# Survey on Recurrent Neural Network in Natural Language Processing

Kanchan M. Tarwani[#1], Swathi Edem[*2]

[#1] *Assistant Professor, Department of Information Technology, Chaitanya Bharathi Institute of Technology(Autonomous), Hyderabad, India*

[*2] *Assistant Professor, Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology(Autonomous), Hyderabad, India*

*Abstract — Natural Language Processing(NLP) is a way for computers to analyze, understand, and derive meaning from human language in a smarter way. Recurrent neural networks (RNN) have revolutionized the field of NLP. RNNs are used at modelling units in sequence.*

*Unlike feed forward neural networks, RNNs have cyclic connections making them more powerful for modeling inputs of sequences. They have been successfully used for sequence labeling and sequence prediction tasks, such as handwriting recognition, language modeling, machine translation, phonetic labeling of acoustic frames and etc. This paper gives an overview of how RNNs are being used and capable of dealing with Natural Language Processing. This paper also summarizes LSTM based RNNs architectures. .*

Keywords — *Recurrent Neural Network(RNN), Natural Language Processing(NLP), Back Propagation Through Time (BPTT), Long Short Term Memory (LSTM).*

## I. INTRODUCTION

Natural Language Processing is a field that covers computer understanding and manipulation of human language and its ripe with possibilities for gathering information and news. By utilizing NLP[9], developers can organize and structure knowledge to perform various tasks such as automatic summarization, translation, named entity recognition, relationship extraction, semantic analysis, sentiment analysis, speech recognition, and topic segmentation.

Apart from common word processor operations that treat text like a mere sequence of symbols or words, NLP[8] also considers the hierarchical structure of language such as several words make a phrase, several phrases make a sentence and, ultimately, sentences convey thoughts or ideas.

A recurrent neural network (RNN) is a type of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic behavior. Unlike feed forward neural networks, RNNs can use their internal memory to process and work on arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition[1] , speech recognition[2], Natural language processing, Machine Translation etc. This is another class of neural network that is dominating difficult machine learning problems that involve sequences of inputs called recurrent neural networks.

Recurrent Neural Networks were created in the 1980's but have just been recently gaining popularity from advances to the networks and increased computational power from graphic processing units. They are especially useful with sequential data because each neuron can use its internal memory to maintain information about the previous input. Another way to think about RNNs is that they have a "memory" which keeps/captures information about what has been calculated so far. Fig. 1 shows a rolled up RNN[3] where $X_t$ is the input vector containing sequences of characters of a words while $H_t$ is as ouput vector.

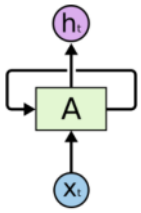


**Fig 1: A Rolled up RNN**

## II. RNN IN NATURAL LANGUAGE PROCESSING

Considering an example from a natural language such as, "I had washed my house" is much more different than "I had my house washed". This allows the network to gain a deeper understanding of the language statements. This is important to note because reading through a sentence even as a person, you're picking up the context of each word from the words before or after it.

A RNN has loops in them that allow information to be carried across neurons while reading input.

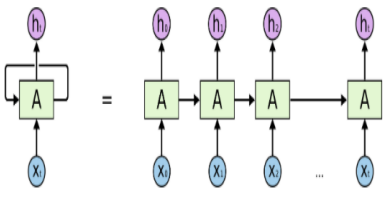


**Fig 2: An Unrolled RNN**

In these diagrams $x_t$ is some type of input, $A$ is a part of the RNN and $h_t$ is its output. Essentially you can feed in language words from the sentence or even characters from a string as $x_t$ and through the RNN it will result with a $h_t$. The goal is to use $h_t$ as output and compare it to test data (which is usually nothing but a small subset of the original data). Then we will get error rate information. After comparing output to the test data, with error rate in hand, we can use a new technique called Back Propagation Through Time (BPTT)[5] which back checks through the network and adjusts the weights based on error rate and makes it learn to get a good result.

