SARCASM DETECTION ON TWITTER DATA

By

Soumyabrata Maity
Kumarjit Ghosh
Nishan Singh Sekhon
Rohit Kumar Jaiswal

UNDER THE GUIDANCE OF

Mrs. Sukla Banerjee

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF INFORMATION TECHNOLOGY AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Session 2017-2018

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY [Affiliated to West Bengal University of Technology] CANAL SOUTH ROAD, BELIAGHATA, KOLKATA-700105

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

TO WHOM IT MAY CONCERN

I hereby recommend that the Project entitled
SARCASM DETECTION ON TWITTER DATA
prepared under my supervision by
Soumyabrata Maity (Reg. No 141170110070**, Class Roll No**. CSE/2014/066**)**
Kumarjit Ghosh (Reg. No 141170110038**, Class Roll No**. CSE/2014/070**)**
Nishan Singh Sekhon (Reg. No 141170110042**, Class Roll No**. CSE/2014/068**)**
Rohit Kumar Jaiswal (Reg. No 141170110052**, Class Roll No**. CSE/2014/061**)**
of B.Tech 8th Semester, may be accepted in partial fulfillment for the degree of **Bachelor of Technology in Computer Science & Engineering** under **Maulana Abdul Kalam Azad University of Technology** (**MAKAUT**).

….....

Project Supervisor

Department of Computer Science and Engineering

RCC Institute of Information Technology

Countersigned:

….....

Head

Department OF Computer Sc. & Engineering,

RCC Institute of Information Technology

Kolkata- 700105

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

CERTIFICATE OF APPROVAL

The foregoing Project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR

EVALUATION OF PROJECT

1. .....

2. .....

(Signature of Examiners)

ACKNOWLEDGEMENT

I take the opportunity to express my profound gratitude and deep regards to our mentor Mrs. Sukla Banerjee for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. Her relentless effort for teaching in the right way and in the correct manner has helped us to attain a high standard in this aspect.

Also, not to forget the coordinated work of our group and hereby we thank each other for the successful completion of the documentation.

.....

Soumyabrata Maity (CSE/2014/066)
Kumarjit Ghosh (CSE/2014/070)
Nishan Singh Sekhon (CSE/2014/068)
Rohit Kumar Jaiswal (CSE/2014/061)

Table of Contents

Page No.

Introduction

Sarcasm is defined as a cutting, often ironic remark intended to express contempt or ridicule. Sarcasm detection is the task of correctly labeling the text as 'sarcastic' or 'non-sarcastic'. It is a challenging task owing to the lack of intonation and facial expressions in text. Nonetheless humans can still spot a sarcastic sentiment in the text and reason about what makes it so.

Recognizing sarcasm in text is an important task for Natural Language processing to avoid misinterpretation of sarcastic statements as literal statements. Accuracy and robustness of NLP models are often affected by untruthful sentiments that are often of sarcastic nature. Thus, it is important to filter out noisy data from the training data inputs for various NLP related tasks. For example, a sentence like" So thrilled to be on call for work the entire weekend!" could be naively classified as a sentence with a high positive sentiment. However, its actually the negative sentiment that is cleverly implied through sarcasm.

The use of sarcasm is prevalent across all social media, micro-blogging and e-commerce platforms. Sarcasm detection is imperative for accurate sentiment analysis and opinion mining. It could contribute to enhanced automated feedback systems in the context of customer-based sites. Twitter is a micro-blogging platform extensively used by people to express thoughts, reviews, discussions on current events and convey information in the form of short texts. The relevant context of the tweets are often specified with the use of #(hash-tag). Twitter data provides a diverse corpus for sentences which implicitly contain sarcasm.

We first present a rule-based approach to detect sarcasm expressed due to numbers. Our approach compares numerical magnitudes with those seen in similar contexts in a training dataset. Since 'similar context' is key here, we consider two variants of our approach in order to match the context. Then we present Machine learning based approach and its variant that take different features as input for learning. Further we propose deep learning-based approaches to numerical sarcasm detection on social media that does not require extensive manual feature engineering. We develop Long-short Term Memory (LSTM) network which is able to handle sequences of any length and capture long term dependencies. We compare our approaches with four past works and show an improvement.

Review of Literature

1. Sarcasm Detection on Twitter by Hao Lyu, M.S.INFO.STDS, The University of Texas at Austin, 2016 SUPERVISOR: Byron Wallace : The Data Processing part is done with the help from this paper which states, “The raw tweet and contextualizing data are difficult to use: tweets are written in various languages and length; and each author has a different time zone, which is hard to categorize. I normalize all tweets and process other data. I remove tweets not written in English, retweets, and tweets that contain fewer than three words. URLs and user mentions are replaced. The hashtag #sarcastic and #sarcasm in the Sarcastic tweets are also removed. For the profile data, like time zone, I use Google geocoder package to map different locations to a similar area. Numbers in Twitter are displayed in string, e.g., “22K” or “2 Million”, and they need to be converted to numeric type. All the data are stored into a database and the tweet IDs are used as the unique key for each tweet”.

2. “Having 2 hours to write a paper is fun!”: Detecting Sarcasm in Numerical Portions of Text Lakshya Kumar, Arpan Somani, Pushpak Bhattacharyya Dept. of Computer Science and Engineering IIT Bombay, India lakshya,somani,pb@cse.iitb.ac.in : From this paper we have looked up to the portion in which the neural networks are trained with the LSTM-FF model. We have applied this model to for both training the dataset as well as on the streaming twitter data to obtain the required results.

Objective of the Project

Our primary goal is to analyze weather a tweet is sarcastic or not.

Computational detection of sarcasm has seen attention from the sentiment analysis community in the past few years. Sarcasm is an interesting problem for sentiment analysis because surface sentiment of words in a sarcastic text may be different from the implied sentiment. For example, 'Being stranded in traffic is the best way to start a week' is a sarcastic sentence because the surface sentiment of the word 'best' (positive) is different from the implied sentiment of the sentence (negative), considering remaining portions of the text.

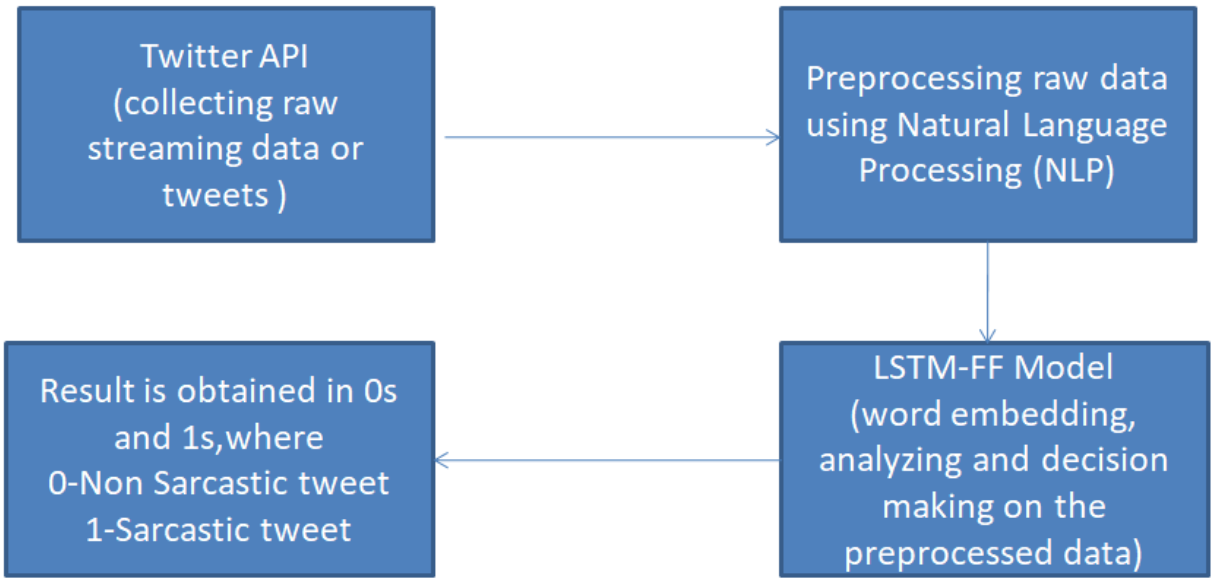
We have proposed an algorithm to understand a sentence or tweet is sarcastic or not. From the dataset we have acquired we are trying to train our model so that test live streaming tweets from Twitter API to detect weather they are sarcastic or not. This will result in obtaining accurate sentiment analysis and opinion mining. It could contribute to enhanced automated feedback systems in the context of customer-based sites.

