Supervised Sentiment Analysis with BiLSTM and Text CNN

Jin Huang
Heidelberg University
huang@cl.uni-heidelberg.de

Abstract

I report 2 experiments with different word-level Neural Network for Tweet-level classification tasks. In this project, I made naive experiments with two deep-learning systems. I trained both models on the corpus "*Sentiment 140*" (English tweets). I use bidirectional Long Short-Term Memory (BiLSTM) networks and Text Convolutional Neural Network(text CNN) on top of word glove embeddings pre-trained on a huge amount of Twitter messages.

Motivation

Sentiment analysis is an important area in Natural Language Processing (NLP), which mainly studies the recognition and classification of emotions in the text. To analysis the emotion/sentiment of social media texts is very complicated and difficult because of the incorrect spelling, improper use of grammar, Internet language, informal abbreviations, and so on. Besides, the shot texts also limited the contextual information which the model could learn from. I would like to have a model that could properly and efficiently preprocess Twitter text. Although there are many challenges, it can still mostly successfully predict emotions after training.

Preprocessing

- tokenization and lowercase
- removes URLs and non-asciis(emojis)
- removes hashtags and mentions
- removes punctuation(except for "’" to keep the Syntax abbreviation) and extra whitespaces
- removes stop words(except for negative words because they are important for sentiment analysis)
- filters out sentences which contain less than 4 words
- padding and uses `nn.utils.rnn.packed_padded_sequence`, this makes the Network only process the elements, which wasn’t padded. Besides, all the padded element in the output will be a zero tensor.

The first 6 pairs of preprocessed tweets and their labels:

label	text
0	wendy potato far well nuggets lunch decision fail
1	yeeeeeesssss ashley win happy
0	another lush day weather wish stuck work day night soooooooo not fair
1	week till shopbroker course over
1	rest ready next week
0	bed cousin not look forward work

Pretrained Embeddings

The word embeddings will be initialized with the `glove.twitter.27B.100d` pre-trained vectors. These *100-dimensional* vectors of `glove` were trained on 2 billion tweets(27 billion tokens). Theoretically, these pre-trained vectors provide a pretty good initialization to the embedding layer, which is much better than to train again from the training data set, because in this vector space there are already words with a similar semantic meaning close together.

Corpus

The corpus *Sentiment140* allows us to detect the sentiment on Twitter. It contains 1,600,000 tweets extracted using the twitter API. Although this corpus officially claims to have three labels for the emotions (*0* = negative, *2* = neutral, *4* = positive), in fact there are only two labels (*0* = negative, *4* = positive). So I replaced 4 with 1 for an easier binary classification (*0* = negative, *1* = positive). I only extracted the label column and the text column for supervised learning. After the preprocessing there are 1,295,743 tweets remained.

Corpus statistic(Preprocessed)

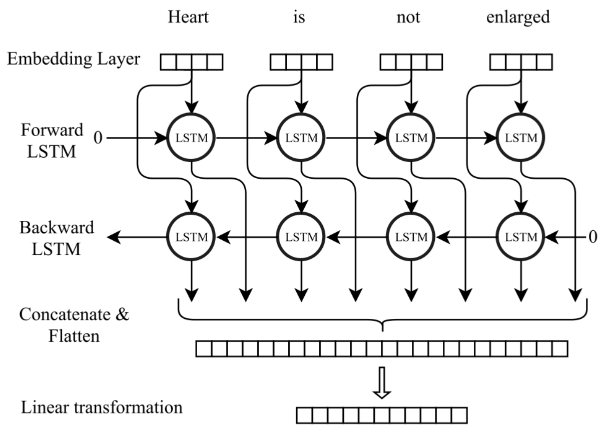
	Train	Valid	Test	
0 (negative)	464631	99419	99662	663712
1 (positive)	442389	94942	94700	632031
	907020	194361	194362	1295743

