Twitter data sentiment analysis

Lize Du, Chen Xing, Jitian Zhao

STAT 479

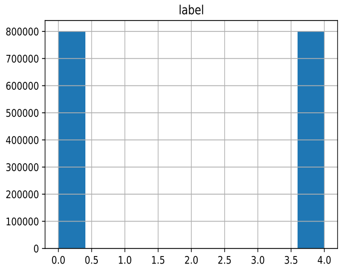
April 29, 2019

Outline

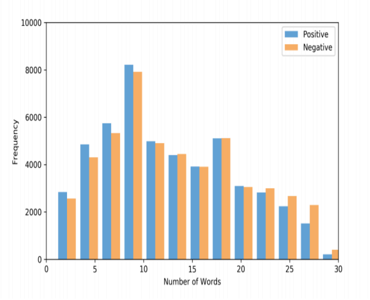
- Data cleaning
  - ✓ NLTK and regular expression
  - ✓ Word Embedding
- Naive Bayes
- Multi-layer perceptron
- Convolution neural network
- RNN
- LSTM
  - ✓ Normal LSTM
  - ✓ Bidirectional LSTM
- GRU

Sentiment140

- 2-class text classification problem
- 1.6 million labeled trainning samples randomly seperated into validation and training dataset.



(a) label overview



(b) label over number of words

Figure: label distribution

Data Cleaning: NLTK

- Stopwords use *nltk.stopwords.words* except:
  1. Adverb of degree: few, most, more...
  2. Negative: don't, didn't, doesn't aren't...
- Pattern matching: words, abbreviation, [a-zA-Z]-[a-zA-Z], ... , ?, !
- Substitute: n'/n't→not, 'd→would...
- Delete: noun's, number+th/st/nd/rd;
- Change to lower case;
- Add _neg to the words between not/never and the first punctuation(naive bayes);
- Use porter stemmer to do stem extracting, such as amazing→amaz(naive bayes);
- Use wordnet lemmatizer to lemmstize the verb to a normal form, such as loving→love

Data Cleaning

Figure: a text example from Twitter

- Replace the certain features in tweets:
  1. '!!!!!' to '!'<repeat>
  2. website to <url>
  3. @user to <user>
  4. number to <number>
  5. 'loveeeee' to 'love'<elong>

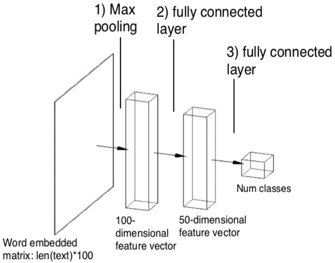
Word Embedding

- Use pretrained word embedding: glove.twitter.27B.100d
  1. Use large set of twitter texts to train word vector.
  2. Special tokenization for tweets dataset.
  2. Word vector has similar semantic meaning to their neighbors.

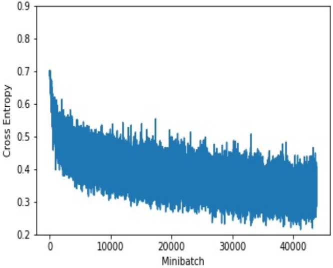
Naive Bayes

- Use it as a baseline to compare with our deep learning method
- Use the data processed by NLTK and regular expression
- Use the most frequent 371 words(term frequency>=6000)
- The test accuracy is 0.7194, the confusion matrix is $\begin{matrix} 0.66 & 0.34 \\ 0.22 & 0.78 \end{matrix}$
- The advantage of naive bayes is it is fast and simple; The disadvantage is its accuracy is relatively low.

Multi-layer perceptron



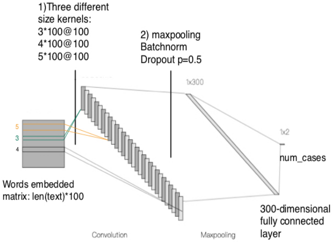
(a) model architecture



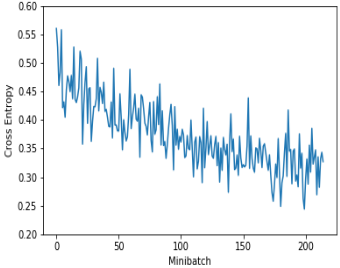
(b) minibatch loss

- Use max-pooling to make size match.
- Test accuracy:80.5%
- Advantage: simple model, very fast (1 min/epoch). Disadvantage: accuracy is not satisfactory. begin to over-fit after 3 epoches.

