Task 2 - Text Clf By NN

Methods and Targets

We use neural network methods (including CNN and RNN) to solve this problem. Within each method, we want to find out the impacts of different initial seeds (word embeddings like Word2Vec and GloVe) to the classification results.

0. Data Description

Sentiment Analysis on Movie Reviews, including training dataset and test dataset.

In the training dataset, each item contains a phrase and its sentiment (from 0 to 4). In the test dataset, there are only phrases without sentiment. We need to train a classifier with training dataset and predict the sentiments of the phrases in the test dataset via the trained model.

1. Word Embedding

Word embedding, that is, embedding high-dimensional word vectors into a low-dimensional space. This is a pre-trained model that transforms text into vector or matrix. On this basis, we can carry out a series of natural language processing like sentiment analysis, text similarity analysis and translations.

1.1 Word2Vec

The `Word2Vec` model is actually a simplified neural network, which trains the input one-hot vector into a well-trained vector with a fixed length.

The `Word2Vec` model is generally divided into two sub-models: `CBOW` (continuous bag of words) and `skip-gram`. The training input of `CBOW` model is the word vector corresponding to the **context** of a specific word, and the output is the word vector of the specific word. The idea of `skip-gram` model is just the opposite, that is, the input is the word vector of a specific word, and the output is the word vector of **context**.

When the model is trained, the final result is actually the **weights of the neural network**. For example, now input an X one-hot encoder: [1,0,0,...,0], corresponding to the word "You", only the weight corresponding to the position 1 is activated in the weight from the input layer to the hidden layer, and the **number of these weights is consistent with the number of nodes** in the hidden layer. So these weights form a vector VX to represent word X. Because the position of element 1 in the one-hot encoder of each word is different, this vector VX can be used to uniquely represent word X.

More details in Tomas Mikolov's papers.

1.2 GloVe

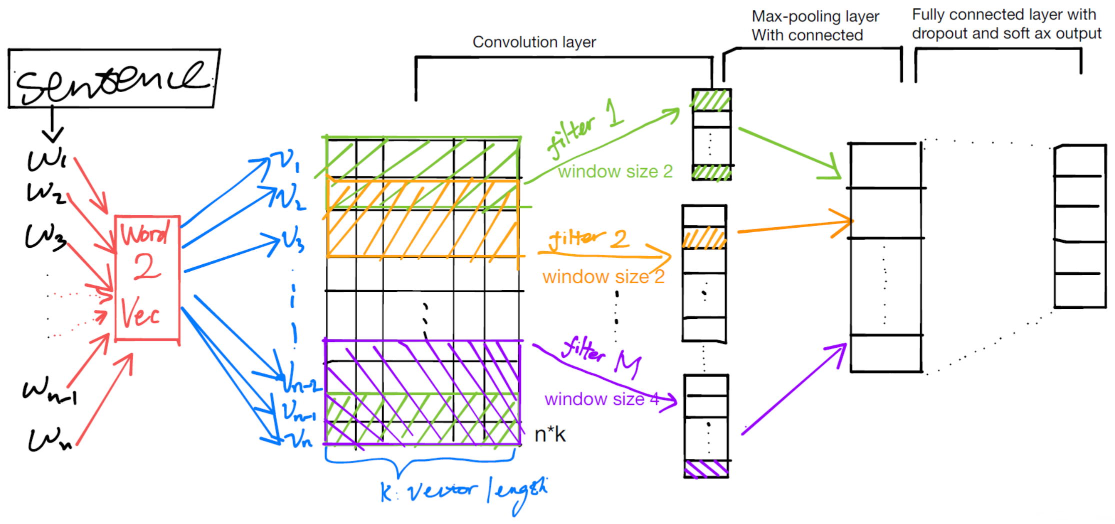
In order to overcome the defects of local context window, in 2014, Jeffrey Pennington et al. proposed a new method named `GloVe`, which is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Thus it combines the advantages of statistical information and local context window method.

more details in Jeffery's GloVe: Global Vectors for Word Representation.

2. Neural Network Architecture

2.1 CNN

Referring to Yoon Kim's Convolutional Neural Networks for Sentence Classification, here we take a simplified form - CNN-static, with different initial word vectors. And we only consider the convolutional input layer with one channel.

