International Journal of Computer Architecture and Mobility (ISSN 2319-9229)
Volume 8 - Issue 7, July 2020

Detection of Sarcasm on Social Networks Using Neural Networks

Pratham Patel[1], Hardik Soni[2], Bhavishya Lohia[3]

Department of Computer Science Engineering, Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

[1]Email: prathamashokpatel@gmail.com
[2]Email: soni.hardik936@gmail.com
[3]Email: bhavishya.lohia@gmail.com

***Abstract*— An innovative knowledge-based technique for sarcasm detection using sentimental analysis by using Recurrent Neural Networks (RNN). The proposed methodology learns the typical behaviour of sarcasm on social platforms by using sentiment analysis models and word embedding.**

***Keywords*— Recurrent Neural Networks, Word Embedding, Long short term memory.**

I. Introduction

Sarcasm are a nuanced type of language where often the speaker explicitly states the alternative of what they require to imply. Organizations tap into social media for belief on their products and services and real time customer assistance. to help this Sentimental analysis is that the key offering in any and each CRM tool. Customers often use sarcasm to precise their review/ frustration with their Product/ services. Sarcasm can even reflect a state of ambivalence. It contradicts the meaning, in the context which is alleged. Sarcasm are often expressed in many ways. It is often expressed in speech and text. Sarcasm are often conveyed through various ways like a direct conversation, speech, text etc. In direct conversation, countenance and body gestures provide the hint of sarcasm. within the speech, sarcasm is often inferred if there's any change in tone.

Most sentiment analysis system fails miserably in handling sarcasm. Most sentimental analysis systems lack the sophistication needed to detect sarcasm. very similar to QnA, text summarization, computational linguistics, sarcasm detection involves complexity of language and is believed to be a way harder task. Any progress in sarcasm detection, is an affirmative step towards pushing the boundaries of tongue processing (NLP).

Through our method we would like to seek out the proportion of sarcasm in any sentence (tweet). If sarcasm is present directly within the tweet, it will detect its present and provides one hundred pc as an output. There are often some sentences where we can't know the of use of sarcasm but through our (RNN), Long Short-Term Memory (LSTM) and word embedding.

II. Recurrent Neural Network (RNN) Model

Recurrent Neural Network could be a generalization of feedforward neural network that has an indoor memory. RNN is known as recurrent neural network, as the name suggest it perform identical function for each input producing an output that is used by the next stage, we can say that it is working recurrently. After producing the output, it's copied and sent back to the recurrent network. In order to make a choice, the model uses the input and the outputs that are been acquired from previous states. Sometimes the previous output can also be used as an input for the next state.

Unlike feedforward neural networks, RNNs can use their memory to process sequences of inputs. This makes them applicable to tasks like unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of every other. But in RNN, all the inputs are associated with one another.

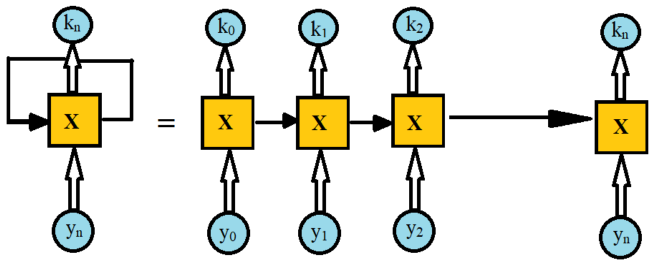


Fig. 1 An unrolled recurrent neural network

First, we will take the input y_0 and then outputs k_0 will be obtained, which together with y_1 will be the input for the next step. Now, the k_0 and y_1 will be the input for the next step. Similarly, k_1 from the next step is the input along with k_2 for the next step and so on. This way, the model will keep remembering the context while training.

The formula for the current state is: -

$$k_n=f\,(k_{n-1}, y_n)$$

Detection of Sarcasm on Social Networks Using Neural Networks

Activation Function:

$$k_n = \tanh(W_{kk}k_{n-1} + W_{yk}y_n)$$

Here,

W is *weight*, **k** is the *single hidden vector*, **W**$_{kk}$ is *the weight at previous hidden state*, **W**$_{yk}$ is the *weight at current input state*, **tanh** is the *Activation function.*

Output:

$$I = W_{kI}\,k_n$$

Here, I is the *output state*. **W**$_{kI}$ is the *weight at the output state.*

III. Word Embedding

Word embedding is one of the most popular concept of document vocabulary. It is capable of capturing context of a word in a document, syntactic and semantic similarity, relation with other words, etc.

For example: We have two sentences “Have a good day” and “Have a great day”. If we construct a dictionary (let’s call it D), it would have D = {Have, a, good, great, day}.

Let us create ‘a’ encoded vector for every of the word in D. Length of our one-hot encoded vector would be adequate to the dimensions of D (=5). we might have a vector of zeros aside from the element at the index representing the corresponding word within the vocabulary. that exact element would be one. The encodings below will help us to elucidate better.

Have= [1,0,0,0,0]` ; a= [0,1,0,0,0]` ; good= [0,0,1,0,0]` ; great= [0,0,0,1,0]` ; day= [0,0,0,0,1]` (` represents transpose)

If we attempt to examine these encodings, we will think about a 5-dimensional space, where each word occupies one amongst the size and has nothing to try and do with the remainder (no projection along the opposite dimensions). this suggests ‘good’ and ‘great’ are as different as ‘day’ and ‘have’, which isn't true.