The idea behind using RNNs for Natural Language Processing is to make use of sequential information. In a traditional neural network, we consider that all inputs (and outputs) are independent of each other. But for many tasks that's not a good idea. If we want to predict the next word in a sentence we should know which word came before it. RNNs are called *recurrent* because they perform the same task for every element of a sequence of input, with the output being depended on the previous computations as well as current input.

RNNs can handle context from the beginning of the statement which will allow more accurate predictions of a word at the end of a statement. In practice this isn't necessarily true for all type of RNNs, since RNNs are actually limited to looking back only a few steps. This is a major reason why RNNs need to be used with a Long Short Term Memory (LSTM) to achieve great results. Adding the LSTM to the network is like adding a memory unit inside the network that can remember context from the very beginning of the input. So, if the sentence is of 10 words and we want to predict 11th word, all 10 words are being processed by RNNs and their weights on each step are being saved using LSTM and the probability of 11th word being predicted accordingly.

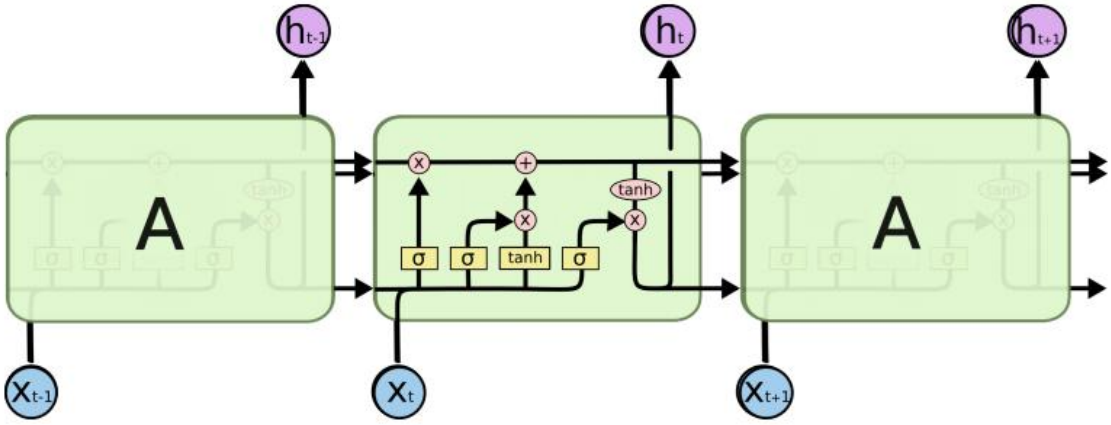


**Fig 3: Long short term memory Network**

A memory cell is used in addition to the hidden layer to pass information that might not be used in prediction. These little memory units allow for RNNs to give much more accurate results. These memory units allow the network to remember the context across inputs.

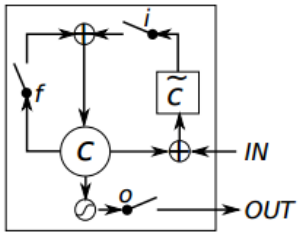


**Fig 4: Illustration of Long short term memory**

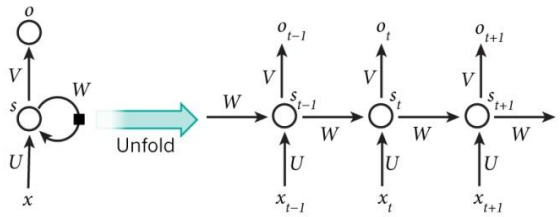


**Fig 5: An unfolded RNN**

The Fig. 5 shows a RNN being *unrolled* (or unfolded) into a fully connected network. By unrolling we simply mean that we write out the network for the whole sequence. For example, if the sequence we care about is a sentence of 6 words, the network would be unrolled into a 6-layer neural network, one layer for each word. The formulas that show the computation happening in a RNN for NLP are as follows:

- $x_t$ is the input at time step $t$. For example, $x_t$ could be a one-hot vector corresponding to the second word of a sentence.

