On the other hand, the contents are added on top of current content in the case of LSTM and GRU.

Because of this additive nature of LSTM and GRU, it is easy for them to remember some important feature based on the value being passed by update gate and forget gate. Such feature is not overwritten by present content. Also, the errors can be back-propagated without the issue of vanishing gradient.

However, they have differences as well. The exposure difference is present in LSTM and GRU, LSTM has a controlled exposure of the content passing through. The usage of memory by other units is handled by output gate. On the other hand, GRU exposes all of its contents without keeping any control over it. Dissimilarity lies in the positioning of the input gate or reset gate. The new memory content is produced by LSTM, it does not have a separate hold on the previous content generated. LSTM adds the memory content independent of the forget gate. But in the case of GRU previous activations are being considered. It does have a control over the activations as in the case of LSTM.

So, it is not very clear as to which one is better that the other. However, they do show different results based on the task in hand. For our research, GRU has proven to given better results.

To solve the problem of vanishing gradient which prevails in a vanilla RNN, GRU has two gates, update gate and reset gate. The information that has to be passed is to be decided by these two gates. The advantage of these gates is that their training can be done to remember information for a long time.

The update gate equation is given by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \qquad (3.4.1)$$

When x_t which is input is given in the network, it is multiplied by the weight vector. Same is the case with hidden layer vectors which holds all the information from t-1 units. Both the multiplication results are being added together and activation function of sigmoid is being applied, so that the output comes between 0 and 1. The functionality of the update gate is to determine the amount of past information that need to be passed to the future. One of the advantages of this is the ability of update gate to pass all the information through, without removing anything and hence eliminate the problem of vanishing gradient.

As for the reset gate, it does the opposite of update gate, it decides upon which part of the information to forget. The formula is given by:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \qquad (3.4.2)$$

The formula is quite similar to that of the update gate. The difference lies in the weight factor and the use of the reset gate. So, finally the equation which is used to calculate the memory content is given by:

$$\tilde{h}_t = \tanh(W_h x_t + U_h(h_{t-1} r_t) + b_h) \qquad (3.4.3)$$

Now after the multiplication part of weights and inputs, we will come to the element wise product in the second term. That will determine the content to be removed. Then the sum up operation takes place and finally tanh is applied over it.

At the last step, the information which has to be passed down, is being given by f

ollowing equation:

$$h_t = \sigma(z_t h_{t-1} + (1 - z_t)\tilde{h}_t) \qquad (3.4.4)$$

These terms determine what has to be kept and what to be discarded.

CHAPTER 3.5. INDEPENDENTLY RECURRENT NEURAL NETWORK

RNNs and its variations have been used extensively in sequential data. RNNs have a famous vanishing and exploding gradient problems. LSTM and GRU have tried their bit to get away from the problem, but the usage of activation functions like sigmoid and tanh result in decay of gradient with increased number of layers. Hence, a deep neural network is difficult to train which has number of layers. Moreover, the entangled RNNs make the task of understanding the network even more difficult. Hence, IndRNN is being proposed[16] where neurons which are present on the same layer are independent of each other, but across layers they are connected. IndRNN has been shown to prevent the gradient problem, allows long-term dependencies to be learnt. Multiple such IndRNN can also be combined together to to form more deeper layers as compared to vanilla RNN.

In the proposed work, a new variant of RNN, which is IndRNN is given, the inputs are being handled using Hadamard product. This has its own advantages over traditional RNNs:-

To address the problem of vanishing and exploding gradients, the gradient backpropagation through time can be regulated.

To process long sequences and preserve long term dependency, long term memory can be kept with IndRNN. The proposed IndRNN can process over 5000 steps while LSTM process 1000 steps.

A non-saturated function like relu can also be used with IndRNN, which was not possible in vanilla RNN. IndRNN can be trained robustly using this activation function.

Depth of the layers can be increased with stacking many IndRNN layers over one another.

Interpretation of IndRNN neurons' behavior is easier as neurons are independent of each other in each layer.

The equation to define IndRNN is:

$$h_t = \sigma(Wx_t + u \odot h_{t-1} + b), \qquad (3.5.1)$$

u is the weight vector and $\odot$ is the Hadamard product.. Since we already know that multiple Ind_RNN layers can be stacked on the top of each other, while being independent on its corresponding layer. So, for the nth neuron, the hidden state $h_{n,t}$ can be derived using:

$$h_{n,t} = \sigma\left(W_n x_t + u_n h_{n,t-1} + b_n\right) \qquad (3.5.2)$$

Where wn and un are the nth row of the input weight and recurrent weight. Information to each neuron is supplied from iput and its own hidden state at the previous time stamp. Spatial-temporal independence is present in the case of IndRNN. RNN is actually a multilayer perceptrons. So, IndRNN provides an altogether a new perspective on RNN. Although neurons are independent of each other, they still correlates with each other by communication among layers. So in the case of IndRNN each neuron processes the outputs of all previous layer neurons.

IndRNN- deeper and longer architectures

The basic architecture of IndRNN has been shown in the figure. 'weight' and 'recurrent+relu' denotes the input processing and recurrent process at every step with activation function of relu. By doing the stacking operation, a deep network can be constructed. As compared to LSTM architecture, it solves the problem of gradient vanishing over layers with time. batch normalization denoted as 'BN' can also be implemented before and after the activation function. Since the weight layer $Wx_t + b$ is the one that processes the input, it is only viable to extend it over layers in order to form deeper layers. The structure of IndRNN is quite simple and hence can be fitted in various architectures. In addition to stacking IndRNNs for input processing, they can also be stacked in the form of residual connections. Fig also shows the same mechanism. At each time stamp, identity mapping is used to propagate gradient to other layers. Also since it solves the problem of vanishing and exploding gradient , gradient can also be easily propagated over different time steps. This makes the network longer and deeper, which can also be trained

end to end.