Convolutional neural network



(c) model architecture



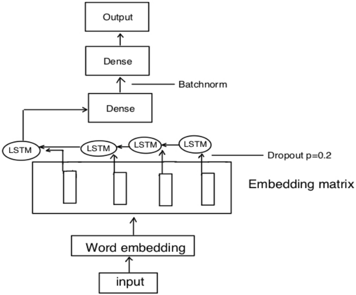
(d) minibatch loss

- Implemented the model in paper *Convolutional Neural Networks for Sentence Classification*.
- Test accuracy:84.15%
- Advantage: fast (2 min/epoch), acceptable accuracy

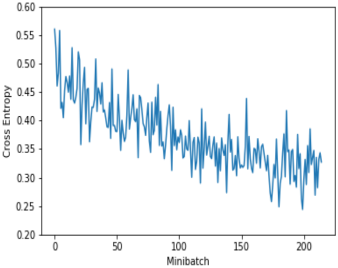
RNN

- Too slow to train(more than 4 hours for one epoch)
- Vanishing gradient problem.

Normal LSTM



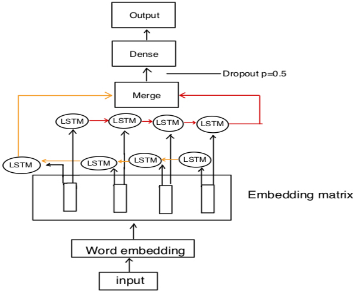
(e) model architecture



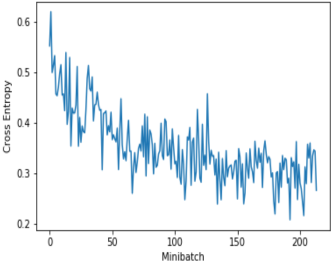
(f) minibatch loss

- The test accuracy is 81.22%
- Disadvantage: low accuracy.

Bidirectional LSTM



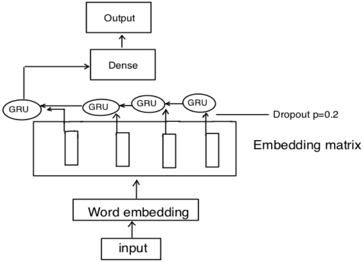
(g) model architecture



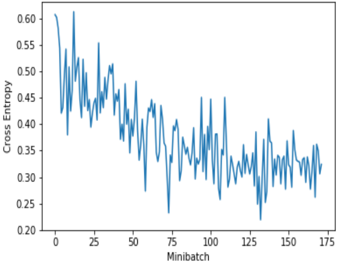
(h) minibatch loss

- The test accuracy is 86.72%
- a forward RNN and a backward RNN.

GRU



(i) model architecture



(j) minibatch loss

- Fewer parameters to train. Much faster.
- Satisfactory result. Test accuracy 86.33%.

Model Comparison

Table: result comparison

model	run-time	test accuracy
Naive Bayes	really fast	71.9%
MLP	$\approx 1min/epoch$	80.5%
CNN	$\approx 2min/epoch$	84.15%
RNN	>4 hour/epoch	/
Normal LSTM	>50 min/epoch	81.27%
Bidirectional LSTM	>3 h/epoch	86.72%
GRU	$\approx 1.5h/epoch$	86.33%

Future work

- Try other RNN models.
- Use a more stable cloud computing engine to estimate a more precise training time.


not_neg
back_neg
use
sweet
lot
spot
hope
larg
long
absolut
do_neg
half
walk
excel
fresh
fun
select
were_neg
instead again_neg
me_neg
star
thank
still
ask
pay
disappoint
howev
mayb
manag
highli
minut
bad
decent
ok
seem
tell
call
noth
leave
tasti
final
if_neg
fantast
he
bite
sit
amaz
awesom
worth
arriv
favorit
notder
last
perfect
or_neg
at_neg
away