- $s_t$ is the hidden state at time step $t$. It's the memory" of the network. $s_t$ is calculated based on the previous hidden state and the input at the current step: $s_t=f(Ux_t+Ws_{t-1})$. The function $f$ usually is a nonlinearity such as tanh or ReLU. $s_{-1}$, which is required to calculate the first hidden state, is typically initialized to all zeroes.
- $o_t$ is the output at step $t$. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary.
  $o_t= \text{softmax}(Vs_t)$

The detailed explanation of the problem setup is as follows:
Training data is a large set of symbol sequences (words in sentences),

$Dtrain = \{x^1, \ldots, x^M\}$

$x = [x_1, \ldots x_N]$

for example (notice the terminal symbol):
X =[its, water, is, so, transparent, STOP]
We want to learn a model that returns:
P(X), for $\forall X \in V^{MaxN}$ where V is the vocabulary and $V^{MaxN}$ all possible sentences.

$P(X)= P(X_{1,\ldots\ldots}X_N)$
$P(X_1)P(X_2\ldots\ldots X_N|X_1)$
$P(X_1)P(X_2|X_1)\ldots\ldots.. P(X_N|X_1)$

So, the probabilities as logistic regression can be written as:

$$P(xn = k|xn-1\ldots x1) = \frac{\exp(wk \cdot \phi(xn-1\ldots x1))}{\sum k' = 1 \text{ to } V \exp(wk \cdot \phi(xn-1\ldots x1))}$$

$w_k$ are the weights for word k, and $\phi(xn-1 \ldots x1)$ are the features extracted from the previous words (one-hot encoding of $xn-1\ldots\ldots x1$) ,

$p(xn|xn-1\ldots x1) = \text{sof tmax}(W\phi(xn-1\ldots x1))$

$W \in \Re^{|V|*|contexts|}$ has weights for each wordand the context with a vector $s_{n-1} \in \Re^d$

$p(xn|Xsn-1\ldots x1) = \text{sof tmax}(VSn-1)$

$V \in \Re^{|V|*d}$ maps the context to a probability distribution over the words,
Now to get Sn-1, we need RNN

$s_n = \sigma(Ws_{n-1} + Ux_n)$

$sn-1 \in \Re d$ is the ``memory'' of the context $X_{n-1.}$
$W \in \Re^{d*d}$ until word controls how this memory is passed on.
$U \in \Re^{|v|*d}$ is the matrix containing the word vectors for all the words, $X_n$ picks one.

And to get the probability distribution for word $X_n$:

$O_{n-1} = p(x_n|x_{n-1}\ldots x_1) = \text{softmax}(Vs_{n-1})$

We need to learn the word vectors U, hidden and output layer parameters W, V. Standard back-propagation can't work because of the recurrence, i.e. we reuse the hidden layer parameters W. Back-propagation through time (BPTT)[5] is used to unroll the graph for n steps and sum the contributions of the gradients in updating.

## III. OTHER APPLICATIONS OF RNN

There are many different applications of RNNs. A great application is in collaboration with Natural Language Processing (NLP). RNNs have been demonstrated by many people who created amazing models that can represent a language model. These language models can take input such as a large set of shakespearean poems, and after training these models they can generate their own Shakespearean poetics that are very difficult to differentiate from originals.

This particular type of RNNs is fed in a dataset of text and reads the input in character by character. The amazing thing about these networks in comparison to feeding in a word at a time is that the network can create it's own unique words that were not in the vocabulary you trained it on.

Another amazing application of RNNs is machine translation [6]. This method is interesting because it involves training two RNNs simultaneously. In these networks the inputs are pairs of sentences in different languages. For example you can feed the network an English sentence paired with its French translation. With enough training you can give the network an English sentence and it will translate it to French! This model is called a Sequence 2 Sequences model or Encoder Decoder model.