System Design

System Requirements:
Hardware Requirement:

- 64-bit CPU that supports 64bit virtualization - and of course 64-bit OS is needed
- At least 8GB RAM required (16GB recommended)
- Linux installed.
- Anaconda installed.
- Spyder installed.
- There should have updated version of Python (at least 3.0). Li-braries like Pandas, Keras, Numpy, Tensorflow, Scikit learn is required.

System Architecture:



Methodology for Implementation (Formulation & Algorithm**)**

Machine Learning: Machine learning is a subset of artificial intelligence that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs, those can teach themselves to grow and change when exposed to new data.

We consider the prediction problem as a problem of supervised learning problem, where we have to infer from historical data the possibly nonlinear dependence between the input (past embedding vector) and the output (future value).

Text Preprocessing: NLP is a branch of data science that consists of systematic processes for analyzing, understanding, and deriving information from the text data in a smart and efficient manner. By utilizing NLP and its components, one can organize the massive chunks of text data, perform numerous automated tasks and solve a wide range of problems such as – automatic summarization, machine translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation etc.

Here we have used RE package of python to apply NPL in order to do the following tasks:

1. Removal of new lines and tabs.
2. Removal of punctuations.
3. Separating hash tagged words.
4. Tokenizing the inputs.
5. Replacing emoticons with appropriate texts.

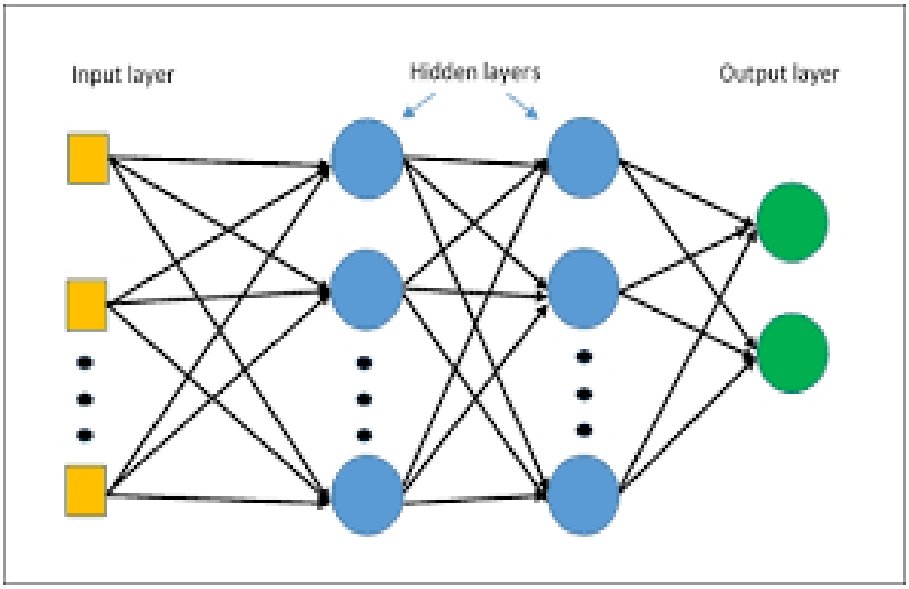
Neural network: For this prediction task we are using *Backpropagation Neural Network*. A neural Network is a machine learning algorithm to perform *Classification* and *Regression* related task.
We are using *logistic sigmoid* as activation function for the neural network. A logistic sigmoid function always outputs between 0 and 1 (both inclusive).

$$F(x) = 1 / (1 + exp(-x))$$

The derivative of sigmoid function $F'(x) = F(x)(1 – F(x))$

Feedforward:- The inputs from the input layer to the hidden layer are multiplied with the respective weights and then the each hidden node sums up all the inputs it is getting. Then the value is passed through the activation function and again the values from hidden layer to output layer are multiplied by respective weights and the output sums up the input it is receiving, then it passes the sum through the activation again and produces output.

Backpropagation:- The output from the output layer is then compared with the target output. Our goal with backpropagation is to update each of the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole.



Word Embedding: In very simplistic terms, Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text. As it turns out, many Machine Learning algorithms and almost all Deep Learning Architectures are incapable of processing *strings* or *plain text* in their raw form. They require numbers as inputs to perform any sort of job, be it classification, regression etc. in broad terms. And with the huge amount of data that is present in the text format, it is imperative to extract knowledge out of it and build applications. Some real-world applications of text applications are – sentiment analysis of reviews by Amazon etc., document or news classification or clustering by Google etc.

Let us now define Word Embeddings formally. A Word Embedding format generally tries to map a word using a dictionary to a vector.

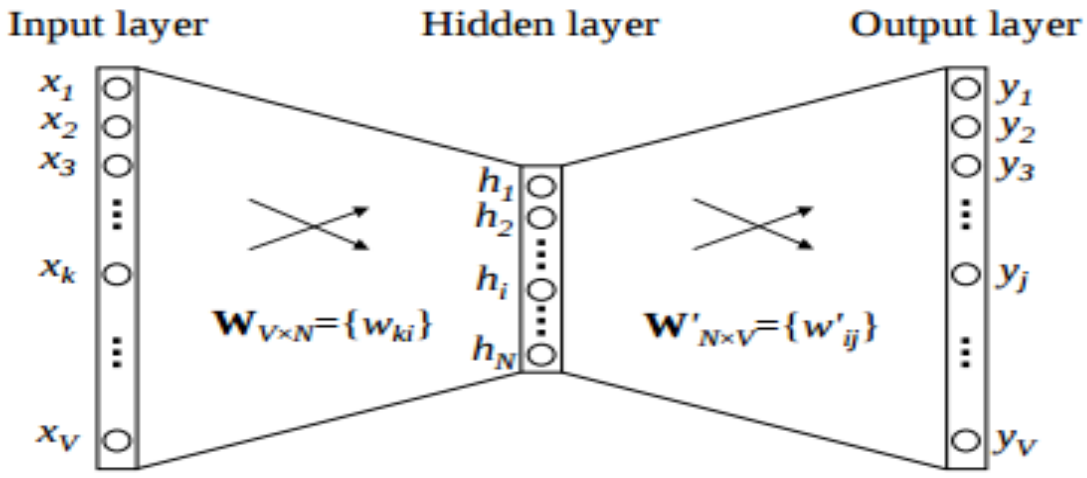
The different types of word embeddings can be broadly classified into two categories-

1. Frequency based Embedding
2. Prediction based Embedding

We have used Prediction based Embedding to map a word using a dictionary to a vector. One of the prediction-based techniques is the CBOW technique which we have used to create our word vector.

Continuous Bag of words(CBOW): The way CBOW work is that it tends to predict the probability of a word given a context. A context may be a single word or a group of words.

