Extraction Of Tweet Sentiments

Vijayalakshmi S
Division of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Chennai, India
vijayalakshmi.s2021@vitstudent.ac.in

Naresh Kumar J
Division of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Chennai, India
nareshkumar.j2021@vitstudent.ac.in

Venkatalakshmi S
Division of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Chennai, India
venkatalakshmi.2021@vitstudent.ac.in

Dr Shruthi Mishra
Division of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Chennai, India
kalyanidesikan.@vit.ac.in

Abstract – With some many tweets being sent out every second, it's difficult to know whether the sentiment behind a tweet will positively affect a company's or individual's brand or negatively impact profit. In today's world, where decisions and reactions are made and updated in milliseconds, capturing sentiment in language is critical. Which terms, on the other hand, lead to the sentiment description? In this study, we will be extracting the sentiments from the tweets using deep learning technique and the performance metric as Jaccard distance.

I INTRODUCTION

Sentiment Analysis can be defined as the process of analysing text data and categorizing them into Positive, Negative, or Neutral sentiments. Sentiment Analysis is used in many cases like Social Media Monitoring, Customer service, Brand Monitoring, political campaigns, etc. Analysing customer feedback such as social media conversations, product reviews, and survey responses allows companies to understand the customer's emotions better which is becoming more essential to meet their needs.

II LITERATURE REVIEW

It is almost impossible to manually sort thousands of social media conversations, customer reviews, and surveys. So, we have to use either ML/DL to build a model that analyses the text data and performs the required operations. Here, we are using the GRU (*Gated Recurrent Unit*) algorithm since it is fast and easy to compute when compared with LSTM (*Long Short-Term Memory*). The problem I am trying to solve here is part of this Kaggle

competition. In this problem, we are given some text data along with their sentiment(positive/negative/neutral) and we need to find the phrases/words that best supports the sentiment.

III DATASET

The dataset used here is from the Kaggle Tweet Sentiment Extraction. The dataset used in this competition is from phrases from Figure Eight's Data for Everyone platform. It consists of two data files train.csv and test.csv, where there are 27481 rows in training data and 3534 rows in test data.

List of columns in the dataset

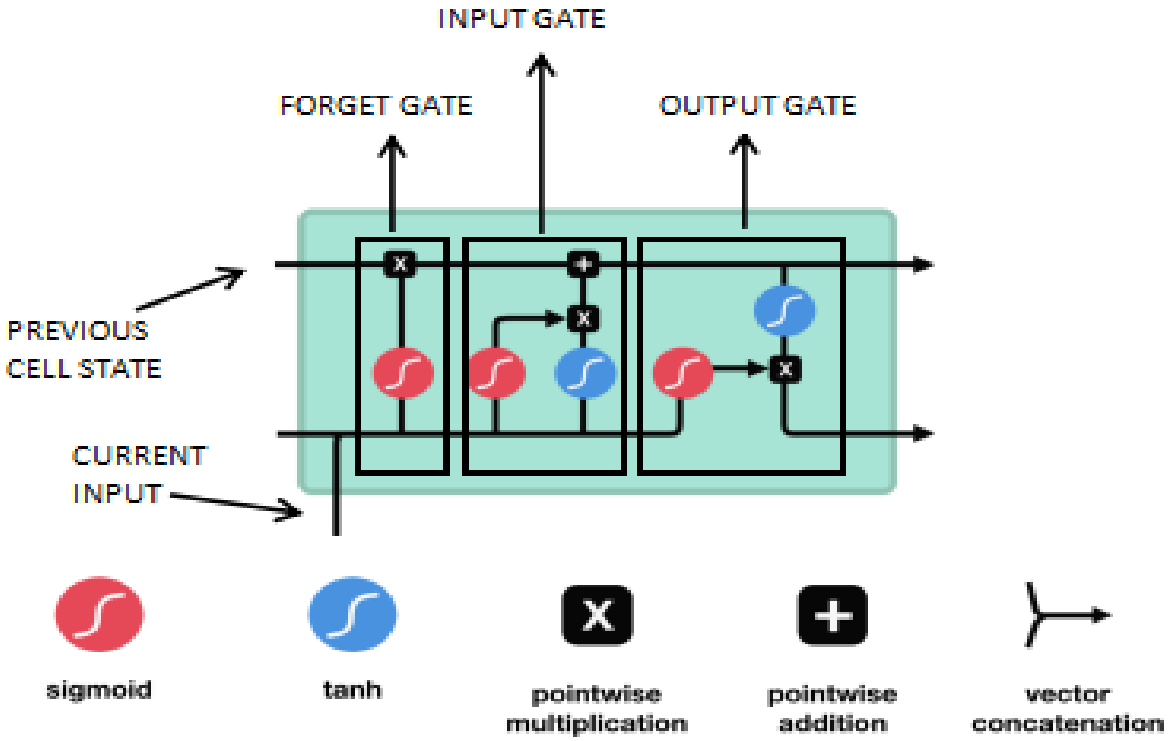
textID: unique id for each row of data

text: this column contains text data of the tweet.