Setting

- *Batch size*: 128
- *Optimizer*: `Adaptive Moment Estimation(Adam)`, which adapts the learning rate for every parameter. It gives lower learning rates to parameters that are updated more often, while giving higher learning rates to parameters that are updated less often, so that we don't have to do experiments for choosing a learning rate
- *Loss*: `binary cross-entropy`(performed by `BCEWithLogitsLoss()`)
- *Activation function*: `sigmoid`(performed by `BCEWithLogitsLoss()`)
- *Prediction Output*: 0/1

Models

1. Model 1: Bidirectional Long Short-Term Memory (BiLSTM)



BiLSTM
(Source: Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks, Cornegruta et al)

RNNs are difficult to train (*Pascanu et al., 2013*), especially because the gradients may grow or decay exponentially over long sequences (*Bengio et al., 1994; Hochreiter et al., 2001*). To avoid this, I employ a 2-layer Bidirectional LSTM. LSTM was introduced by *Hochreiter & Schmidhuber (1997)*. While Standard RNNs suffer from the vanishing gradient problem, a LSTM network is a RNN model which is explicitly designed to avoid the long-term dependency problem. It learns to keep the important content for long periods of time and forget the non-relevant content. In addition to letting the LSTM forward process the words of the sentence from the beginning to the end, there is a second LSTM that backwards processes the words in the sentence from the end to the first. At the end, the last hidden state of the forward LSTM(from the final word) and the last hidden state of the backward LSTM (from the first word) will be concatenated and been predicted.

Regularization

To avoid overfitting, I use Dropout. The possibility that the neurons in a layer during a forward pass would be dropping out is 50% so that we won't let all the parameters always be learned.

Run Time

10-70m per epoch.

2. Model 2: Text Convolutional Neural Network (CNN)

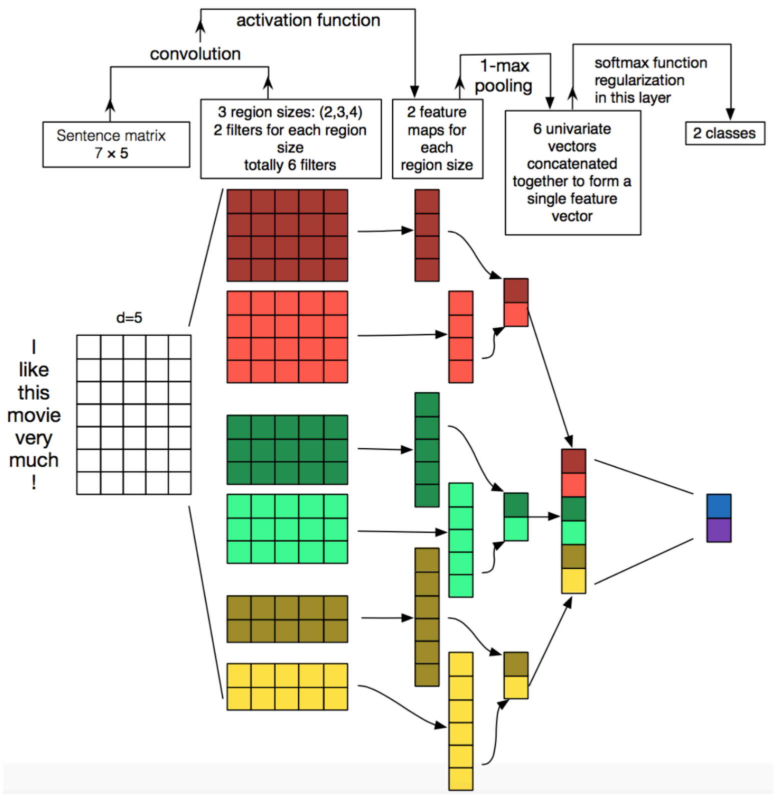

activation function
convolution
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

Illustration of a Convolutional Neural Network (CNN) architecture for sentence classification
(Source: Source: Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.)