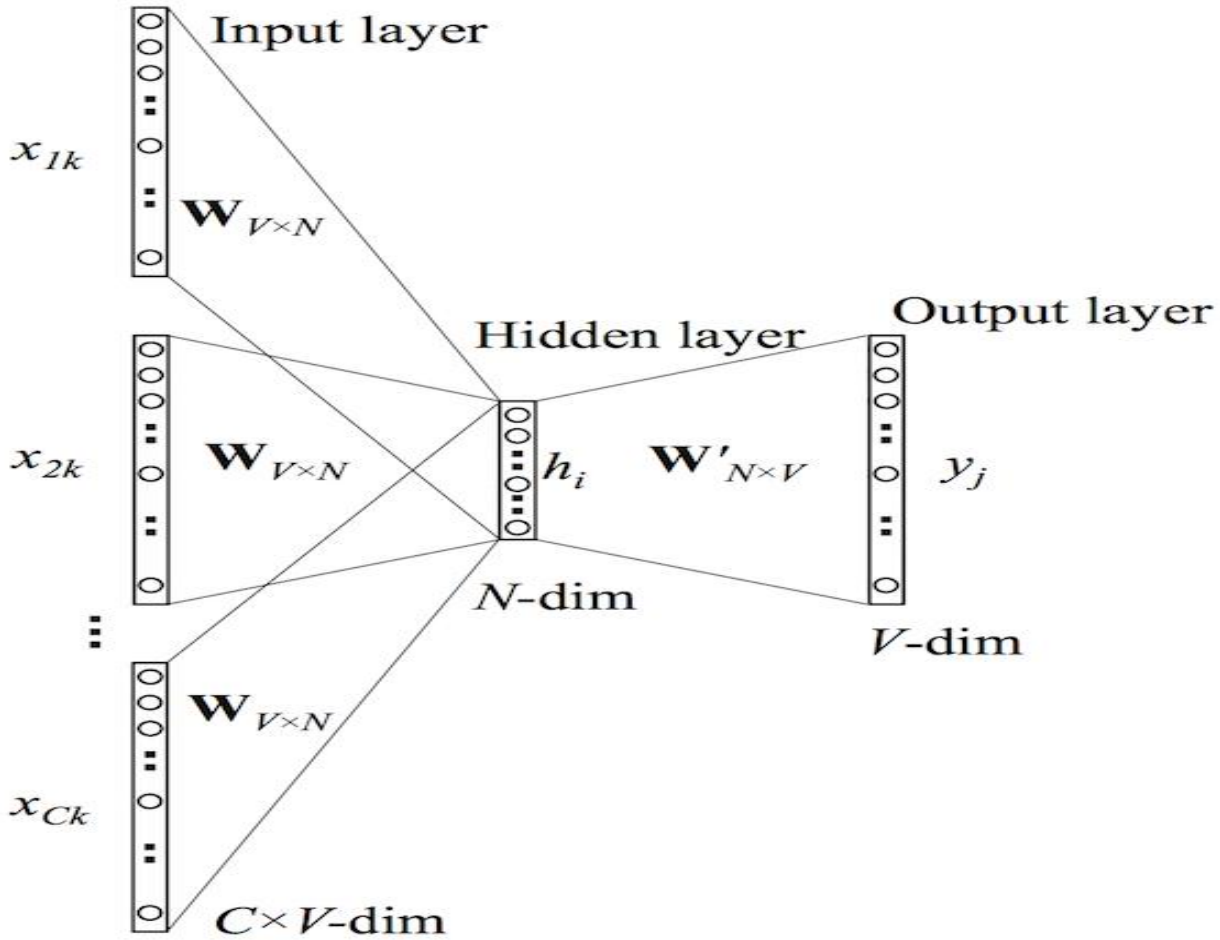
Let us first see a diagrammatic representation of the CBOW model.



The flow is as follows:

1. The input layer and the target, both are one- hot encoded of size [1 X V]. Here V=10 in the above example.
2. There are two sets of weights. one is between the input and the hidden layer and second between hidden and output layer.
   Input-Hidden layer matrix size = [V X N], hidden-Output layer matrix size = [N X V]: Where N is the number of dimensions we choose to represent our word in. It is arbitrary and a hyper-parameter for a Neural Network. Also, N is the number of neurons in the hidden layer. Here, N=4.
3. There is a no activation function between any layers. (More specifically, I am referring to linear activation)
4. The input is multiplied by the input-hidden weights and called hidden activation. It is simply the corresponding row in the input-hidden matrix copied.
5. The hidden input gets multiplied by hidden- output weights and output are calculated.
6. Error between output and target is calculated and propagated back to re-adjust the weights.
7. The weight between the hidden layer and the output layer is taken as the word vector representation of the word.

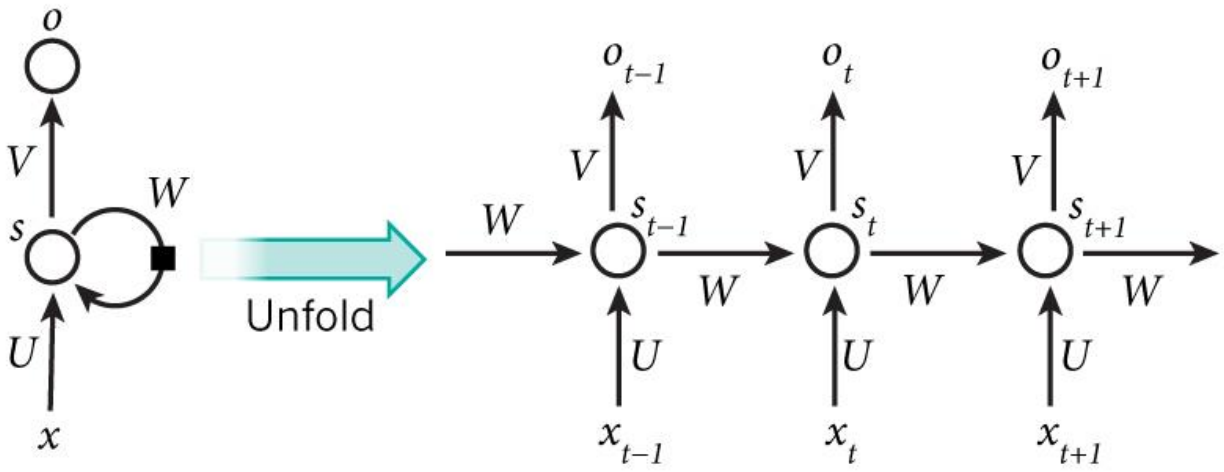
For multiple context words, the image of the architecture is as follows:



*Recurring Neural Network(RNN Model)**:*** A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled.

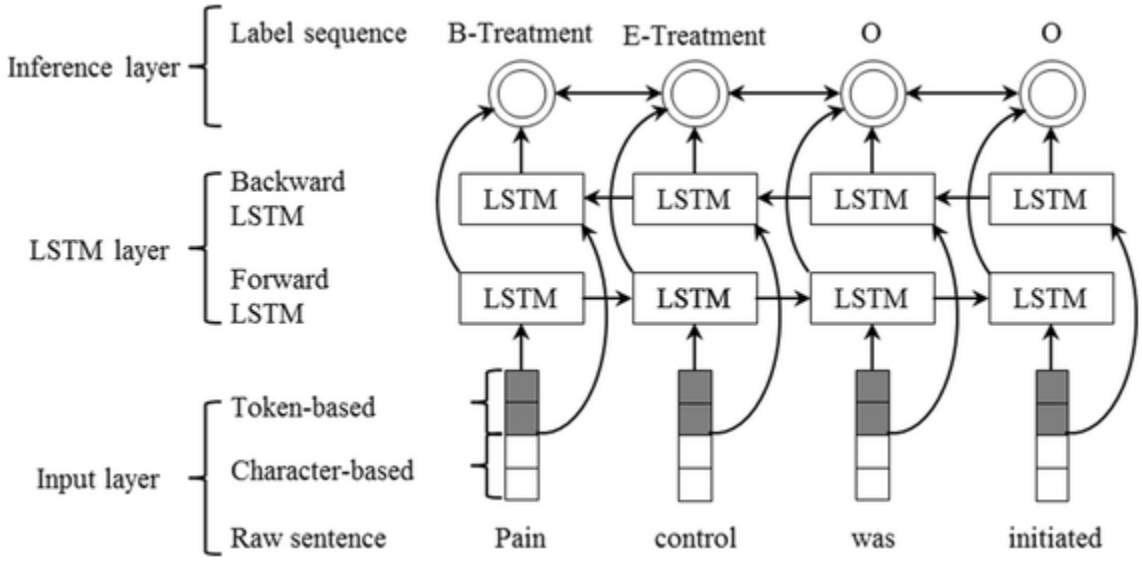
Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of Long short-term memories (LSTM) and gated recurrent units.



