Twitter Sentiment Analysis

Kuo-Wei Ho[1], Hao-Chien Wang[2]

[1]NTUEE

[2]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

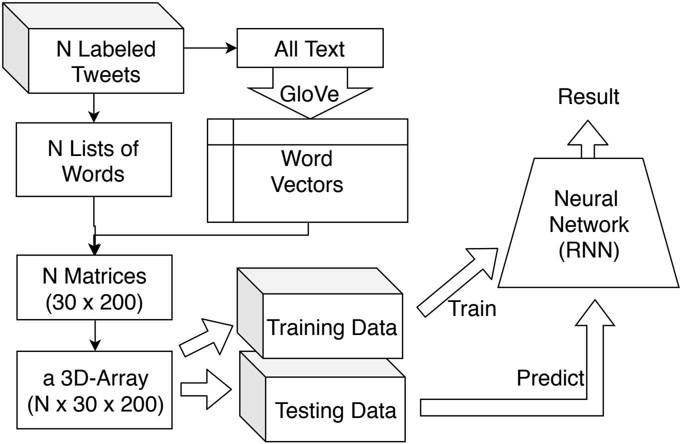
Data: `https://www.kaggle.com/kazanova/sentiment140`

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

Key Tools

- Vectorizing text: *GloVe (Global Vectors for Word Representation* by Standford 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 matrices.
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 “name”
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!!!

⇒

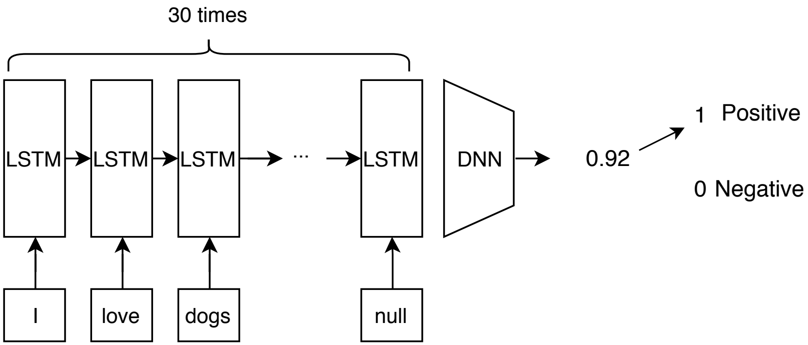
sentiment	tweets
0	"blagh" "class" "at" "num" "tomorrow"
0	"i" "need" "a" "hug"
4	"name" "thx" "for" "the" "tweet"
4	"sunshine" "!" "i" "love" "this" "weather" "!" "!" "!"

⇒

sentiment	tweets
0	$\mathbf{A}_{30\times 200}$
0	$\mathbf{B}_{30\times 200}$
4	$\mathbf{C}_{30\times 200}$
4	$\mathbf{D}_{30\times 200}$

Steps: RNN Training

- LSTM(*Long Short-Term Memory*):
  A Type of RNN(*Recurrent Neural Network*) that has a memory cell and can change the value stored depends on the input vector.



Result

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

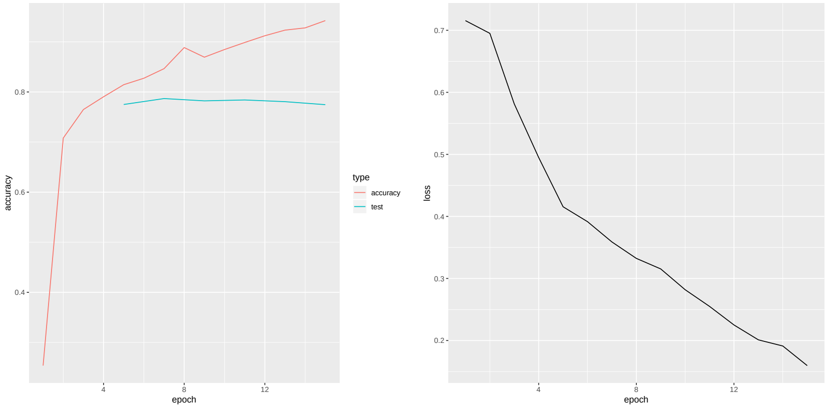


Figure: Training process.

Result

- In fact, we got an accuracy of $60 \sim 80\%$ ourselves. (10 testing data)
- We found that the performance of the model is better when the embedding word vectors are computed only from testing datas.
- The performance is better with bigger amount of data.

Difficulties Encountered

1. Hardware limitations (Ram size, CPU/GPU speed): Work in TTY, `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.

Dark Magic

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