sentiment: the sentiment of the text (positive/negative/neutral)

selected_text: phrases /words from the text that best supports the sentiment

IV METHODOLOGY



Source:

https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2Fanalytics-vidhya

%2Flstm-whats-the-fuss-about-1ae9d4c3e33e&psig=AOvVaw3zJCU9rSx2Mr7j0mnKwAvG&ust=1654590072641000&source=images&cd=vfe&ved=0CAwQjRxqFwoTCIiQndaymPgCFQAAAAAdAAAAABAI

PERFORMANCE METRIC

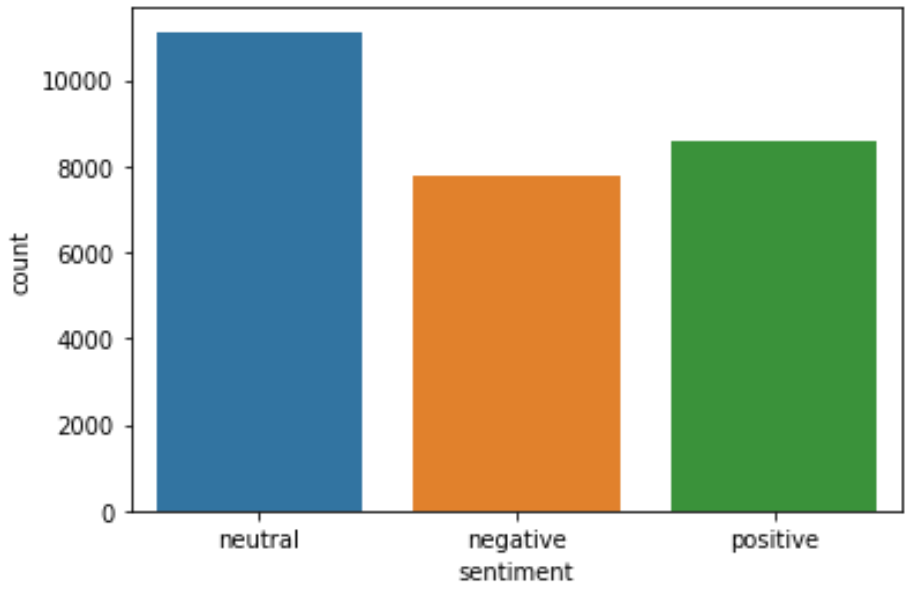
The performance metric used in this problem is the **word-level Jaccard score**. The Jaccard Score or Jaccard Similarity is one of the statistics used in understanding the similarity between two sets.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

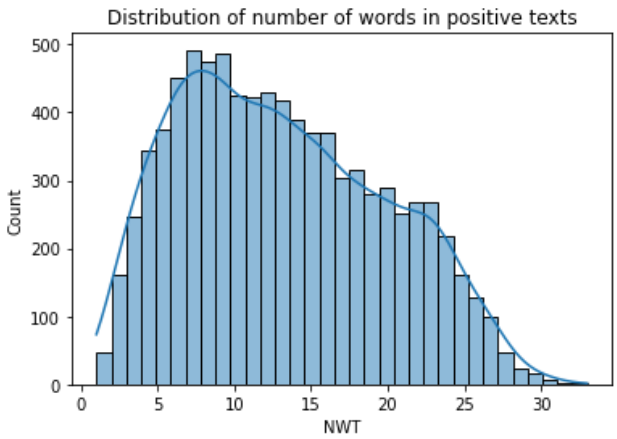
Source: https://en.wikipedia.org/wiki/Jaccard_index

