Text Classification Using CNN, LSTM and Pre-trained Glove Word Embeddings: Part-3

Sabber Ahamed Follow
Jan 13, 2018 · 3 min read

This is a part of series articles on classifying Yelp review comments using deep learning techniques and word embeddings. In the last part (part-2) of this series, I have shown how we can use both CNN and LSTM to classify comments. In this part, I keep the same network architecture but use the pre-trained glove word embeddings.

Followings are the list of brief contents of the different parts:

- **Part-1**: In this part, I build a neural network with LSTM and word embeddings were learned while fitting the neural network on the classification problem.
- **Part-2**: In this part, I add an extra 1D convolutional layer on top of the LSTM layer to reduce the training time.
- **Part-3**: In part-3, I use the same network architecture as part-2, but use the pre-trained glove 100 dimension word embeddings as initial input.
- **Part-4**: In part-4, I use word2vec to learn word embeddings.

Import Libraries

```
# Keras
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Flatten, LSTM, Conv1D, MaxPooling1D,
Dropout, Activation
from keras.layers.embeddings import Embedding

## Plotly
import plotly.offline as py
```

```
import plotly.graph_objs as go
py.init_notebook_mode(connected=True)

# Others
import nltk
import string
import numpy as np
import pandas as pd
from nltk.corpus import stopwords

from sklearn.manifold import TSNE
```

Data processing

Data processing involves the following steps:

1. Remove numeric and empty texts
2. Convert five classes into two classes (positive = 1 and negative = 0)
3. Remove punctuation from texts
4. Convert words to lower case
5. Remove stop words
6. Stemming

Detail code of the data processing can be found on Part-1.

Use pre-trained Glove word embeddings

In this subsection, I use word embeddings from pre-trained Glove. It was trained on a dataset of one billion tokens (words) with a vocabulary of 400 thousand words. The glove has embedding vector sizes: 50, 100, 200 and 300 dimensions. I chose the 100-dimensional one. I intentionally keep the “trainable” parameter as ‘False’ (see in the code below) to see if the model imporves while keeping the word embeddings fixed.

Extract word embeddings from the Glove

```
embeddings_index = dict()
f = open('glove.6B/glove.6B.100d.txt')
```

```
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
```

Create a weight matrix

```
embedding_matrix = np.zeros((vocabulary_size, 100))
for word, index in tokenizer.word_index.items():
    if index > vocabulary_size - 1:
        break
    else:
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[index] = embedding_vector
```

Develop a model

```
## create model
model_glove = Sequential()
model_glove.add(Embedding(vocabulary_size, 100, input_length=50,
weights=[embedding_matrix], trainable=False))
model_glove.add(Dropout(0.2))
model_glove.add(Conv1D(64, 5, activation='relu'))
model_glove.add(MaxPooling1D(pool_size=4))
model_glove.add(LSTM(100))
model_glove.add(Dense(1, activation='sigmoid'))
model_glove.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

## Fit train data
model_glove.fit(data, np.array(labels), validation_split=0.4, epochs
= 3)
```

Word embedding visualization

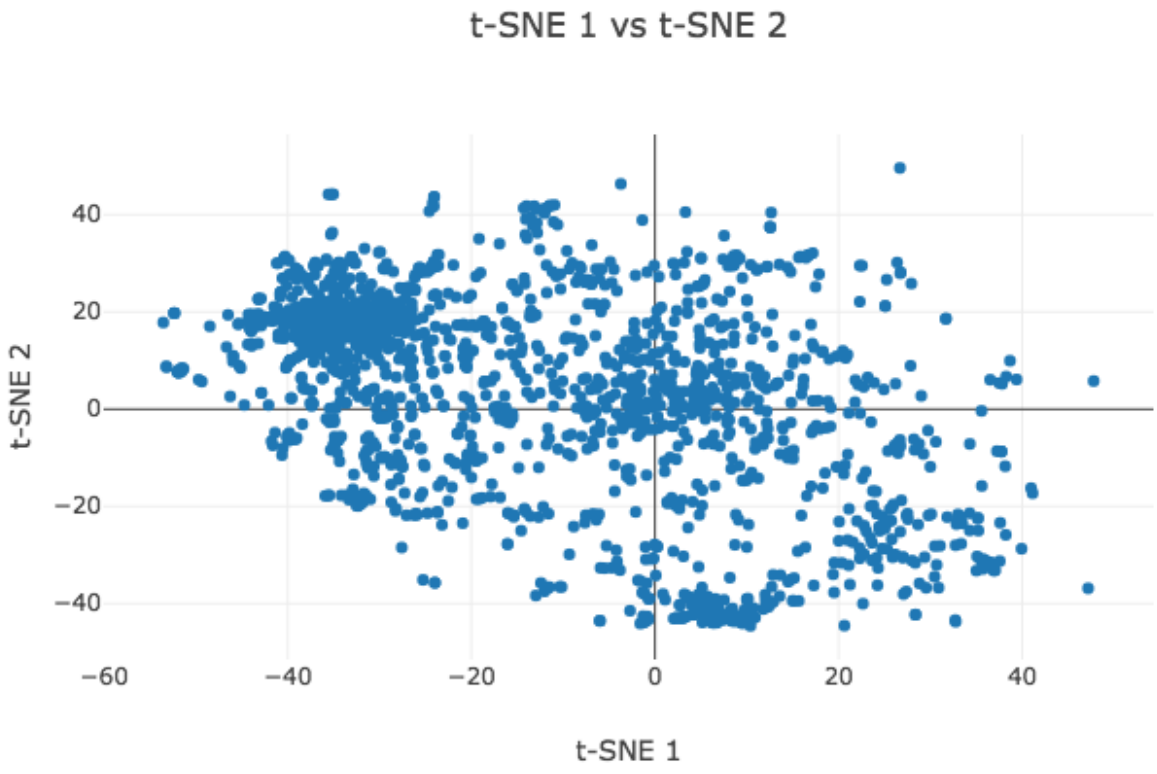
In this subsection, I want to visualize word embedding weights obtained from trained models. Word embeddings with 100 dimensions are first reduced to 2 dimensions using

t-SNE. Tensorflow has an excellent tool to visualize the embeddings nicely, but here I want to visualize the word relationship.

```
## Get weights
glove_embds = model_glove.layers[0].get_weights()[0]

## Plotting function
def plot_words(data, start, stop, step):
    trace = go.Scatter(
        x = data[start:stop:step,0],
        y = data[start:stop:step, 1],
        mode = 'markers',
        text= word_list[start:stop:step]
    )
    layout = dict(title= 't-SNE 1 vs t-SNE 2',
                  yaxis = dict(title='t-SNE 2'),
                  xaxis = dict(title='t-SNE 1'),
                  hovermode= 'closest')
    fig = dict(data = [trace], layout= layout)
    py.iplot(fig)

## Visualize words in two dimensions
glove_tsne_embds = TSNE(n_components=2).fit_transform(glove_emds)
plot_words(glove_tsne_embds, 0, 2000, 1)
```



Thank you very much for reading. In the next part, I will discuss how we can obtain and visualize word embeddings using Word2Vec. The full code can be found on Github. Let me know if you have any question, or if this article needs any correction.

Convolutional Network Recurrent Neural Network Word Embeddings Gloves

Natural Language Process