Long Short-Term Memory(LSTM): Long short-term memory (LSTM) units (or blocks) are a building unit for layers of a recurrent neural network (RNN). A RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer (or feedforward) neural network: that is, they compute an activation (using an activation function) of a weighted sum. Intuitively, they can be thought as *regulators* of the flow of values that goes through the connections of the LSTM; hence the denotation "gate". There are connections between these gates and the cell.

The expression *long short-term* refers to the fact that LSTM is a model for the short-term memory which can last for a *long* period of time. An LSTM is well-suited to classify, process and predict time series given time lags of unknown size and duration between important events. LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs. Relative insensitivity to gap length gives an

advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods in numerous applications.



LSTM For Sequence Classification with Dropout:

Recurrent Neural networks like LSTM generally have the problem of overfitting. Dropout can be applied between layers using the Dropout Keras layer. We can do this easily by adding new Dropout layers between the Embedding and LSTM layers and the LSTM and Dense output layers. We can see dropout having the desired impact on training with a slightly slower trend in convergence and in this case a lower final accuracy. The model could probably use a few more epochs of training and may achieve a higher skill (try it a see). Alternately, dropout can be applied to the input and recurrent connections of the memory units with the LSTM precisely and separately. Keras provides this capability with parameters on the LSTM layer, the dropout for configuring the input dropout and recurrent dropout for configuring the recurrent dropout.

LSTM-FF Model: RNN have demonstrated the power to capture sequential information in a chain-like neural network. Standard RNNs becomes unable to learn long-term dependencies as the gap between two-time steps becomes large. We adopted the standard architecture of LSTM proposed by Hochreiter and Schmidhuber. The LSTM architecture has a range of repeated modules for each time step as in a standard RNN. At each time step, the output of the module is controlled by a set of gates in IR^d as a function of the old hidden state h_{t-1} and the input at the current time step x_t: the forget gate f_t, the input gate it, and the output gate o_t. These gates collectively decide how to update the current memory cell c_t
and the current hidden state h_t. We use d to denote the memory dimension in the LSTM and all vectors in this architecture share the same dimension.
The LSTM transition functions are defined as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

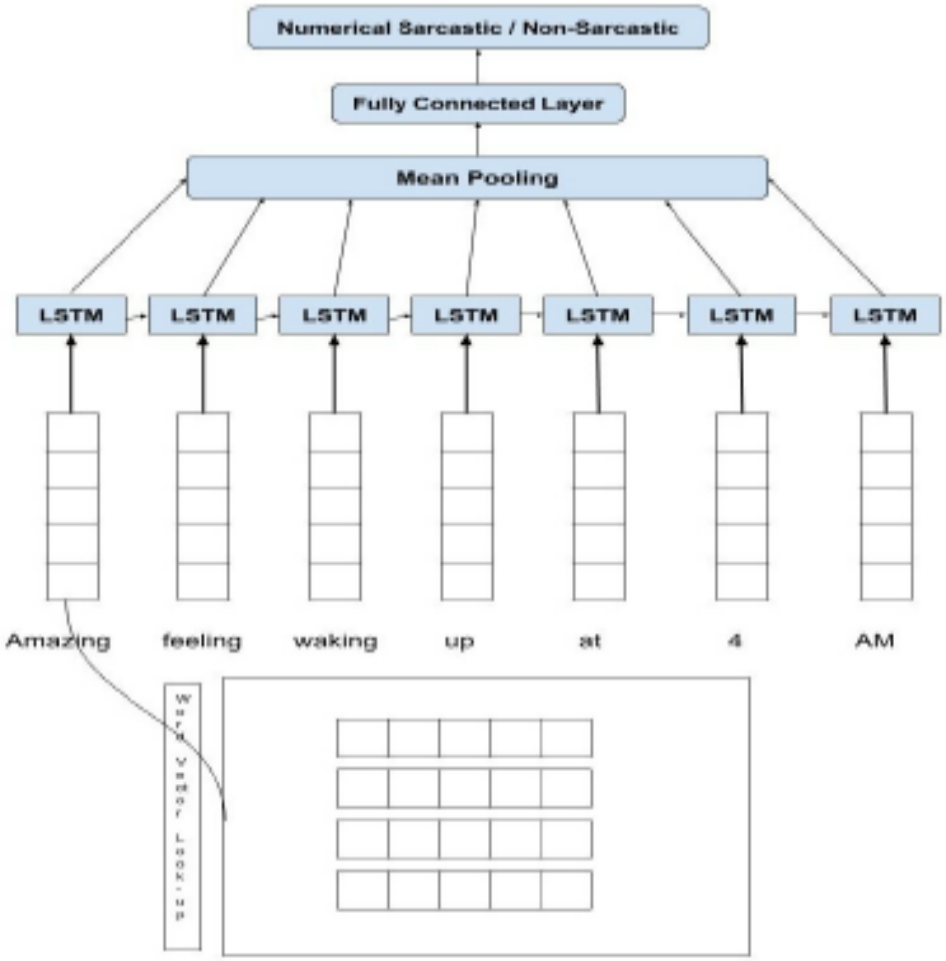
$$\widetilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C_t}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(C_t)$$

This architecture is shown in the figure below. In order to convert the input tweet T into its matrix representation I, embedding matrix E is used. This input matrix is given as input to LSTM cell one word at a time. The output from each time step is stored, on which mean-pooling operation is performed to get the final feature vector of the tweet. This feature vector is passed to the fully connected layer and model is trained by minimizing binary cross-entropy error.