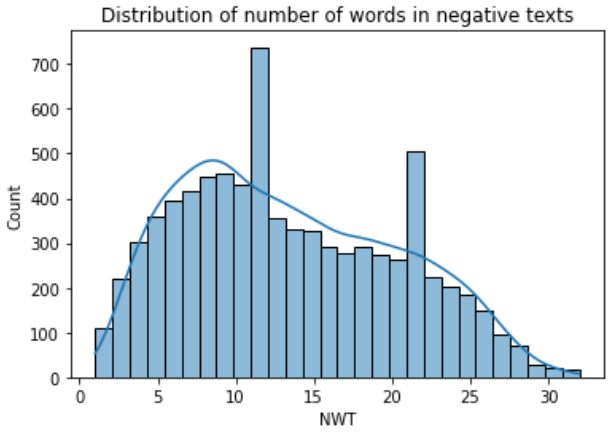
EXPLORATORY DATA ANALYSIS

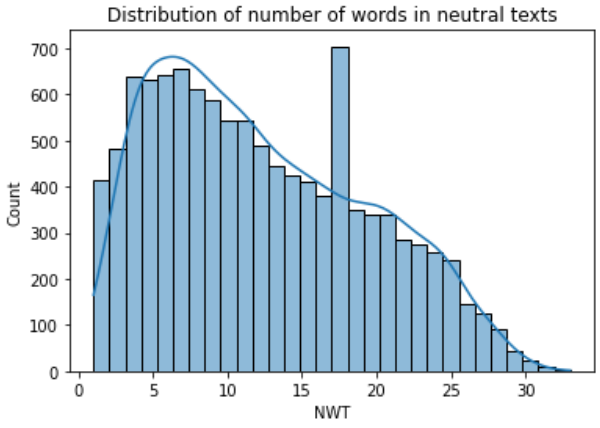
(Inferences of the basic analysis)



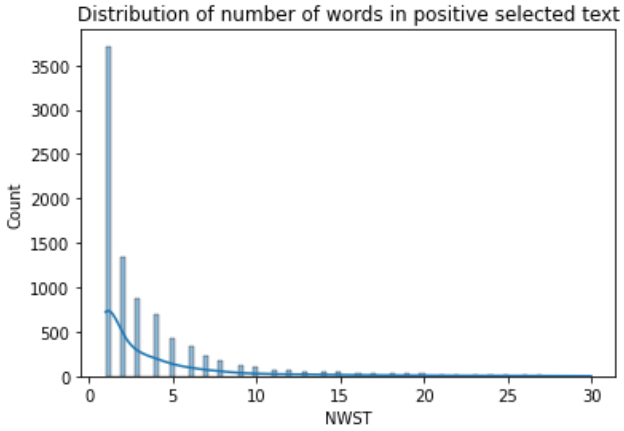
From the above plot the majority of points belong to **Neutral** followed by Positive and Negative texts.