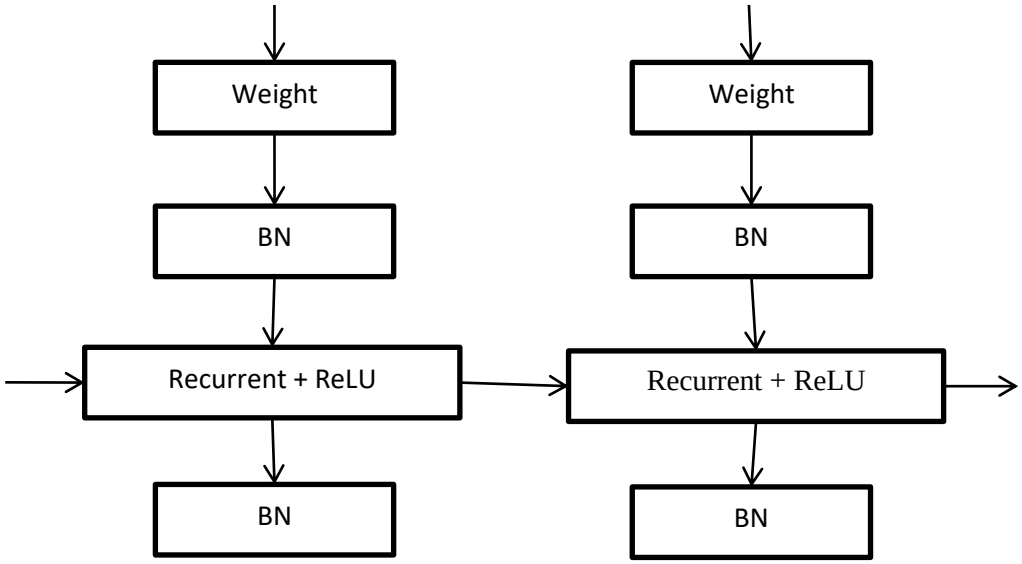


Fig. 6. Basic IndRNN architecture[16]

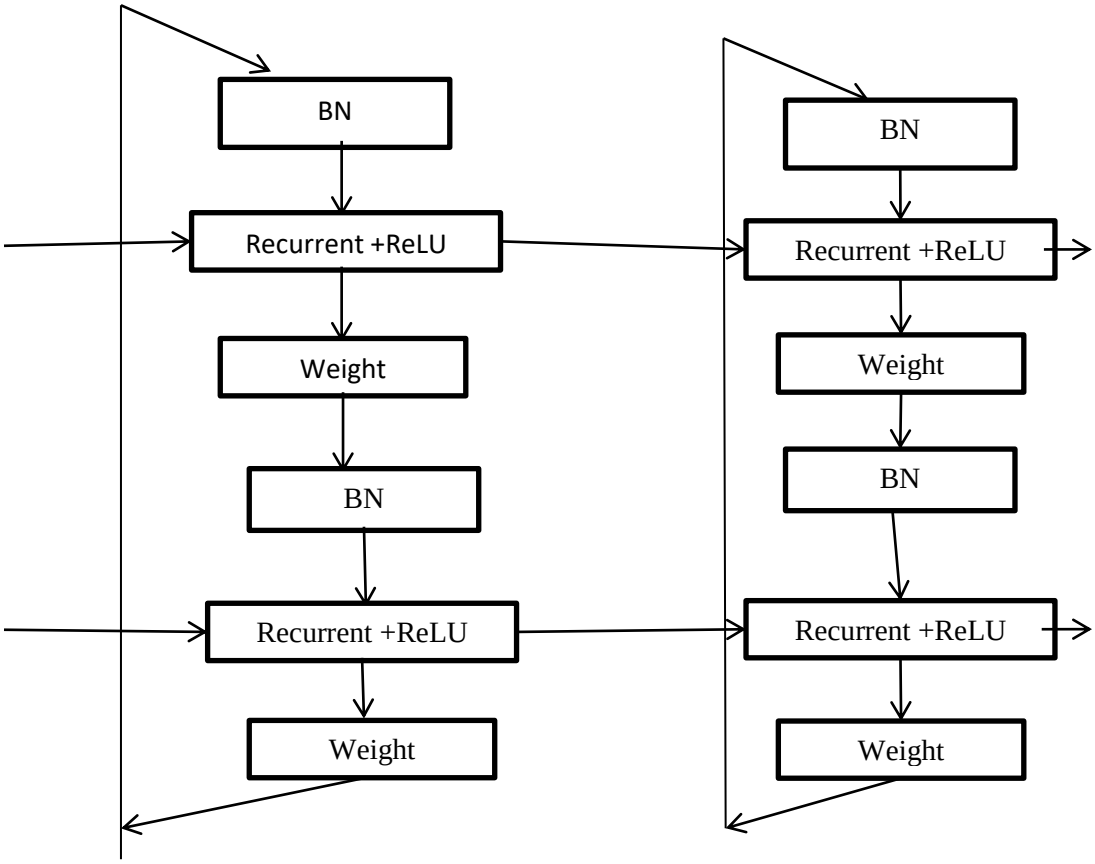


Fig.7. Residual IndRNN architecture[16]

CHAPTER 4.THE PROPOSED WORK

Problem denotation:

The problem of Named Entity Recognition has long been solved with various traditional methods like rule based ones and one which has gained the most popularity which is machine learning based CRF. The greatest challenge for this task is to find the features that can represent data well. Here the role of deep learning comes in. Our approach also makes use of deep learning model with the help of IndRNN which make us get better results for NER task. RNN has previously been used to process sequential data, but the problems of vanishing and exploding gradient have always prevailed. Also the problem of long term dependencies could be solved by IndRNN. Previously, LSTM and GRU has been employed to get away with the problems, but the activation functions of sigmoid and tangent still lead to decay when many layers are concerned. Therefore, a good deep learning network depends on a variety of factors. Also in the case of

LSTM and GRU, the neurons are pretty entangled together and difficult to interpret. Therefore, for the task of NER, IndRNN has been proposed. Multiple IndRNN can also be stacked together to given even deeper layers of training and eventually better results.