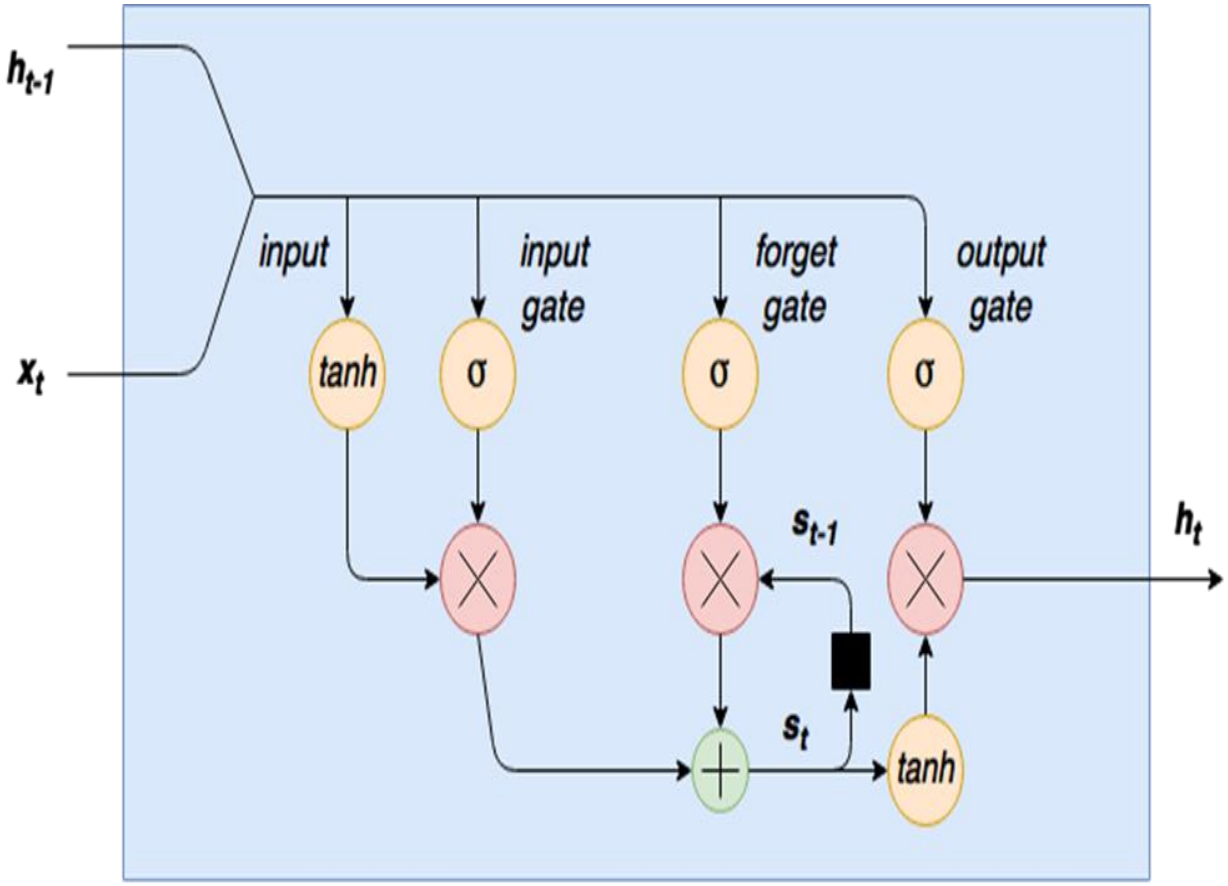


There is an embedding matrix E ∈ IR|V |×d where |V | is the vocabulary size and d is the tweet word embedding dimension. For the input tweet we obtain an input matrix I ∈ IR|S|×d where |S| is the length of the tweet including padding, where Ii be the d-dimension vector for i-th word in the tweet in the input matrix. Let k be the length of the filter, and the vector f ∈ IR|k|×d is a filter for the convolution operation. For each position p in the input matrix I, there is a window wp of k consecutive words, denoted as:

$$w_p = [I_p, I_{p+1}, ..., I_{p+k-1}]$$

A filter f convolves with the window vectors (k-grams) at each position in a valid way to generate a feature map c ∈ IR|S|−k+1 each element c_p of the feature map for window vector w_p is produced as follows:

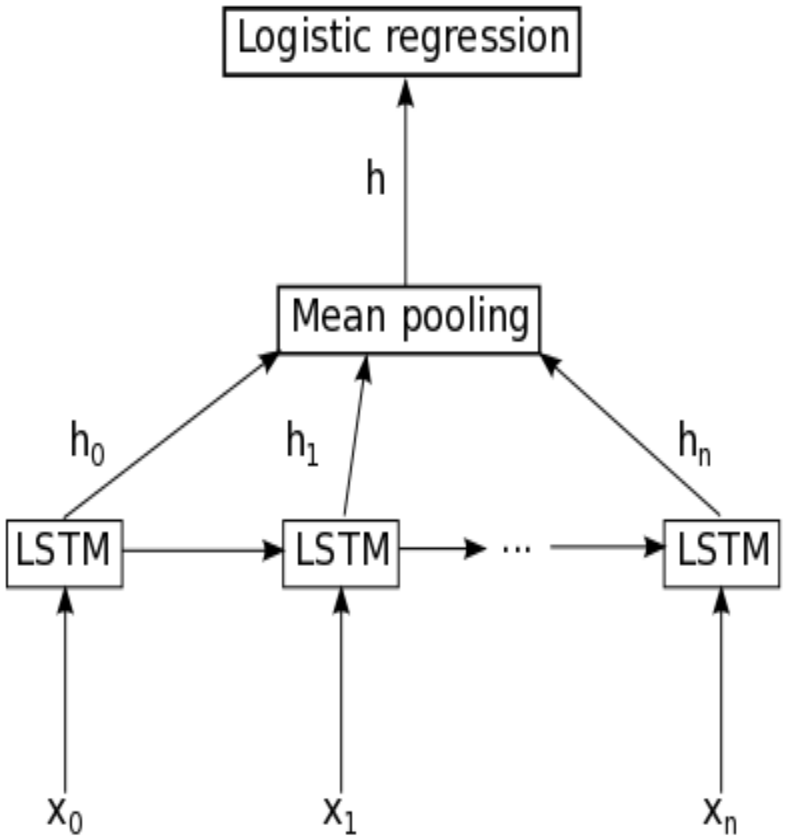
$$c_p = func(w_p \circ f + b)$$



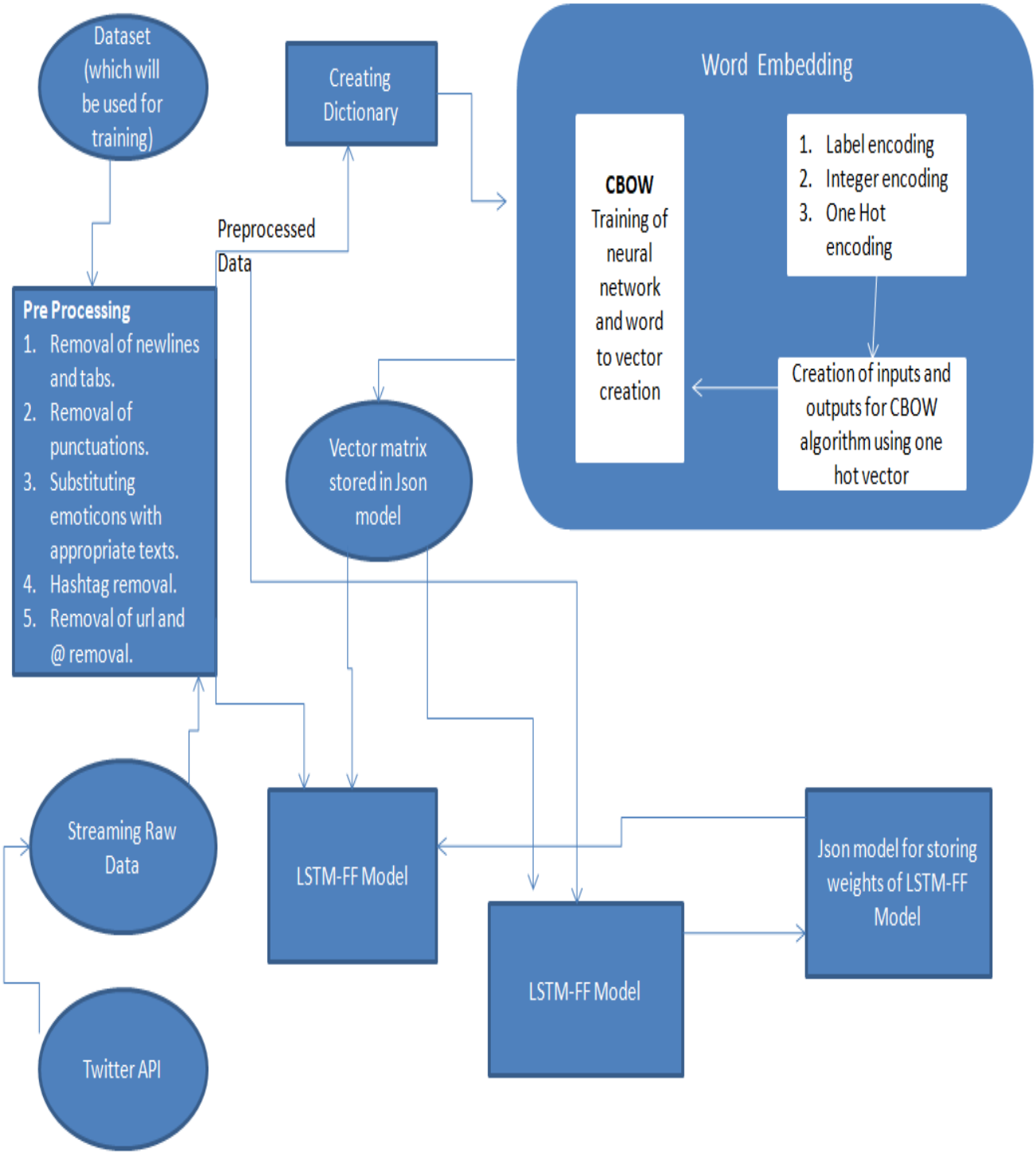
Mean Pooling: A mean-pool layer compresses by taking the mean activation in a block. If large activations are balanced by negative activations, the overall compressed activations will look like no activation at all.