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features *(LeCun et al., 1998)*. They are made up of convolutional layer(s), followed by linear layer(s). The first ever "convolutional network" was the Neocognitron by *Fukushima (1980)*, and the Convolutional Neural Network(CNN) was proposed by *Yann LeCun et al in 1989*, who also proposed the Back-propagation. Nowadays, CNNs are not only responsible for the breakthroughs in image classification, but also in Natural Language Processing.

Filter

*Size:[2, 3, 4]*100 = 300 filters*
A 1xn filter can view a n-gram in a sequence.
I used 3 filters with sizes, which will look at the bi-grams tri-grams, and four-grams. Using even bigger sizes of filters would still make semantical sense. Ideally, the co-occurrence of some of the n-grams within the tweets will be a clue of the final sentiment. Each filter would learn a different feature and extract it. Within each filter, each element has one weight associated with it, which is learned through the back-propagation. Therefore, if we have good embedding of all the vectors, we can detect more high-level meaning of n-grams using convolution. The output of a filter is a weighted sum of all elements covered by the filter. And these outputs are the output of the convolutional layers.

Activation Function

ReLU
I used the `ReLU` activation function on the outputs of the convolutional layers.

Pooling

Max pooling
The maximum value over a dimension of the output of the convolutional layers would be selected, because theoretically it should contain the most important information(n-gram) of the sequence.
We would then concatenate the 300 different outputs and pass them into a linear layer.

Regularization

I use dropout on the concatenation of the filter outputs.

Run Time

<10m per epoch.

Result

Model	Train Loss	Train Acc.	Valid. Loss	Valid. Acc.	Test Loss	Test Acc.	Epoch
BiLSTM	**0.348**	84.65%	**0.485**	**79.47%**	**0.423**	**80.51%**	50
Text CNN	0.149	**93.98%**	1.626	74.56%	0.442	79.42%	100

For the *Sentiment140* corpus, the approach achieves reasonable results for sentence sentiment prediction in BiLSTM, with. 80.51% accuracy, and Text CNN, with 79.42% accuracy. The test accuracies of both models are almost identical. However, in the CNN model, the training accuracy is subjectively far higher than the test accuracy, which may indicate over-fitting.

Conclusion

For this project, as a Neural Network beginner, I made experiments on two naive deep-learning systems for tweet text sentiment analysis. Comparing to RNN, one of the greatest advantages of CNNs is that they are really very fast and really easy to train, because they learn good representations automatically without representing the whole vocabulary. However CNNs have no notion of order, thus when applying them to NLP tasks the crucial information of the word order is lost. On the contrary, BiLSTMs may understand the entire context better. But I personally prefer CNN, because of the speed and the detection of the high-level sentiment relation using filters with different sizes.

For future work, I would definitely work on a more prorate preprocessing for social media text and maybe add some attentions to optimize the nets. And try to combine BiLSTM and CNN in one model.

References

1. LONG SHORT-TERM MEMORY, Hochreiter & Schmidhuber (1997)

2. Bidirectional recurrent neural networks, Schuster, Mike, and Kuldip K. Paliwal

3. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition, Alex Graves, Santiago Fernández, and Jürgen Schmidhuber

4. Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks, Savelie Cornegruta, Robert Bakewell, Samuel Withey and Giovanni Montana

5. Convolutional Networks for Images, Speech, and Time-Series, Yann LeCun and Yoshua Bengio

6. Neocognitron, Dr. Kunihiko Fukushima

7. glove.twitter.27B.100d

8. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis, Christos Baziotis, Nikos Pelekis, Christos Doulkeridis

9. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification, Zhang, Y., & Wallace, B. (2015)

10. Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks, Cornegruta et al)

11. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies , *Bengio et al., 1994; Hochreiter et al., 2001*

12. On the difficulty of training recurrent neural networks, *Pascanu et al., 2013*

13. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts, C´ıcero Nogueira dos Santos and Ma´ıra Gatti