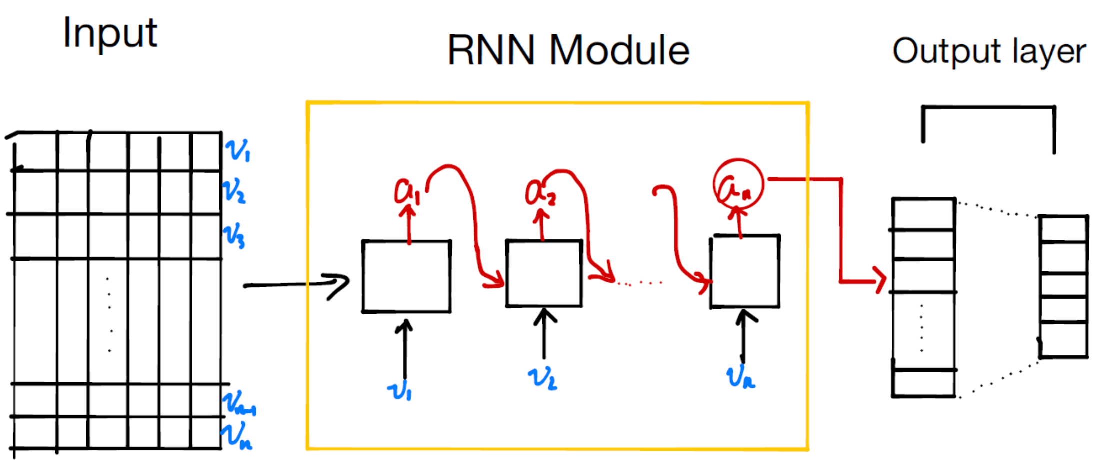


In this project, we consider the following pre-trained word vectors:

- Google News Word2Vec with length 300
- Reuters New Word2Vec trained by ourselves with length 100
- Reuters New Word2Vec trained by ourselves with length 300
- Word2Vec trained by the training dataset with length 100
- Word2Vec trained by the training dataset with length 300
- Trained GloVe with length 100
- Trained GloVe with length 300

2.2 RNN

To imply RNN, we also need to vectorize the words first.



In RNN Module, we can stack several RNNs (in other words, RNN layer is greater than 1). More layers leads to more connections and also more computations. But the task is not so hard to deal with, hence, we take number of RNN layers to be 1. And each cell in the RNN Module can take typical RNN cell or LSTM cell which introduces input gate, output gate and forget gate to solve the long term memory dependence problem of RNN. Here we use the LSTM cell.

In this part, we mainly focus on

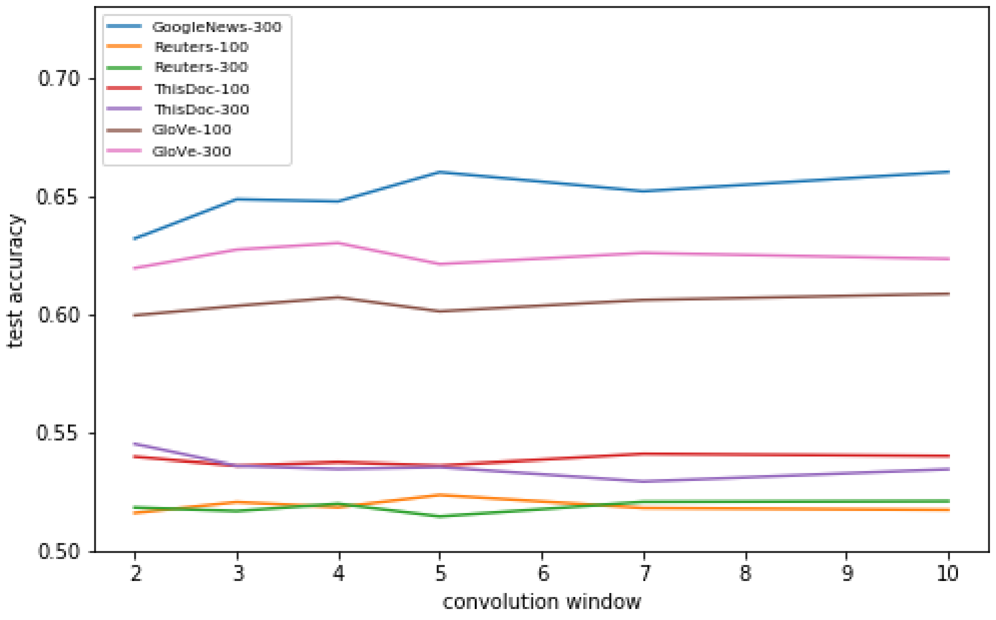
- Google News Word2Vec with length 300
- Word2Vec trained by the training dataset with length 100
- Word2Vec trained by the training dataset with length 300
- Trained GloVe with length 100
- Trained GloVe with length 300