We present a novel approach of incorporating IndRNN for NER to deal with the problems being faced by RNN, LSTM and GRU.

The steps are as follows:

1. First of all, data cleaning takes place which includes preprocessing the data for removing irrelevant symbols and replacing tokens to remove Out Of Vocabulary words.
2. Preparation of dictionary takes place from the current vocabulary.
3. A bi-directional LSTM model is being applied on the data, and the results for various tags are being evaluated, in terms of precision, F1 score and recall.
4. The LSTM cell is then replaced with IndRNN cell in the model, in the following way:
   4.1. A forward cell, incorporated with IndRNN is being formed. To prevent over-fitting, dropout wrapper has been applied over the cell.
   4.2. A backward cell, again incorporated with IndRNN is being formed. Dropout is applied on this cell as well.
   4.3. Both the cells formed are then encapsulated into a single bidirectional RNN unit.
   4.4. A dense layer is formed on the top of it, whose output can further directly be used in the loss function.
   4.5. To finally compute the predictions, softmax activation function is applied to the last layer.
5. The parameters tuning is being done to get the best performance.

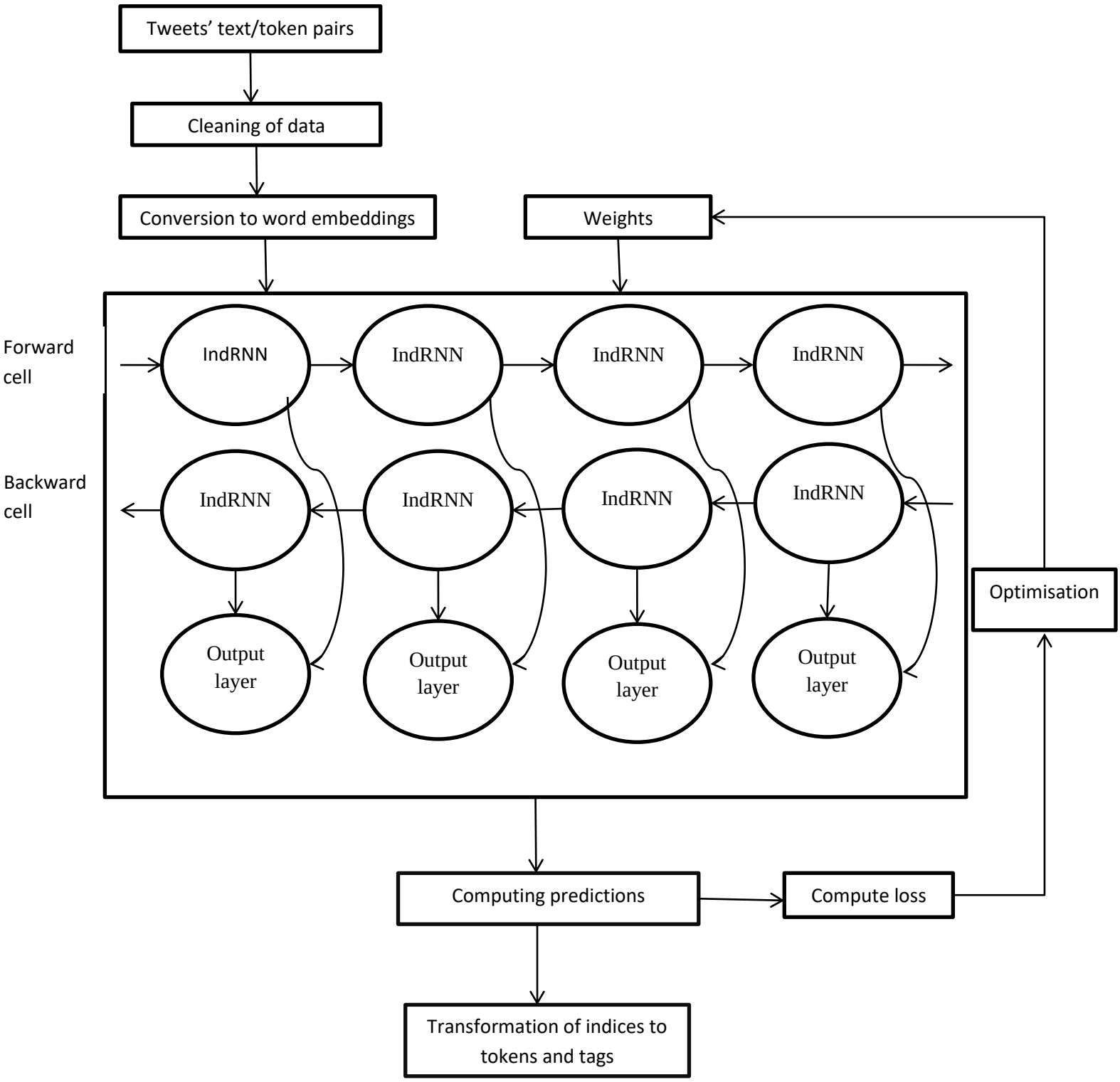


Fig.8. Flowchart demonstrating proposed approach

CHAPTER 5. RESULTS

Data set Used:

WNUT 2016

This dataset focuses on NLP which is applied to noisy user-generated data. This data can be found in social media, clinical records, web forums, essays, online reviews[17]

Evaluation metrics:

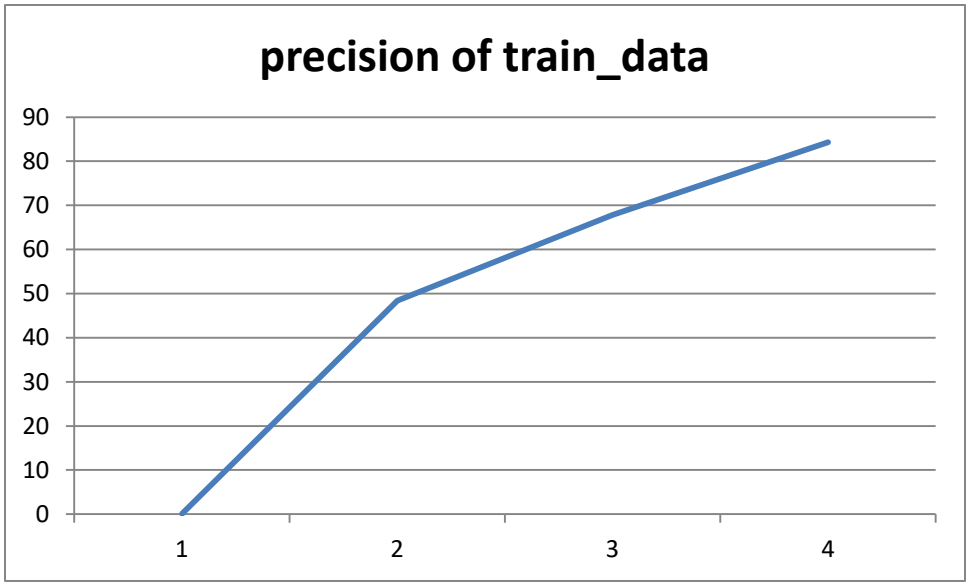
Precision= true positive/(true positive + false positive) *100

Recall= true positive/ (true positive +false negative) *100

F1 score= 2*precision*recall/(precision + recall)

