

# Tweets Sentiment Analysis

Kuo-Wei Ho<sup>1</sup>, Hao-Chien Wang<sup>2</sup>

<sup>1</sup>NTUEE

<sup>2</sup>NTUPhys

Data Science Programing, July 2019

Instructor: 蔡芸琤

TA: 王冠人, 萬俊彥, 何承諭

# Table of Contents

- 1 Problem
- 2 Key tools
- 3 Steps
- 4 Results
- 5 Difficulties encountered

# Problem

Given a set of data containing 1,600,000 tweets and the sentiment of each tweets. Create a model that can analyze sentiment of new tweets.

Table: Data example

sentiment	Post ID	User ID	tweets
0	1467814192	Ljelli3166	blagh class at 8 tomorrow
0	1467821455	CiaraRenee	I need a hug
4	1677796507	FoodAllergyBuzz	@otibml Thx for the tweet!
4	1677796519	lakido	Sunshine.....I LOVE this weather!!!

0: Negative

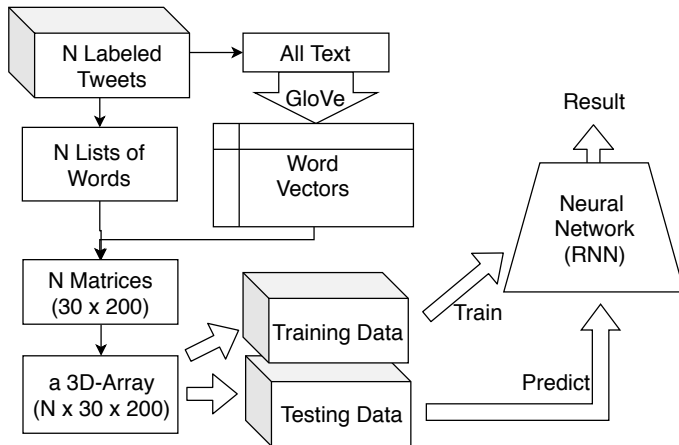
4: Positive

Data: <https://www.kaggle.com/kazanova/sentiment140>

Github link: <https://github.com/b07901135/2019dsp-summer-project>

- Vectorizing text: *GloVe* (*Global Vectors for Word Representation* by Stanford University.)
- Neural network: *RNN* (*Recurrent Neural Network*)

# Steps: Overview



*Note: The testing data are not in the text fed to GloVe.*

# Steps: Overview

- 1 Clean the data: remove non-UTF8 symbols, numbers and URLs.
- 2 Combine all tweets into one string and tokenize.
- 3 Feed the tokens to GloVe to generate word vectors.
- 4 Tokenize all tweets and search each words in the vectors to transform it into a list of matrices.
- 5 Train the RNN model with the list of metrics.
- 6 Test the result with testing data.

# Steps: Data Cleaning and Vectorization

- 1 Replace URLs as “url”
- 2 Replace name tags ( e.g. @allen1234 ) as “names”
- 3 Remove other non-UTF8 characters (`stri_enc_toutf8()` doesn't help)
- 4 Combine tweets into a string, tokenize (and remove stopwords).
- 5 Generate TCM, feed it to the neural network to fit the model.
- 6 Generate word vectors (  $Dim = 200$  ).

Table: Word vectors

"peanuts"	-0.55638	0.04843	-0.14483	-0.47563	...
"permission"	0.15835	0.06962	0.04398	-0.27275	...
"beast"	-0.20607	0.16818	-0.17708	-0.26557	...
"eva"	0.32598	0.04554	-0.72075	-0.04571	...
"pounding"	0.67231	0.00862	-0.07067	-0.15407	...

# Steps: Tweets Vectorization

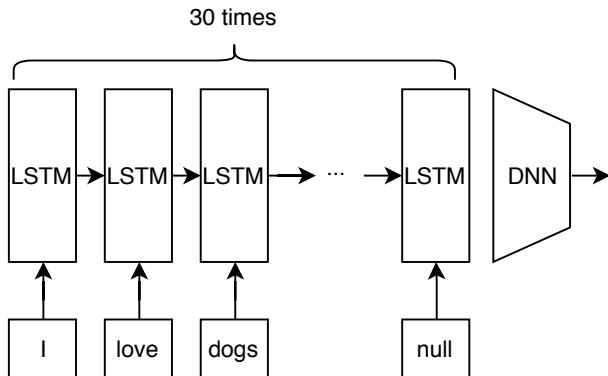
- Discard data other than **sentiment** and **tweets text**
- Tokenize tweets and lookup the tokens in the word vectors.
- Discard tweets containing more than **30 tokens** so that the matrices will not contain too much zeros.
- **Due to the limitation of RAM size, we are only able to use 50,000 tweets data.**

Table: Data manipulation

sentiment		tweets				
0		blagh	class at 8 tomorrow	⇒		
0			I need a hug			
4		@otibml	Thx for the tweet!			
4		Sunshine!	I LOVE this weather!!!			
sentiment		tweets			sentiment	tweets
0		"blagh"	"class" "at" "num" "tomorrow"	⇒	0	<b>A</b> <sub>30×200</sub>
0			"i" "need" "a" "hug"		0	<b>B</b> <sub>30×200</sub>
4			"name" "thx" "for" "the" "tweet"		4	<b>C</b> <sub>30×200</sub>
4		"sunshine"	"!" "i" "love" "this" "weather" "!" "!" "!"		4	<b>D</b> <sub>30×200</sub>



# Steps: RNN Fitting



# Result

- The best accuracy we got on 5,000 testing data is 78.68%

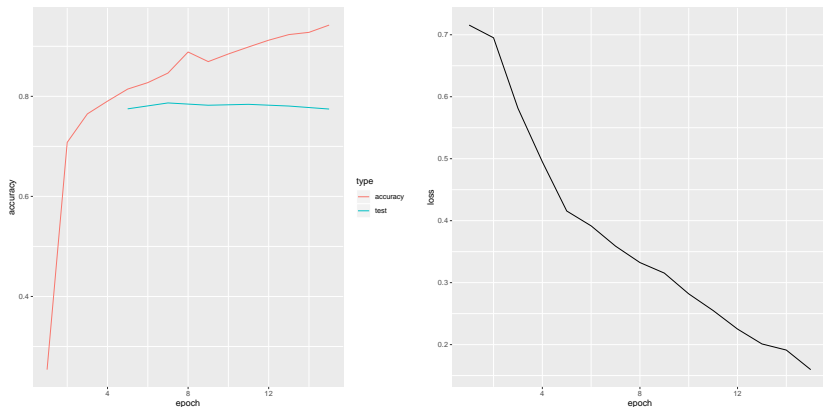


Figure: Training process.

- In fact, we got an accuracy of 70~ 90% ourselves. (20 testing data)

# Difficulties Encountered

- 1 Hardware limitations (Ram size, CPU/GPU speed): Kill X session, `gc()/rm()`
- 2 Package problems (Tensorflow)
- 3 Carelessness on manipulating data, leading to incorrect results.
- 4 Large data size causing difficulties checking results and big waste of time.

- `save()/load()`
- `pbapply`
- `gc()`
- `rm()`
- `abind()`
- `melt()`