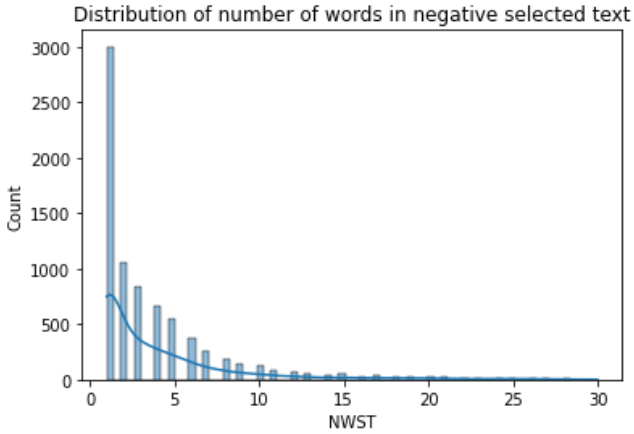


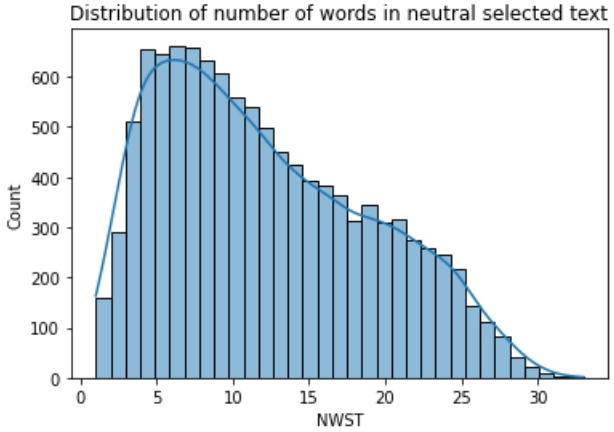




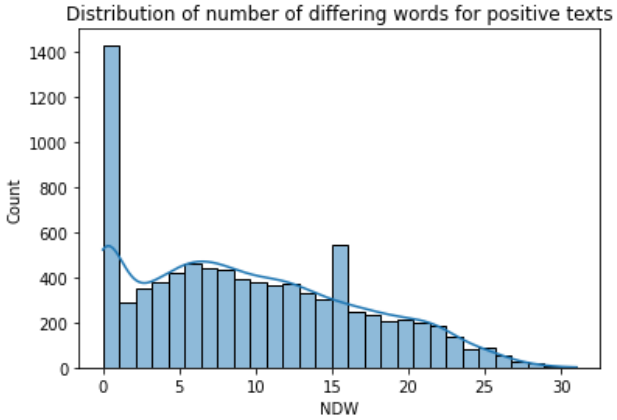
The above can be used to infer the length of the text (no.of words) for all the sentiments are between a count **0–20** and only a few points are greater than 25 words.

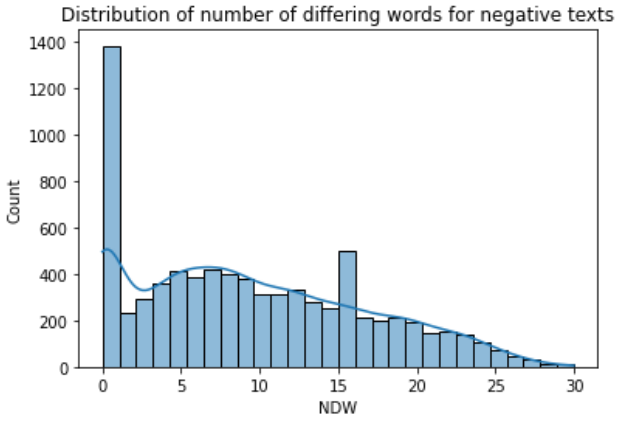


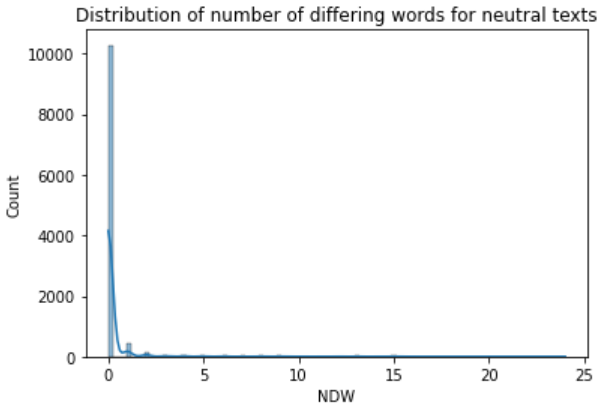




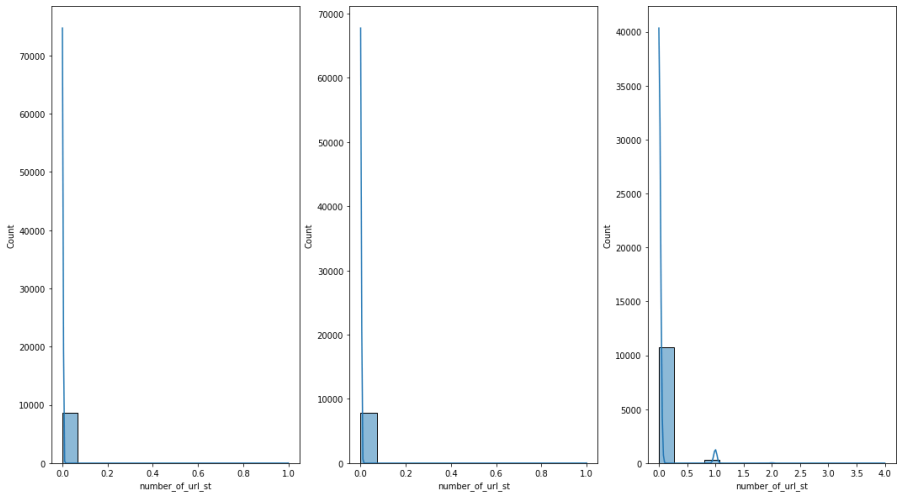
There is a certain difference in no.of words for the **selected_text** column for different sentiments. We could see a **clear spike** in two of the three above plots which suggest that majority of selected_text phrases length lie between **0–10** and only there are a few sentences that are greater than 10 words of length. For **Neutral** sentiment, the **majority of selected_text phrases are longer** when compared to positive/negative sentiment labels as most of the phrases lie between a **length of 0–20.**