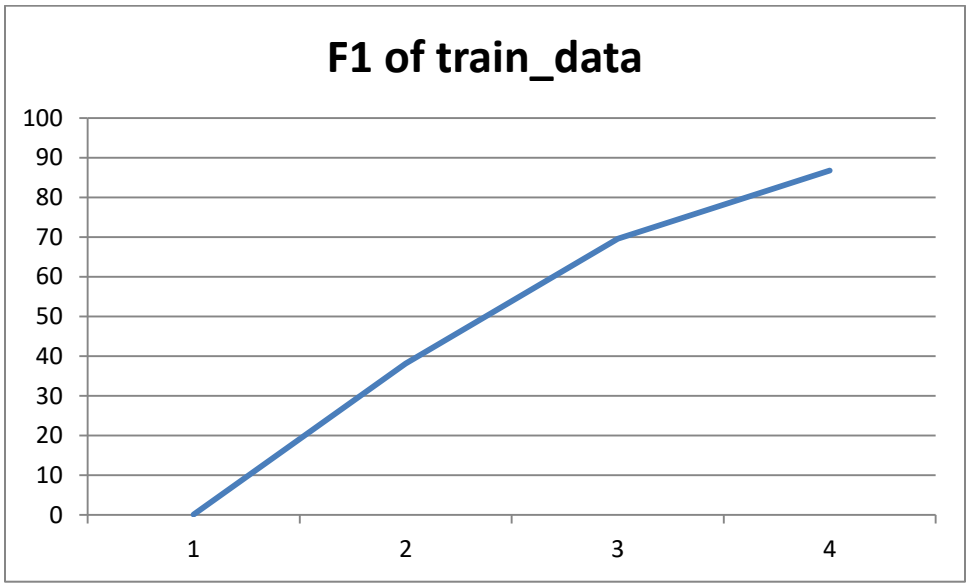
Training data Evaluation

Precision of training data



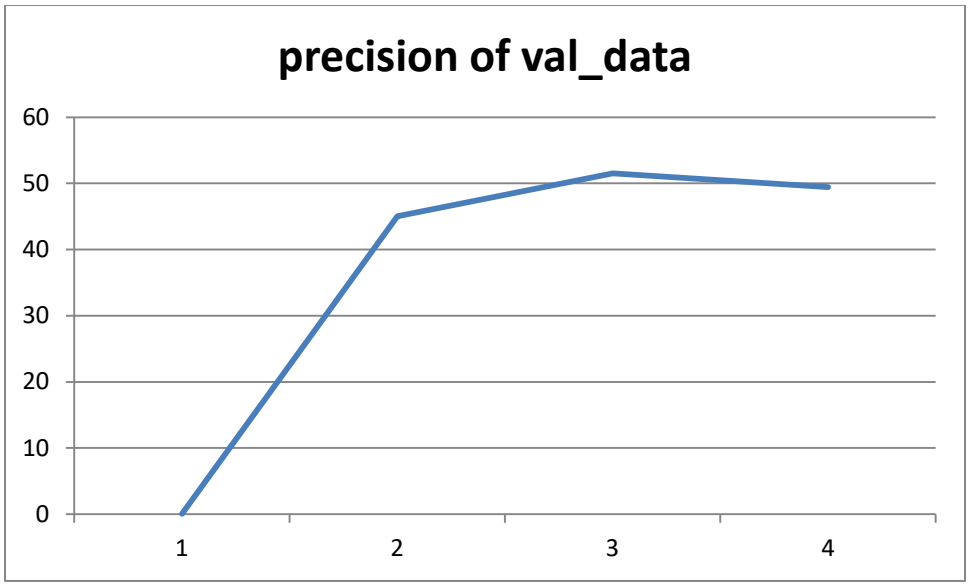
Recall of training data

F1 score of Training data

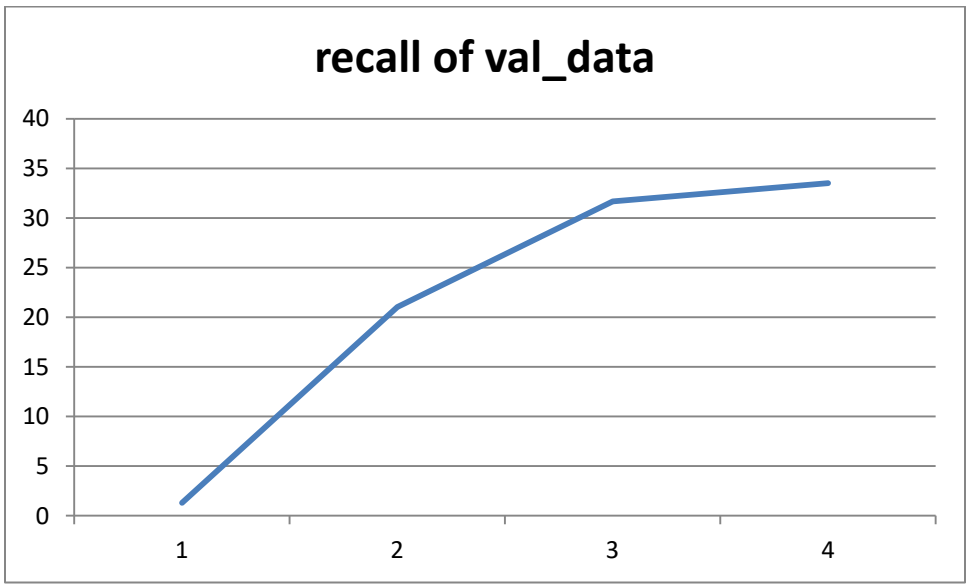


Validation data evaluation

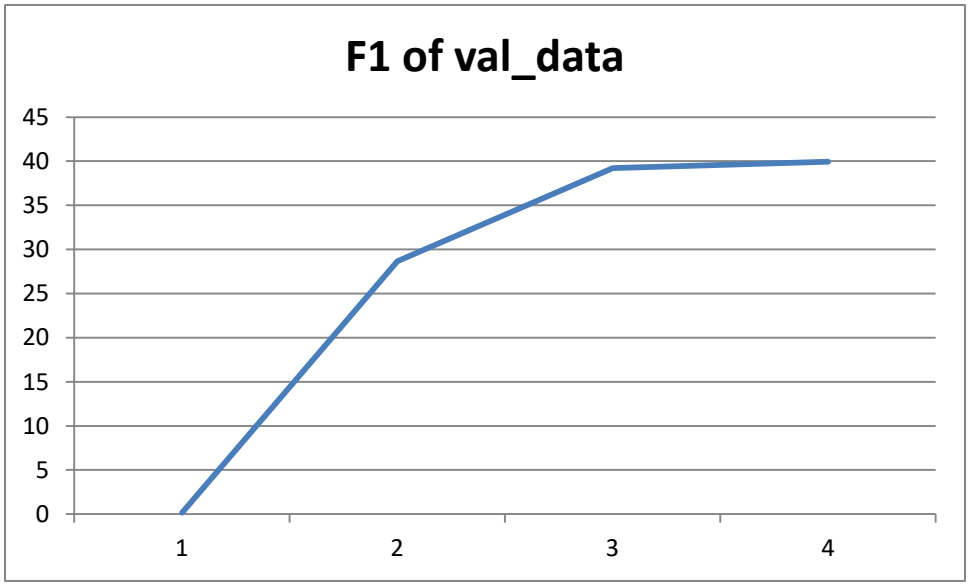
Precision



Recall of validation data

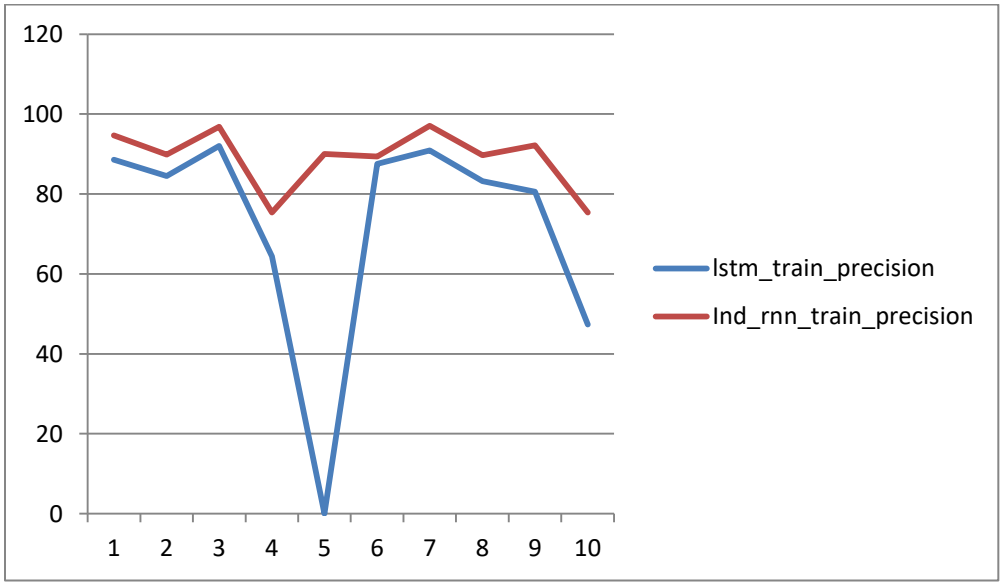


F1 of validation data

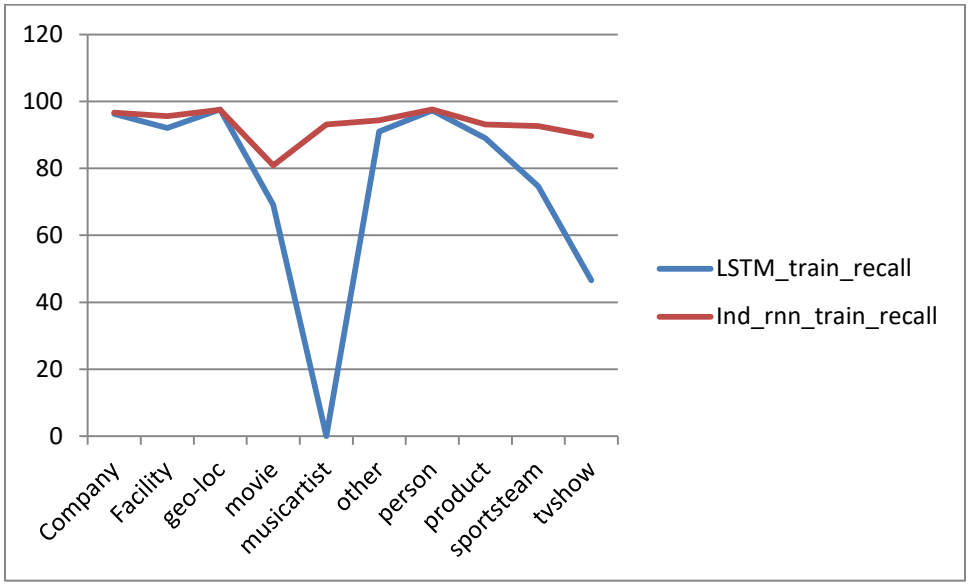


Comparison of LSTM and Ind_RNN for NER

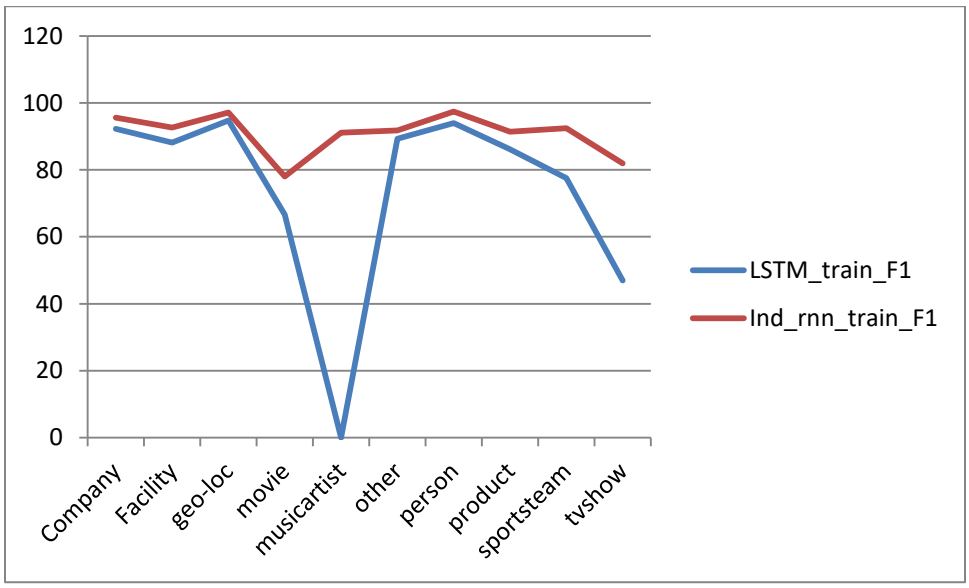
Precision



Recall

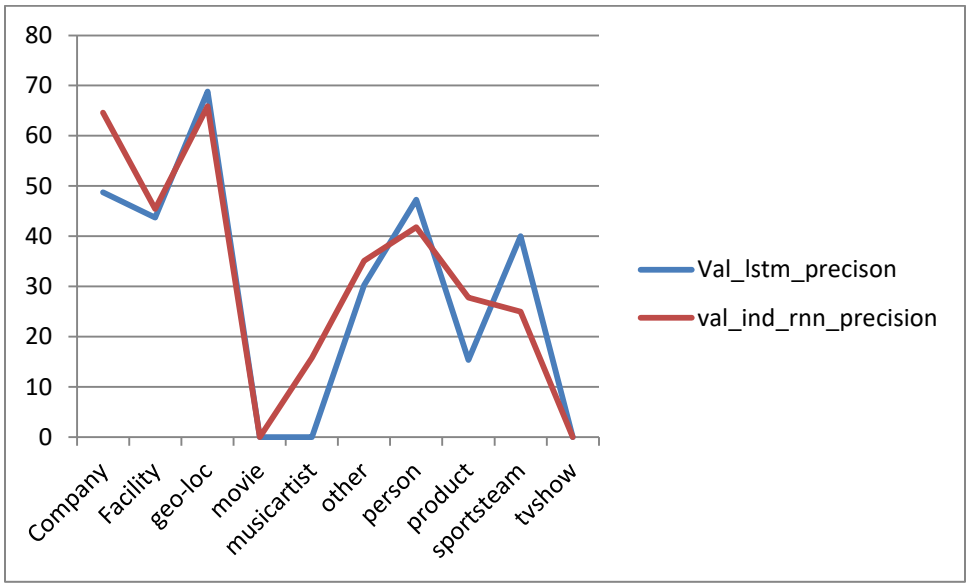


F1 score

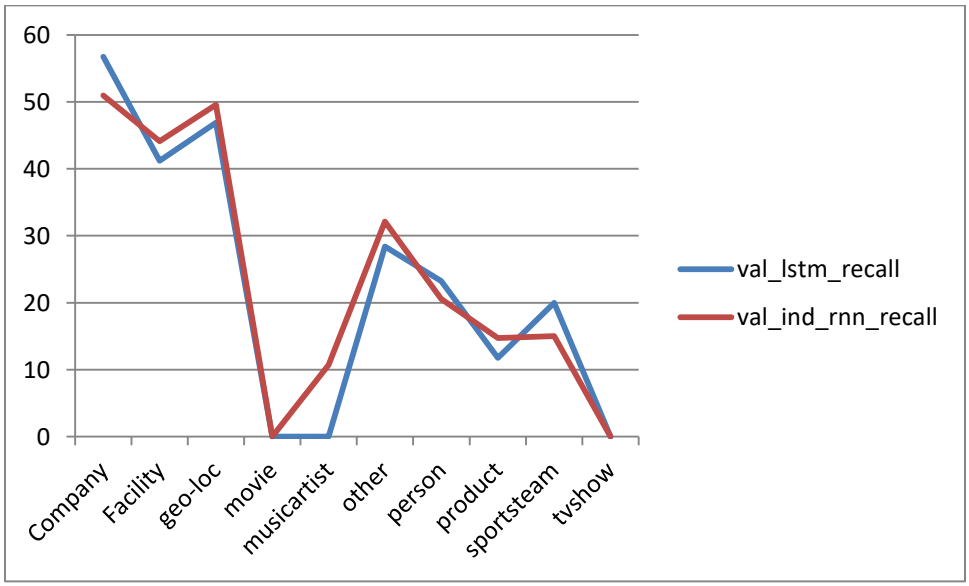


Validation set comparison

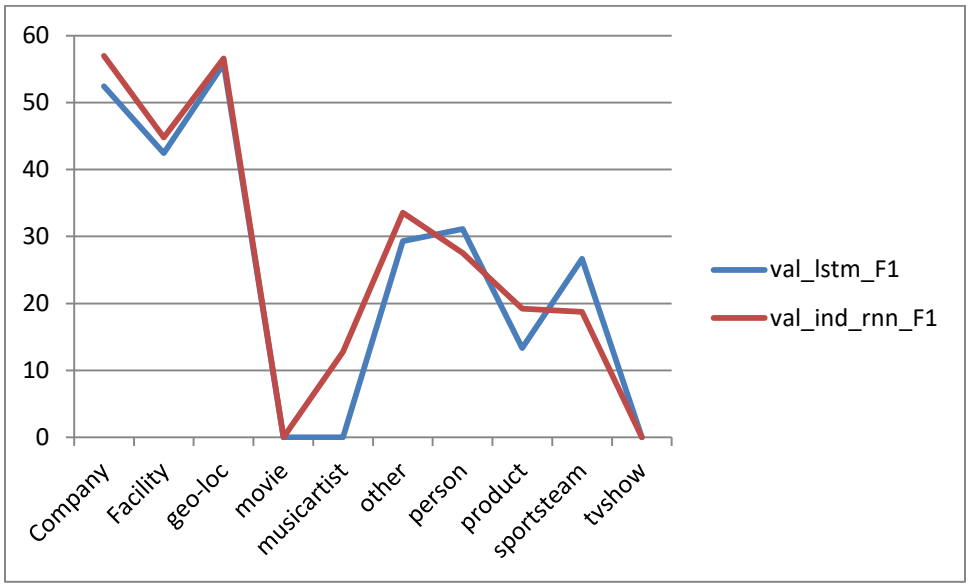
Precision



Recall

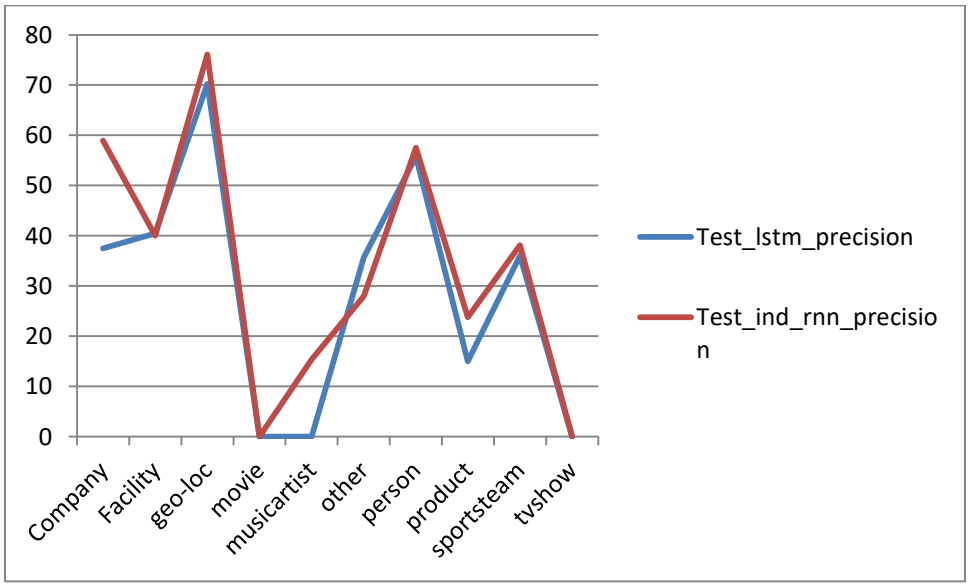


F1-score

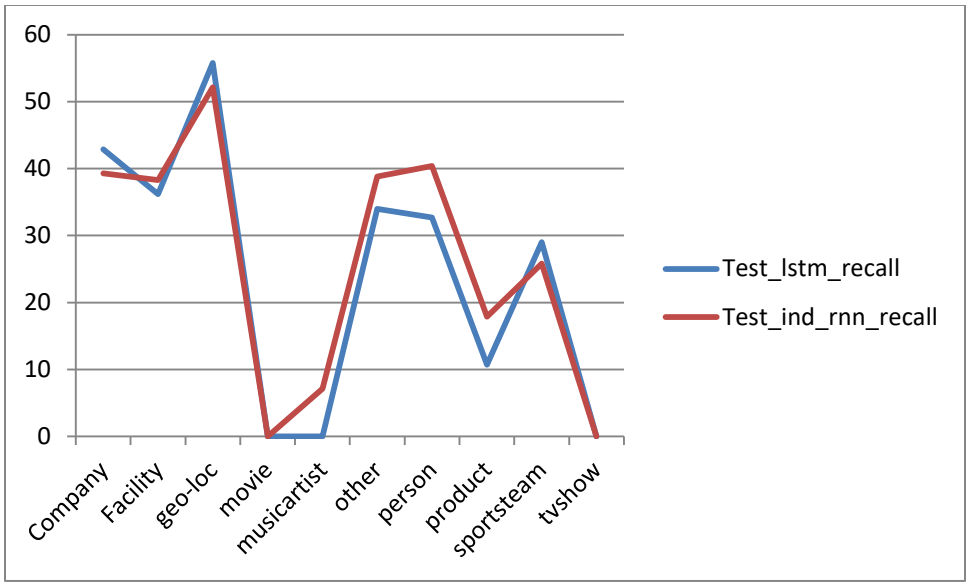


Test set comparison

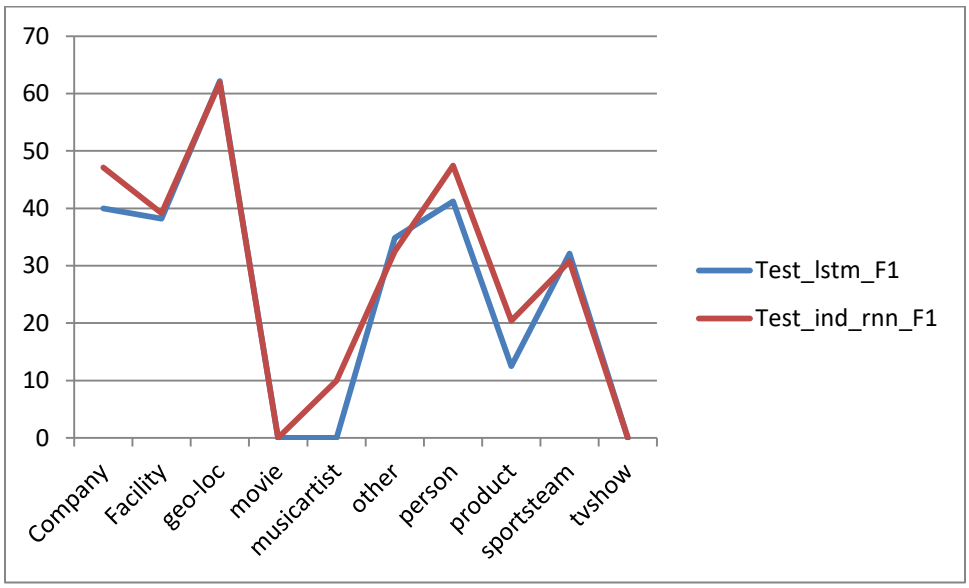
Precision



Recall



F1-score



Final Results:

Technique used	Precision	Recall	F1score
Bi-LSTM	46.89	37.42	41.62
Proposed	47.46	38.74	42.66

CHAPTER 6. CONCLUSION