Imagine learning to recognize an 'A' vs 'B' (no variation in A's and in B's pixels). First in a fixed position in the image. This can be done by a logistic regression (1 neuron): the weights end up being a template of the difference A - B.

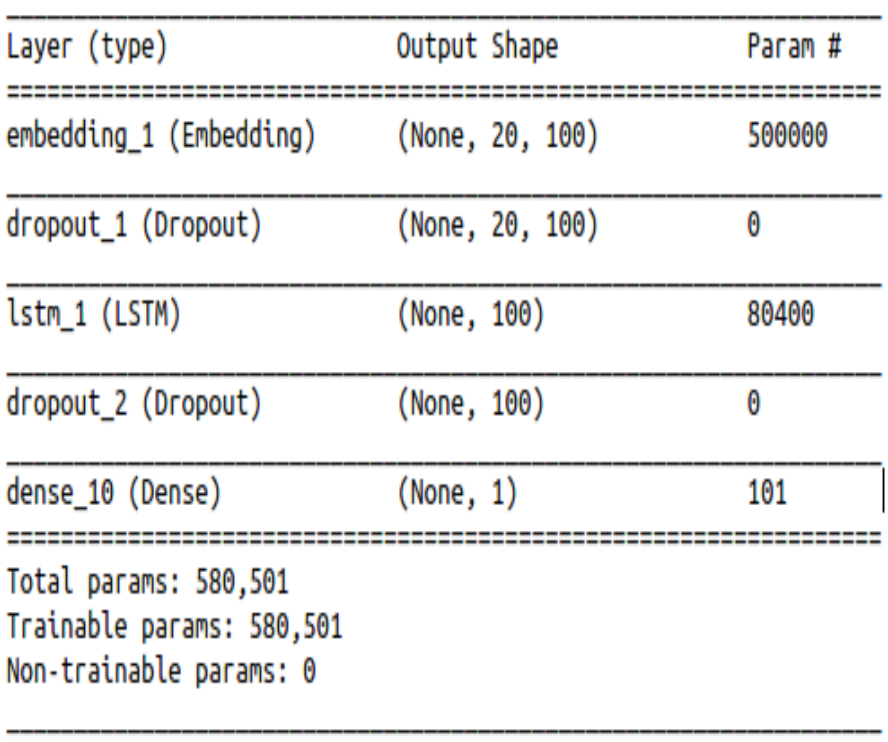
Now what happens if you train to recognize on different locations in the image. You cannot do this with logistic regression, sweeping over the image (i.e. approximating a convolutional layer with one filter) and labelling all sweeps of the image A or B as appropriate, because learning from the different positions interferes - effectively you try to learn the average of A-B as A/B are passed across your filter - but this is just a blur.



Implementation Details:



Model Layout:



Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 20, 100)	500000
dropout_1 (Dropout)	(None, 20, 100)	0
lstm_1 (LSTM)	(None, 100)	80400
dropout_2 (Dropout)	(None, 100)	0
dense_10 (Dense)	(None, 1)	101

Total params: 580,501
Trainable params: 580,501
Non-trainable params: 0

Dataset:

Training Dataset: The datasets we have used for training, is being downloaded by us from GitHub. It contains 40 k tweets for embedding and training. The link is given below:

1. https://raw.githubusercontent.com/AniSkywalker/SarcasmDetection/master/resource/train/Train_v1.txt
2. https://github.com/AniSkywalker/SarcasmDetection/blob/master/resource/emoji_unicode_names_final.txt
3. https://github.com/AniSkywalker/SarcasmDetection/blob/master/resource/abbreviations.txt

Screenshot of the training datasets are as follows:

TrainSen	0	@0430yes i hope youre lurking rn. i want to listen to hallucination & wanna love you again live someday, pretty please?! 😭😭😭
TrainSen	0	05 really taught me a valuable lesson I'm never gonna be late again! #Not
TrainSen	0	@098BERRY Never had a voice to protest, so you fed me shit to digest. I wish I had a reason, my flaws are open season.
TrainSen	0	@0hMySt4rs Rest in peace & love to you and your family
TrainSen	0	100 days until Christmas! 🎄 #too soon #not ready yet
TrainSen	0	@100_ThingsILove @WhatNinaSpotted yay! Can't wait to be reunited with you huni! Xx
TrainSen	0	100 words short of the word requirement but I don't care, I'm going to bed. #rebel #ihatehistory
TrainSen	0	@1010xlhacker it was nice hanging out this afternoon. I had a great time!
TrainSen	0	10k walk this morning. We did an awesome job.
TrainSen	0	10 minutes to eat #challengeaccepted
TrainSen	0	10 Times One Direction Had the Perfect Love Song for Every Relationship :
TrainSen	0	#10TurnOns Yet again if you list them on twitter you're garbage.
TrainSen	0	@11aawwgg021 wow! sweet butt for all of them i love :)
TrainSen	0	11 o'clock cant come no faster
TrainSen	0	11pm on a Saturday night and Im in my room feeling bad for the lady who didn't get bail yesterday
TrainSen	0	12:30 can't come any faster
TrainSen	0	$13, 180 invested in our mission on #GiveMiamiDay! Thank you! So close to our 15k goal. Help us get there!
TrainSen	0	15-year-old boy fighting for his life and another injured after stabbing at London school
TrainSen	0	17 years ago today we made one of the best choices of our lives. Here " s to 17 years of being smoke free!
TrainSen	0	180 days until summer! #notthatbad on the bright side we only have 2 days until a 4 day weekend
TrainSen	0	1966 was when In Cold Blood scared us, Bruno Sammartino slammed us & John Lennon shocked us :
TrainSen	0	19 Million Project brings journalists, coders and humanitarians together around the migrant crisis
TrainSen	0	1 am and I can already tell today will be a great day
TrainSen	0	@1dChicas_ @kingjennny first reaction to the pants I yelled WHAT ARE THOSE PANTS! But I fucking love those pants. They're so Harry
TrainSen	0	1D family! We want to see all of the fan art you've made over the years, show us using #1DUltimateArt!
TrainSen	0	1D give boring performances it's always the same #MTVStars One Direction
TrainSen	0	#1DonJonathanRoss i have never laughed so much in an interview 😂 H : His face says " shit i've got chlamydia "
TrainSen	0	#1DonJonathanRoss IVE NEVER LAUGHED SO MUCH IN MY LIFE
TrainSen	0	1d the kings of never promoting yet keeping their winning streak
TrainSen	0	1) Good Grades 2) Social Life 3) Excel in your sport 4) Sleep You may only choose 2... Good luck
TrainSen	0	@1kunalbahl Can you please reply
TrainSen	0	1 sleep until #StrictlyAgainstBreastCancer at Dublin's convention centre. Can't wait 2 c all the brave dancers @BreastCancerIre
TrainSen	0	1st i wana thank everyone for the calls, texts, messages, condolences and prayers. My grandmother is very close 2
TrainSen	0	1st World society helped 'create ' terrorists, b/c they r JEALOUS & thus hate us. Send em 2 TEXAS 4 Target Practice.
TrainSen	0	@1truseatlefan KJR is awful is why. Why they get killed in ratings. No expert analysis. Just throw shit against the wall hoping to be right
TrainSen	0	$2000 dollar credit limit on my new credit card #Perfect?
TrainSen	0	2001, #ThisDayInMusic Scott Weiland was arrested after allegedly fighting with his wife at the Hard Rock hotel in Las Vegas
TrainSen	0	2013 Henriette was so full of life man
TrainSen	0	2015 been a weird ass year like hella shit has happened and changed within this one year like damn