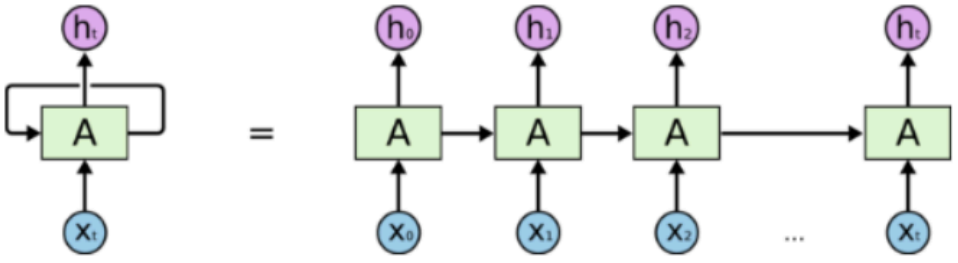
The above plot shows the most common words that are found in the text and selected_text columns for different sentiments.There is a **huge spike** around 1.0 for neutral sentiments, which means that most of the selected_text phrases for the neutral sentiment labels are the text sentences itself, i.e both the text and selected_text values are the **same** for most of the neutral sentiment data. This is considered as the most important analysis as if any data comes with the sentiment neutral, then the whole text can be given as selected text. By this analysis we can just pass positive and negative data into the model for getting selected text, neglecting neutral sentiment texts.



The above plot is used to represent the presence of URL in the selected text. We can see that for both positive and negative sentences there are no URL in the selected text. This is very helpful in the way that URL's can be neglected during preprocessing.

USAGE OF DEEP LEARNING MODEL

One of the most widely used Neural Networks if the input data is sequential or contextual data is **Recurrent Neural Networks (RNN).** One of the major advantages that RNN provides over normal feed-forward neural networks is that RNN doesn't only consider input at the current timestep but it also considers previous values to predict the current output.



An unrolled recurrent neural network.

Source: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

As you can see from the above image, at the first time step, X^0 is passed as input to the model to get H^0. In the next timestep, both X^1 and H^0 are passed as input to the model which gives H^1 as output. Unlike normal Neural networks, all the inputs are related to each other. The

most commonly used RNN networks are **LSTM (Long Short-Term Memory) and GRU(Gated Recurrent Unit).**

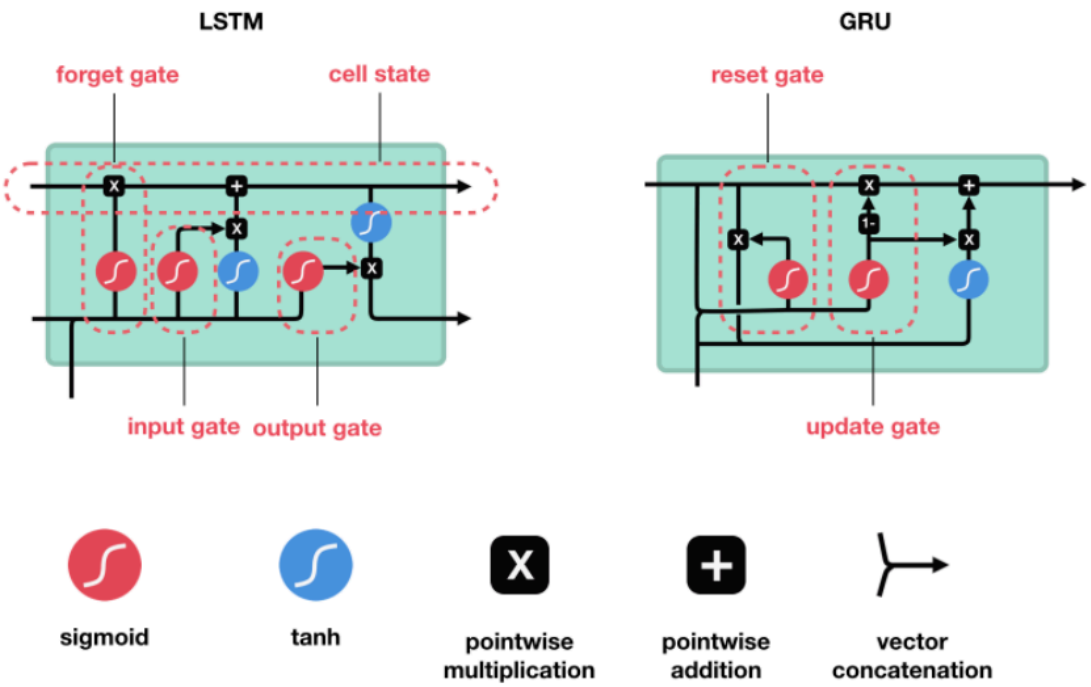


fig3 :LSTM and GRU .image credit — Michael Nguyen

Source: https://towardsdatascience.com/rnn-simplified-a-beginners-guide-cf3ae1a8895b