Fig. 6 shows how information flows through Encoders Decoder model. This diagram is using a word embedding layer to get better word presentation. A word embedding layer is usually GloVe or Word2Vec algorithm that just takes a bunch of words and creates a weighted matrix which allows similar words to be correlated with each other. Using an embedding layer generically makes the RNN more accurate because it is a better representation of how similar words are so the network has less to infer.

One more very important application of RNN is slot filling in spoken language understanding [5]. A major task in spoken language understanding in goal-oriented human-machine conversational understanding systems is to automatically extract semantic concepts, or to fill in a set of arguments or slots embedded in a semantic frame, in order to achieve a goal in a human-machine dialogue.

LSTM based RNN architectures are also play an important role to train acoustic models for large vocabulary speech recognition [7].

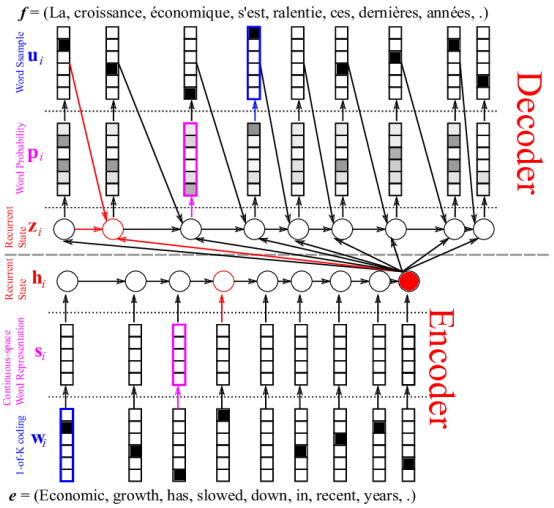


**Fig 6: Encoder Decoder RNN for English to French translation**

## IV. CONCLUSIONS

Recurrent Neural Networks have become very popular as of recently because of their cyclic connections. They deal with the input of sequences and have proved to be one of the most effective models for natural language processing. This paper provides an insight into NLP with RNNs, LSTM based RNNs and their various applications.

## REFERENCES

[1] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009.

[2] **J**ump up^ H. Sak and A. W. Senior and F. Beaufays., "Long short-term memory recurrent neural network architectures for large scale acoustic modelling". Proc. Interspeech, pp338-342, Singapore, Sept. 2010

[3] Hochreiter and Schmidhuber , Long Short Term Memory, in the journal of Neural Computation 9(8):1735{1780, 1997

[4] Schmidhuber and Cummins , "Learning to forget: Continual prediction with LSTM", in the Technical Report IDSIA-01-99 January, 1999.

[5] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, using recurrent neural networks for slot filling in spoken language understanding", IEEE/ACM Transactions on Audio, Speech, and Language Processing ( Volume: 23, Issue: 3, March 2015 )

[6] Andi Hermanto, Teguh Bharata Adji, Noor Akhmad Setiawan, "Recurrent neural network language model for English-Indonesian Machine Translation: Experimental study" IN Science in Information Technology (ICSITech), 2015 International Conference/IEEE transaction,( February 2016)

[7] Haşim Sak, Andrew Senior, Françoise Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition", arxiv.org, Feb 2014.

[8] Rijuka Pathak , Somesh Dewangan," Natural Language Chhattisgarhi: A Literature Survey", International Journal of Engineering Trends and Technology (IJETT) .(Volume -12, Number-2,2014).

[9] Mr. Roshan R. Karwa , Mr.M.B.Chandak ," Word Sense Disambiguation: Hybrid Approach with Annotation Up To Certain Level – A Review", International Journal of Engineering Trends and Technology (IJETT) .(Volume -18, Number-7,2014).

[10] K.Brahmani , K.S.Roy , Mahaboob Ali," Arm 7 Based Robotic Arm Control By Electronic Gesture Recognition Unit Using Mems", International Journal of Engineering Trends and Technology (IJETT) .(Volume -4, Issue-4,2013).