Screenshot of the dataset used to convert emoticons to respective text values:

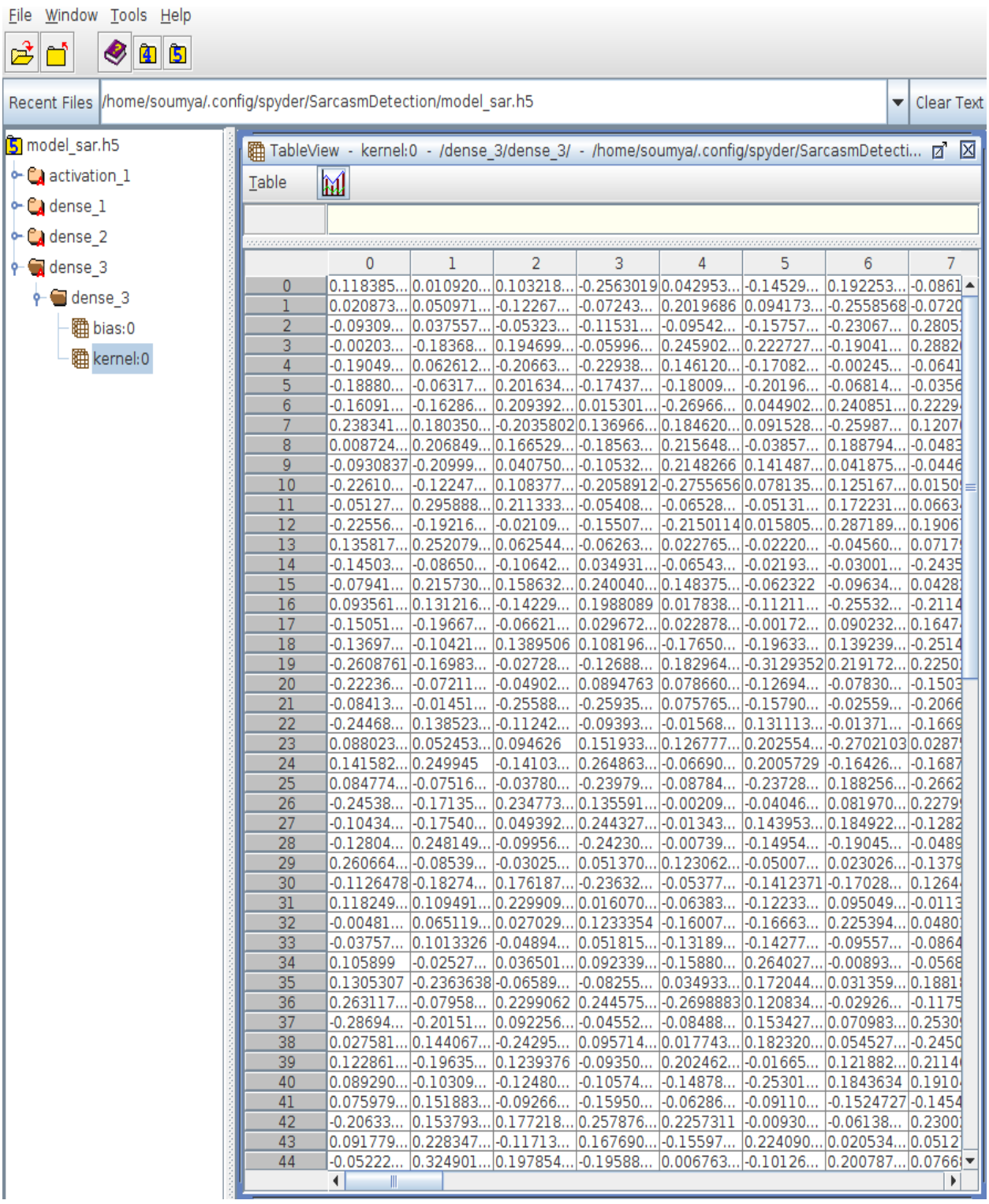
1	©	copyright_sign
2	®	registered_sign
3	‼	double_exclamation_mark
4	⁉	exclamation_question_mark
5	™	trade_mark_sign
6	ℹ	information_source
7	↔	left_right_arrow
8	↕	up_down_arrow
9	↖	north_west_arrow
10	↗	north_east_arrow
11	↘	south_east_arrow
12	↙	south_west_arrow
13	↩	leftwards_arrow_with_hook
14	↪	rightwards_arrow_with_hook
15	⌚	watch
16	⌛	hourglass
17	⌨	keyboard
18	⏏	eject_symbol
19	⏩	black_right_pointing_double_triangle
20	⏪	black_left_pointing_double_triangle
21	⏫	black_up_pointing_double_triangle
22	⏬	black_down_pointing_double_triangle
23	⏭	black_right_pointing_double_triangle_with_vertical_bar
24	⏮	black_left_pointing_double_triangle_with_vertical_bar
25	⏯	black_right_pointing_triangle_with_double_vertical_bar
26	⏰	alarm_clock
27	⏱	stopwatch
28	⏲	timer_clock

Screenshots of the dataset containing slang abbreviations and their respective full forms:

```
i've    i have
we've   we have
can't   can not
i'm     i am
we're   we are
don't   do not
didn't  did not
tt's    it is
that's  that is
he's    he is
she's   she is
let's   let us
there's there is
how's   how is
i'd     i would
2F4U    Too Fast For You
4YEO FYEO       For Your Eyes Only
AAMOF   As a Matter of Fact
ACK     Acknowledgment
AFAIK   As far as I know
AFAIR   As far as I remember
AFK     Away from Keyboard
AKA     Also known as
B2K BTK Back to Keyboard
BTT     Back to Topic
```

Result:

Weight oh hidden layer which represents embedding matrix for the word2vec model. Each column represents vector representation of each word in the dictionary.