LSTM

This network has 3 main gates

1.Forget gate

2. Input gate

3. Output gate

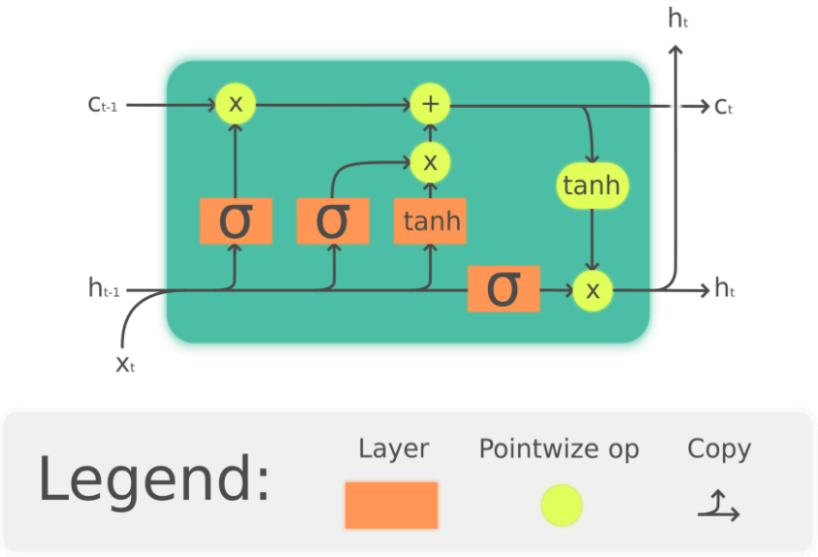


fig4 :Structure of LSTM ,image credit- wikipedia

C_(t-1)**:** old cell state

c_t: current cell state

h_(t-1): output from the previous state

h_t**:** output of the current state

Forget gate decides how much information needs to be retained from previous states and how much can be neglected

Input gate decides which new information will be added to the cell state

Output gate will decide what information will be passed to the network at the next instance

GRU

The major difference between LSTM and GRU is that GRU has only 2 gates **Update gate and Reset gate.**