This thesis work concludes with the fact that IndRNN can provide an improvement for NER task over state-of-the-art Bi-LSTM model. The experiments have been done on Twitter dataset and comparable result shave been derived. The evaluation metrics used are precision, recall and F1 score. NER is a sequential text processing task. Hence, LSTM and GRU have been used for it. But with one other alternative present, we are able to prove that IndRNN can be effectively applied for NER task which not only solves the problem of gradient vanishing over layers, but also provides an independence of RNN within a layer. The connection between text and context is being maintained by each RNN being connected to every other RNN in another layer. The experimental results have been shown for all, training dataset, validation dataset as well as test dataset.

CHAPTER 7. REFERENCES

[1] C. Li, A. Sun, J. Weng, and Q. He, “Tweet Segmentation and Its Application to Named Entity Recognition,” vol. 27, no. 2, pp. 558–570, 2015.

[2] V. H. Nguyen, H. T. N. B, and V. Snasel, “Named Entity Recognition in Vietnamese Tweets,” vol. 1, pp. 205–215, 2015.

[3] M. S. Powar, “Named Entity Recognition and Tweet Sentiment Derived From Tweet Segmentation using Hadoop .,” pp. 194–198, 2017.

[4] V. C. Tran, D. Hwang, and J. J. Jung, “Semi-supervised Approach Based on Co-occurrence Coefficient for Named Entity Recognition on Twitter,” pp. 141–146, 2015.

[5] V. Rachman, S. Savitri, F. Augustianti, and R. Mahendra, “Named Entity Recognition on Indonesian Twitter,” 2017.

[6] V. C. Tran, N. T. Nguyen, H. Fujita, D. T. Hoang, and D. Hwang, “A combination of active learning and self-learning for named entity recognition on Twitter using conditional random Þelds,” *Knowledge-Based Syst.*, 2017.

[7] M. S. Salleh, S. A. Asmai, H. Basiron, and S. Ahmad, “A Malay Named Entity Recognition Using Conditional Random Fields,” vol. 0, no. 5th International Conference on Information and Communication Technology (ICoIC7), pp. 1–6, 2017.

[8] D. Esteves, R. Peres, and J. Lehmann, *Named Entity Recognition in Twitter Using Images and Text*, vol. 3. Springer International Publishing, 2018.

[9] M. Khalifa and K. Shaalan, “Character Convolutions for Arabic Named Entity Recognition with Long Short-Term Memory Networks,” *Comput.*

Speech Lang., 2019.