	0	1	2	3	4	5	6	7
0	0.118385...	0.010920...	0.103218...	-0.2563019	0.042953...	-0.14529...	0.192253...	-0.0861
1	0.020873...	0.050971...	-0.12267...	-0.07243...	0.2019686	0.094173...	-0.2558568	-0.0720
2	-0.09309...	0.037557...	-0.05323...	-0.11531...	-0.09542...	-0.15757...	-0.23067...	0.2805
3	-0.00203...	-0.18368...	0.194699...	-0.05996...	0.245902...	0.222727...	-0.19041...	0.2882
4	-0.19049...	0.062612...	-0.20663...	-0.22938...	0.146120...	-0.17082...	-0.00245...	-0.0641
5	-0.18880...	-0.06317...	0.201634...	-0.17437...	-0.18009...	-0.20196...	-0.06814...	-0.0356
6	-0.16091...	-0.16286...	0.209392...	0.015301...	-0.26966...	0.044902...	0.240851...	0.2229
7	0.238341...	0.180350...	-0.2035802	0.136966...	0.184620...	0.091528...	-0.25987...	0.1207
8	0.008724...	0.206849...	0.166529...	-0.18563...	0.215648...	-0.03857...	0.188794...	-0.0483
9	-0.0930837	-0.20999...	0.040750...	-0.10532...	0.2148266	0.141487...	0.041875...	-0.0446
10	-0.22610...	-0.12247...	0.108377...	-0.2058912	-0.2755656	0.078135...	0.125167...	0.0150
11	-0.05127...	0.295888...	0.211333...	-0.05408...	-0.06528...	-0.05131...	0.172231...	0.0663
12	-0.22556...	-0.19216...	-0.02109...	-0.15507...	-0.2150114	0.015805...	0.287189...	0.1906
13	0.135817...	0.252079...	0.062544...	-0.06263...	0.022765...	-0.02220...	-0.04560...	0.0717
14	-0.14503...	-0.08650...	-0.10642...	0.034931...	-0.06543...	-0.02193...	-0.03001...	-0.2435
15	-0.07941...	0.215730...	0.158632...	0.240040...	0.148375...	-0.062322	-0.09634...	0.0428
16	0.093561...	0.131216...	-0.14229...	0.1988089	0.017838...	-0.11211...	-0.25532...	-0.2114
17	-0.15051...	-0.19667...	-0.06621...	0.029672...	0.022878...	-0.00172...	0.090232...	0.1647
18	-0.13697...	-0.10421...	0.1389506	0.108196...	-0.17650...	-0.19633...	0.139239...	-0.2514
19	-0.2608761	-0.16983...	-0.02728...	-0.12688...	0.182964...	-0.3129352	0.219172...	0.2250
20	-0.22236...	-0.07211...	-0.04902...	0.0894763	0.078660...	-0.12694...	-0.07830...	-0.1503
21	-0.08413...	-0.01451...	-0.25588...	-0.25935...	0.075765...	-0.15790...	-0.02559...	-0.2066
22	-0.24468...	0.138523...	-0.11242...	-0.09393...	-0.01568...	0.131113...	-0.01371...	-0.1669
23	0.088023...	0.052453...	0.094626	0.151933...	0.126777...	0.202554...	-0.2702103	0.0287
24	0.141582...	0.249945	-0.14103...	0.264863...	-0.06690...	0.2005729	-0.16426...	-0.1687
25	0.084774...	-0.07516...	-0.03780...	-0.23979...	-0.08784...	-0.23728...	0.188256...	-0.2662
26	-0.24538...	-0.17135...	0.234773...	0.135591...	-0.00209...	-0.04046...	0.081970...	0.2279
27	-0.10434...	-0.17540...	0.049392...	0.244327...	-0.01343...	0.143953...	0.184922...	-0.1282
28	-0.12804...	0.248149...	-0.09956...	-0.24230...	-0.00739...	-0.14954...	-0.19045...	-0.0489
29	0.260664...	-0.08539...	-0.03025...	0.051370...	0.123062...	-0.05007...	0.023026...	-0.1379
30	-0.1126478	-0.18274...	0.176187...	-0.23632...	-0.05377...	-0.1412371	-0.17028...	0.1264
31	0.118249...	0.109491...	0.229909...	0.016070...	-0.06383...	-0.12233...	0.095049...	-0.0113
32	-0.00481...	0.065119...	0.027029...	0.1233354	-0.16007...	-0.16663...	0.225394...	0.0480
33	-0.03757...	0.1013326	-0.04894...	0.051815...	-0.13189...	-0.14277...	-0.09557...	-0.0864
34	0.105899	-0.02527...	0.036501...	0.092339...	-0.15880...	0.264027...	-0.00893...	-0.0568
35	0.1305307	-0.2363638	-0.06589...	-0.08255...	0.034933...	0.172044...	0.031359...	0.1881
36	0.263117...	-0.07958...	0.2299062	0.244575...	-0.2698883	0.120834...	-0.02926...	-0.1175
37	-0.28694...	-0.20151...	0.092256...	-0.04552...	-0.08488...	0.153427...	0.070983...	0.2530
38	0.027581...	0.144067...	-0.24295...	0.095714...	0.017743...	0.182320...	0.054527...	-0.2450
39	0.122861...	-0.19635...	0.1239376	-0.09350...	0.202462...	-0.01665...	0.121882...	0.2114
40	0.089290...	-0.10309...	-0.12480...	-0.10574...	-0.14878...	-0.25301...	0.1843634	0.1910
41	0.075979...	0.151883...	-0.09266...	-0.15950...	-0.06286...	-0.09110...	-0.1524727	-0.1454
42	-0.20633...	0.153793...	0.177218...	0.257876...	0.2257311	-0.00930...	-0.06138...	0.2300
43	0.091779...	0.228347...	-0.11713...	0.167690...	-0.15597...	0.224090...	0.020534...	0.0512
44	-0.05222...	0.324901...	0.197854...	-0.19588...	0.006763...	-0.10126...	0.200787...	0.0766

Conclusion:

Sarcasm detection is a really fascinating subject. It evaluates diverse feature types for sentiment extraction including sentiments, words, patterns and n-grams, confirming that each feature type contributes to the sentiment classification framework. As we have seen that it is feasible to do sarcasm detection using NLP tools, one quick and easy way to improve this detector is to use a spell corrector along with, for the tweets. This would help in minimizing the order of dimensions of the dictionary for the n-gram features and will improve the sentiment analysis operation as well. In the future, these methods can be applied for automated clustering of sentiment types and sentiment dependency rules and can be expanded to detect some other non-literal form of sentiments like humor.

In this project we have used a dataset containing around 40,000 tweets to train our model, we have preprocessed the raw data using NLP. Then creating a dictionary of the words, we have send it for word embedding, where we trained out neural network using CBOW and also created word2vec matrix. We are using a twitter API from where we will be getting streaming twitter data and give our prediction weather a tweet is sarcastic in nature or not. We also used a test dataset to check the accuracy efficiency of our model. The dataset for testing contained around 2000 tweets. On the testing dataset, after running it in our model, we have obtained a resultant accuracy of 75%.

Experiments with both of the manually annotated datasets yield very similar results, even though the sets are completely independent. This shows that the sets are well annotated and reasonably representative for sarcasm detection in tweets.

This is the link of our source code for sarcasm detection of twitters data:

https://drive.google.com/open?id=1APXmy29V-zhzSPI26vUSOiFMm3NYh3wV

References

Keras documentation: www.keras.io

Analytics Vidya: www.analyticsvidhya.com › Deep Learning

“Having 2 hours to write a paper is fun!”: Detecting Sarcasm in Numerical Portions of Text Lakshya Kumar, Arpan Somani, Pushpak Bhattacharyya, Dept. of Computer Science and Engineering IIT Bombay, India, lakshya,somani,pb@cse.iitb.ac.in

Tweet Sarcasm: Mechanism of Sarcasm Detection in Twitter Komalpreet Kaur Bindra #1, Asst Prof Ankita Gupta*2 # Computer Science Department, PEC University of Technology, Chandigarh, India. * PEC University of Technology, Chandigarh, India

A Large Self-Annotated Corpus for Sarcasm Mikhail Khodak, Nikunj Saunshi, Kiran Vodrahalli Computer Science Department, Princeton University 35 Olden St., Princeton, New Jersey 08540 {mkhodak,nsaunshi,knv}@cs.princeton.edu

Johan G. Cyrus M. Ræder Automatic Sarcasm Detection in Twitter Messages Master’s Thesis, Spring 2016 Artificial Intelligence Group Department of Computer and Information Science Faculty of Information Technology, Mathematics and Electrical Engineering

Sarcasm Detection on Twitter by Hao Lyu, B.M.S Report Presented to the Faculty of the Graduate School of the University of Texas at Austin in Partial Fulfillment of the Requirements for the Degree of Master of Science in INFORMATION STUDIES

Are you serious? Rhetorical Questions and Sarcasm in Social Media Dialog Shereen Oraby1, Vrindavan Harrison1, Amita Misra1, Ellen Riloff 2 and Marilyn Walker1 1 University of California, Santa Cruz 2 University of Utah