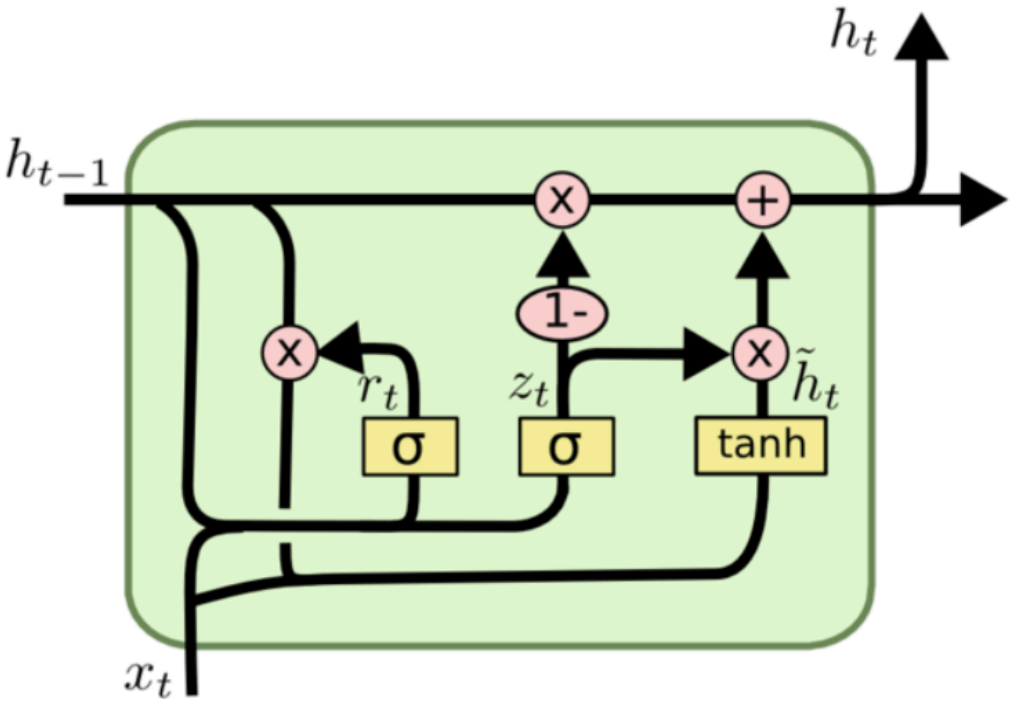


fig5 : Structure of GRU

Update gate is a combination of input gate and forget gate. It decides what information to be retained and what information to be added.
Reset gate decides what information needs to be passed to the network in the next instance.

GRU has less no. of gates when compared to LSTM and hence it's **computationally cheaper and faster** than LSTM.

BASE MODEL

Here we are trying to build a base RNN model using LSTM/GRU which takes the **text** and **sentiment** as input and gives **selected_text** as output.

We have the input in the text format that needs to be converted into numerical data so that we can pass it to the model. Also, we need to perform **Data Cleaning and Preprocessing** steps so that the data is clean and ready for further operations.

Data Preprocessing:

The url links and punctuations in the dataset are removed. Special characters like '****' are replaced by the word or a phrase "curse" so that the Word2Vec can identify that it is cursed word or phrase. This particular Word2Vec is known for the tweets which is named as Glove Vectors and it is also known as *Global Vectors*. This Glove vector is trained especially with twitter data and thereby the word embedding for this word2vec have a btter representation for this objective. Here, the training set and the test set is separated in the ratio 80:20. Hence, the training data contains 13090 rows and the test data contains 3273 rows. The usage of Tokenizer is that values can be assigned to each word in the training data.

Converting the text to integers and padding them to same length:

We have text columns that contain text from various tweets. Each data point would be of different lengths. So, we have to convert this text data into numbers and pad all the points so that all the inputs are of the same length. In TensorFlow, we have **Tokenizer** and **pad sequences** module which can be used to perform these operations.

Embedding Layer:

The embedding layer consists of an embedding matrix which is a matrix containing a high-dimensional representation of a particular word that's is present in the training data Usually, we use pretrained embedding vectors (that is Glove vectors) as these are vectors obtained by training a large amount of data and can be downloaded to use them in this embedding layer. Each word is embedded into a 200 dimensional vector.

Sentiment Embedding:

Along with the text data the sentiments are alos embedded. Here there are only two values for this feature, that is positive and negative and these two are given 200 dimensional embedding so as to give even more better representation for sentiments also.

Output:

Here the output are the text or a sequence of text from the original text. The output is designed in such a way that the model outputs two values. The first value represents the start

index which corresponds of the starting word for the selected text from the original text. The second output represents the end index which corresponds to the last word of the selected text from the original text. The words in between these index values in the original text will be taken as selected text. Here the loss will be Mean squared Error.

Example:

text: What interview! Leave me alone

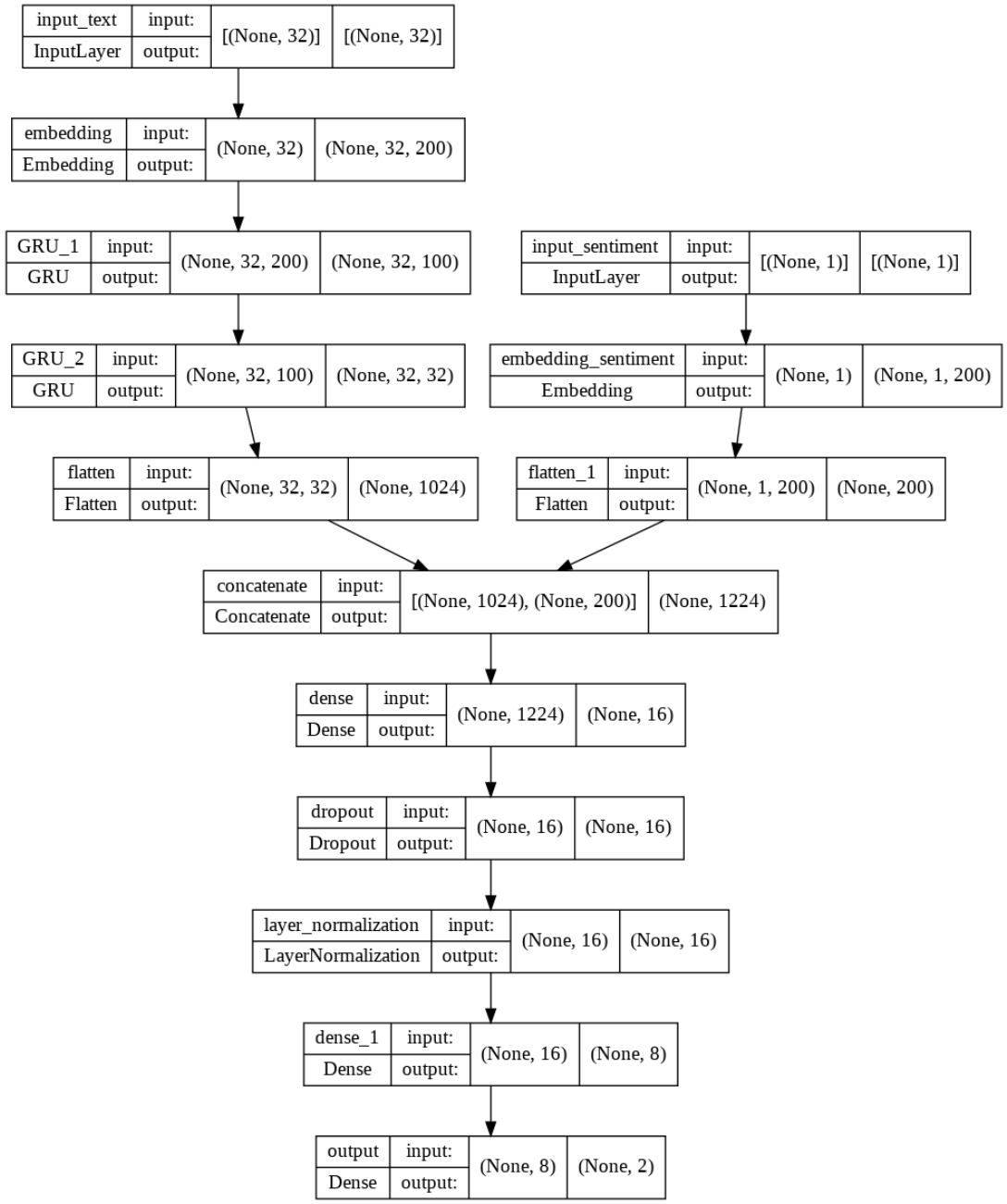
sentiment: Negative

selected_text: Leave me alone

Start index: 2

End index : 4

Model:



GRU:

Activation: Tanh, units for 1st GRU: 100, units for 2nd GRU: 32

Dense:

Activation: Relu, units for 1st Dense: 16, units for 2nd Dense: 8, Kernel regularizer: l2

Dropout:

Rate: 0.6

Output:

Activation: Linear, units: 2

Loss: Mean squared Error

Optimizer: Adam

The above is the model architecture is the model where the input to the model is text and sentiment values. These values are passed to a GRU layer and finally, we have a Dense layer that predicts the start and end indexes to get the selected_text values from the text. The final output layer outputs 2 values where the 1st neuron corresponds to start index and 2nd neuron corresponds to end index.

Results:

Using this base model, we got training Jaccard score of 0.5634 and testing Jaccard score of 0.5596.

Inference:

	text	pred_text
3345	one final down, two more to go!! wish me luck!...	luck!! no a great effort!!
3979	it`s Blockbuster week in New Zealand - 'Wolver...	Blockbuster week in
1077	Up is out? I didn`t get the memo It looks ...	looks amazing.
26479	Having trouble syncing my iphone to my work ex...	trouble syncing my
25706	**** it...Margie said she couldn`t share the i...	it...Margie said
...	...	...
20095	agree with you about facehunter embarrassing...	you about facehunter
7100	Link won`t open, but I will try it when i hav...	a better connection tomorrow. I`m
25804	thanks I`ll check it out! i`ll be staying a T...	thanks I`ll
18041	Somebody accidentely sleep for 3 hours instead...	I can`t hang out WORK
17838	that sounds cool! post video!!	that sounds

Inference:

- https://neptune.ai/blog/recurrent-neural-network-guide
- https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

- Sachin, S., Tripathi, A., Mahajan, N., Aggarwal, S., & Nagrath, P. (2020). Sentiment analysis using gated recurrent neural networks.
  SN Computer Science, 1(2), 1-13.