[10] "Recurrent Neural Networks and LSTM." [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5.

[11] R. Cascade-correlation and N. S. Chunking, "LONG SHORT TERM MEMORY," vol. 9, no. 8, pp. 1–32, 1997.

[12] U. of T. Alex Graves , Abdel-rahman Mohamed and Geoffrey Hinton Department of Computer Science, "SPEECH RECOGNITION WITH DEEP RECURRENT NEURAL NETWORKS," no. 6, pp. 6645–6649, 2013.

[13] X. Ma and E. Hovy, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," pp. 1064–1074, 2016.

[14] V. Raaj, "Named Entity Recognition using Bi-directional Long Short-Term Memory (Bi-LSTM)." [Online]. Available: https://medium.com/@vivanraaj/named-entity-recognition-using-bi-directional-long-short-term-memory-bi-lstm-ab54bbf7cc76.

[15] B. Van Merri and C. S. Fellow, "Learning Phrase Representations using RNN Encoder – Decoder for Statistical Machine Translation," pp. 1724–1734, 2014.

[16] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently Recurrent Neural Network (IndRNN): Building A Longer and Deeper RNN," pp. 5457–5466, 2018.

[17] "2016 The 2nd Workshop on Noisy User-generated Text (W-NUT)," 2016. [Online]. Available: http://noisy-text.github.io/2016/index.html.