3. Experimental Results

3.1 CNN

Hyper Parameters: EPOCH=2, LR=0.01, # conv layer=1, # fc layer=2

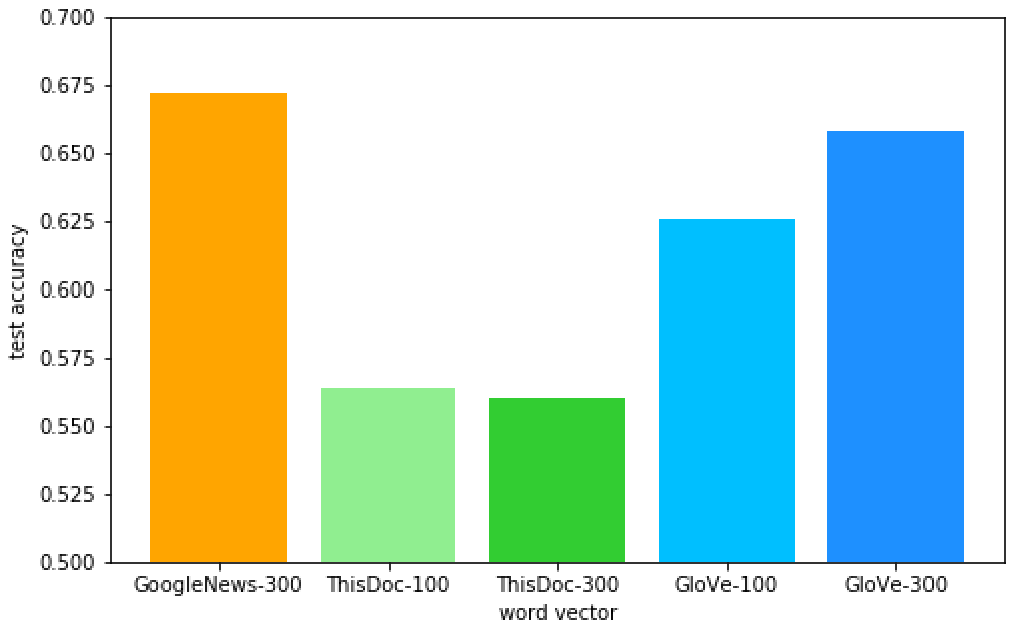
WV\window	2	3	4	5	7	10
Google-300	0.632	0.6486	0.6477	0.6601	0.652	0.6602
Reuters-100	0.5161	0.5207	0.5185	0.5237	0.5182	0.5174
Reuters-300	0.5184	0.5169	0.5200	0.5146	0.5208	0.5211
This Text-100	0.5397	0.5360	0.5375	0.5361	0.5409	0.5400
This Text-300	0.5451	0.5360	0.5347	0.5355	0.5294	0.5346
GloVe-100	0.5995	0.6035	0.6071	0.6012	0.6060	0.6086
GloVe-300	0.6195	0.6273	0.6301	0.6212	0.6259	0.6234



3.2 RNN

Hyper Parameters: EPOCH=2, LR=0.01, # rnn layer=1

WV	google-300	text-100	text-300	GloVe-100	GloVe-300
accuracy	0.6721	0.5638	0.5601	0.6258	0.6583



4. Conclusions

Within CNN method, taking different word vectors as input will lead to quite different test results. In conclusion, `GoogleNews-300` gives the highest test accuracy, followed by `GloVe-300` and `GloVe-100`. Word vectors trained by Reuters corpus give the worst results. As for the selection of convolution window, it can be taken as a value near 5, since in most cases, the test accuracy increases first and then goes down.

And for RNN method, the priority of word vector is still the same, `GoogleNews > GloVe > ThisDoc`.

Compare CNN and RNN horizontally, we can find that RNN gives a better result than CNN does. But RNN cost much more time than CNN does.

Concluded in a word, CNN gives the best test accuracy of 66.02% and RNN gives the best test accuracy of 67.21%.