IV. Long Short-Term Memory (LSTM)

Long Short-term memory (LSTM) are advanced version of RNN. The previous stages (past data) is easily memorized by using LSTM. We now know that RNN works by taking input as the output generated (feedback) from the previous state, which is stored in for a short time (Short Term Memory). There are many applications of RNN like speech recognition, Image processing, Music composition etc. There are also drawbacks of using RNN which are the problem of ***Vanishing gradient***, inability to store the information for a longer period of time, there is no control over which part of the stage needs to be carried forward and how much of the past stage needs to be forgotten**.**

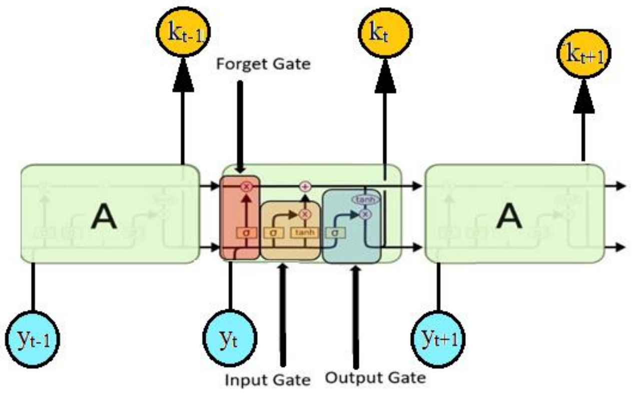


Fig. 2 A LSTM model

- *Input Gate*— The input gate helps to make changes in the memory.

  Here,

  The ***Sigma*** is the decision-making parameter which decides that which value is to be let in (0 or 1).

  Tanh assigns the priority to the stages from range (-1,1)

$$i_t = \sigma(W_i.[k_{t-1}, y_t] + b_i)$$

$$C_t = \tanh(W_C.[k_{t-1}, y_t] + b_c)$$

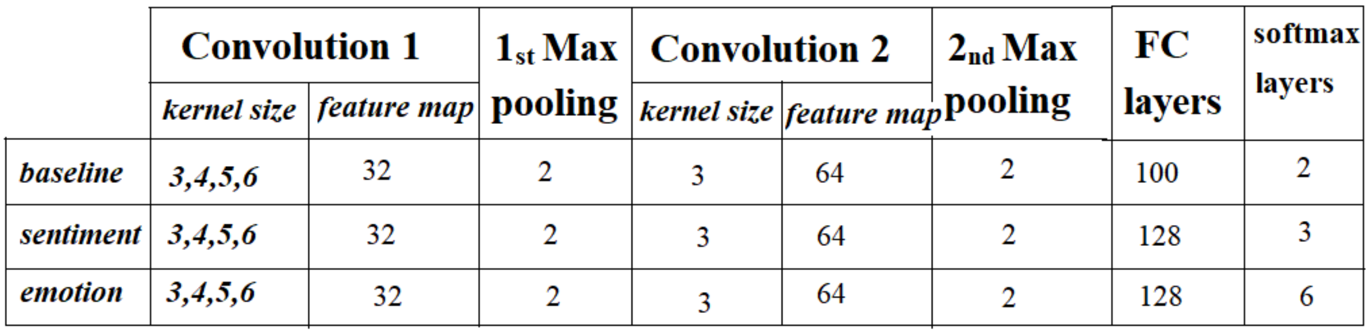
- *Forget gate*— As the name suggest this gate is used to forget (Remove) the data which are no longer in use. It is defined by the sigma (σ).
- *Output gate*— The Output gate is used to determine that which state will be generated from the previous cell state that was been given as an input.

$$O_t = \sigma(W_o[k_{t-1}, y_t] + b_o)$$

$$K_t = O_t * \tanh(C_t)$$

Detection of Sarcasm on Social Networks Using Neural Networks

V. Model Architecture



	Convolution 1		1st Max pooling	Convolution 2		2nd Max pooling	FC layers	softmax layers
	kernel size	*feature map*		*kernel size*	*feature map*			
baseline	***3,4,5,6***	32	2	3	64	2	100	2
sentiment	***3,4,5,6***	32	2	3	64	2	128	3
emotion	***3,4,5,6***	32	2	3	64	2	128	6

TABLE I
MODEL ARCHITECTURE

VI. Conclusion

So, using these deep leaning techniques, the whole system is analysed and then Algorithm approximates the results. Through our neural network model, we can get up to 88% accuracy of determining that weather the tone of the individual is sarcastic or not, which is more favourable for a computer system to detect and return the positive results, for almost 9 out of 10 cases.

There are best results obtained from RNN and LSTM model than that of Logistic Regression.

Scope of future work:

1. We can train our own word embeddings.
2. We can also try other combinations of neural networks.
3. Collection of more data: - collecting right data for negative class is very important.

References

[1] Anuj Gupta: *Sarcasm detection Achilles Heel of sentiment analysis*, YouTube.

[2] Aditi Mittal, *"Understanding RNN and LSTM"* , towardsdatascience.com, Oct 12, 2019.

[3] Dr. Vadivu, G Sindhu Chandra Sekharan, SRM Institute of Technology, ResearchGate.

[4] Sepp Hochreiter & Jürgen Schmidhuber, *Neural Computation 9(8)*: December 1997.

[5] Elgibbor SMS Nigeria– *division of ElgibborTech and Consult*.

[6] Sophia Turol, *Text Prediction with TensorFlow and Long Short-Term Memory*, May 11, 2017.

[7] NPTEL> Courses > Multidisciplinary > NOC: Fuzzy Logic and Neural Networks (Video), *IIT Kharagpur*, Nov 26, 2018.

[8] Department of Information Technology Vasant Kunj, IEEE, New Delhi.11.

[9] Google Images- *all formulae references, with Flow Charts.*