Introduction

In this project, I classify Yelp round-10 review datasets. The reviews contain a lot of metadata that can be mined and used to infer meaning, business attributes, and sentiment. For simplicity, I classify the review comments into two class: either as positive or negative. Reviews that have star higher than three are regarded as positive while the reviews with star less than or equal to 3 are negative. Therefore, the problem is a supervised learning. To build and train the model, I first tokenize the text and convert them to sequences. Each review comment is limited to 50 words. As a result, short texts less than 50 words are padded with zeros, and long ones are truncated. After processing the review comments, I trained three model in three different ways:

- **Model-1: In this model, a neural network with LSTM and a single embedding layer were used.**
- **Model-2: In Model-1, an extra 1D convolutional layer has been added on top of LSTM layer to reduce the training time.**
- **Model-3: In this model, I use the same network architecture as Model-2, but use the pre-trained glove 100 dimension word embeddings as initial input.**

Since there are about 1.6 million input comments, it takes a while to train the models. To reduce the training time step, I limit the training epoch to three. After three epochs, it is evident that Model-2 is better regarding both training time and validation accuracy.

Project Outline

In this project I will cover the follwouings :

- **Download data from yelp and process them**
- **Build neural network with LSTM**
- **Build neural network with LSTM and CNN**
- **Use pre-trained GloVe word embeddings**
- **Word Embeddings from Word2Vec**

Import libraries

In [ ]:

```
%%capture
%pip install keras nltk pandas
```

In [ ]:

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

## Plot
import plotly.offline as py
import plotly.graph_objs as go
py.init_notebook_mode(connected=True)
import matplotlib as plt

# NLTK
import nltk
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

# Other
import re
import string
import numpy as np
import pandas as pd
from sklearn.manifold import TSNE
```

```
2023-03-19 17:40:33.853882: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is
optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in perfor
mance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-03-19 17:40:33.990616: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] C
ould not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open shared object f
ile: No such file or directory
2023-03-19 17:40:33.990635: I tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore above
cudart dlerror if you do not have a GPU set up on your machine.
2023-03-19 17:40:34.798360: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] C
ould not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot open shared object file:
No such file or directory
2023-03-19 17:40:34.798455: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] C
ould not load dynamic library 'libnvinfer_plugin.so.7'; dlerror: libnvinfer_plugin.so.7: cannot open share
d object file: No such file or directory
2023-03-19 17:40:34.798464: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot
dlopen some TensorRT libraries. If you would like to use Nvidia GPU with TensorRT, please make sure the mi
ssing libraries mentioned above are installed properly.
```

Data Processing

In [ ]:

```python
#df = pd.read_csv('train.csv', sep = '|', names = ['stars', 'text'], error_bad_lines=False)
import os

if os.path.isfile("./yelp_reviews_5k.csv"):
    df = pd.read_csv("./yelp_reviews_5k.csv", nrows=5000)
else:
    with open("../yelp_academic_dataset_review.json", encoding='utf-8') as f:
        df = pd.read_json(f, orient="records", lines=True, nrows=5000)
    df.to_csv("yelp_reviews_5k.csv", encoding='utf-8', index=False)

df.drop(["review_id", "user_id", "business_id", "date", "useful", "funny", "cool"], axis=1, inplace=True)
df.head()
```

Out[ ]:

	stars	text
0	3	If you decide to eat here, just be aware it is...
1	5	I've taken a lot of spin classes over the year...
2	3	Family diner. Had the buffet. Eclectic assortm...
3	5	Wow! Yummy, different, delicious. Our favo...
4	4	Cute interior and owner (?) gave us tour of up...

In [ ]:

```
df= df.dropna()
df = df[df["stars"].apply(lambda x: str(x).isnumeric())]
df = df[df["stars"].apply(lambda x: x !="")]
df = df[df["text"].apply(lambda x: x !="")]
```

In [ ]:

```
df.describe()
```

Out[ ]:

	stars
count	5000.000000
mean	3.838600
std	1.357983
min	1.000000
25%	3.000000
50%	4.000000
75%	5.000000
max	5.000000

In [ ]:

```
df.head()
```

Out[ ]:

	stars	text
0	3	If you decide to eat here, just be aware it is...
1	5	I've taken a lot of spin classes over the year...
2	3	Family diner. Had the buffet. Eclectic assortm...
3	5	Wow! Yummy, different, delicious. Our favo...
4	4	Cute interior and owner (?) gave us tour of up...

Convert five classes into two classes (positive = 1 and negative = 0)

Since the main purpose is to identify positive or negative comments, I convert five class star category into two classes:

- **(1) Positive: comments with stars > 3 and**
- **(2) Negative: comments with stars <= 3**

In [ ]:

```
labels = df['stars'].map(lambda x : 1 if int(x) > 3 else 0)
```

Tokenize text data

Because of the computational expenses, I use the top 20000 unique words. First, tokenize the comments then convert those into sequences. I keep 50 words to limit the number of words in each comment.

In [ ]:

```
def clean_text(text):

    ## Remove puncuation
    text = text.translate(string.punctuation)

    ## Convert words to lower case and split them
    text = text.lower().split()

    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops and len(w) >= 3]

    text = " ".join(text)
    # Clean the text
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r",", " ", text)
    text = re.sub(r"\.", " ", text)
    text = re.sub(r"!", " ! ", text)
    text = re.sub(r"\/", " ", text)
    text = re.sub(r"\^", " ^ ", text)
    text = re.sub(r"\+", " + ", text)
    text = re.sub(r"\-", " - ", text)
    text = re.sub(r"\=", " = ", text)
    text = re.sub(r"'", " ", text)
    text = re.sub(r"(\d+)(k)", r"\g<1>000", text)
    text = re.sub(r":", " : ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"\s{2,}", " ", text)

    text = text.split()
    stemmer = SnowballStemmer('english')
    stemmed_words = [stemmer.stem(word) for word in text]
    text = " ".join(stemmed_words)

    return text
```

In [ ]:

```
df['text'] = df['text'].map(lambda x: clean_text(x))
```

In [ ]:

```
df.head(10)
```

Out[ ]:

	stars	text
0	3	decid eat here awar go take hour begin end tri...
1	5	i have taken lot spin class year noth compar c...
2	3	famili diner buffet eclect assort : larg chick...
3	5	wow ! yummi differ delici favorit lamb curri k...
4	4	cute interior owner gave tour upcom patio roof...
5	1	long term frequent custom establish went order...
6	5	love tour ! grab groupon price great perfect w...
7	5	amaz amaz wing homemad bleu chees ribey : tend...
8	3	easter instead go lopez lake went los padr nat...
9	3	parti hibachi waitress brought separ sushi ord...

In [ ]:

```
vocabulary_size = 20000
tokenizer = Tokenizer(num_words= vocabulary_size)
tokenizer.fit_on_texts(df['text'])

sequences = tokenizer.texts_to_sequences(df['text'])
data = pad_sequences(sequences, maxlen=50)
```

In [ ]:

```
print(data.shape)
```

```
(5000, 50)
```

Build neural network with LSTM

Network Architechture

The network starts with an embedding layer. The layer lets the system expand each token to a more massive vector, allowing the network to represent a word in a meaningful way. The layer takes 20000 as the first argument, which is the size of our vocabulary, and 100 as the second input parameter, which is the dimension of the embeddings. The third parameter is the input_length of 50, which is the length of each comment sequence.

In [ ]:

```
model_lstm = Sequential()
model_lstm.add(Embedding(20000, 100, input_length=50))
model_lstm.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model_lstm.add(Dense(1, activation='sigmoid'))
model_lstm.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
2023-03-19 17:41:18.848918: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] C
ould not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No su
ch file or directory
2023-03-19 17:41:18.848944: W tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:265] failed call
to cuInit: UNKNOWN ERROR (303)
2023-03-19 17:41:18.848965: I tensorflow/compiler/xla/stream_executor/cuda/cuda_diagnostics.cc:156] kernel
driver does not appear to be running on this host (dc-central): /proc/driver/nvidia/version does not exist
2023-03-19 17:41:18.849181: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is
optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in perfor
mance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

Train the network

There are about 1.6 million comments, and it takes a while to train the model in a MacBook Pro. To save time I have used only three epochs. GPU machines can be used to accelerate the training with more time steps. I split the whole datasets as 60% for training and 40% for validation.

In [ ]:

```
model_lstm.fit(data, np.array(labels), validation_split=0.4, epochs=3)
```

```
Epoch 1/3
94/94 [==============================] - 7s 51ms/step - loss: 0.5606 - accuracy: 0.7233 - val_loss: 0.3814
- val_accuracy: 0.8280
Epoch 2/3
94/94 [==============================] - 5s 53ms/step - loss: 0.2651 - accuracy: 0.8983 - val_loss: 0.3885
- val_accuracy: 0.8360
Epoch 3/3
94/94 [==============================] - 5s 48ms/step - loss: 0.1370 - accuracy: 0.9527 - val_loss: 0.5608
- val_accuracy: 0.8200
```

Out[ ]:

```
<keras.callbacks.History at 0x7f2186124fa0>
```

Build neural network with LSTM and CNN

The LSTM model worked well. However, it takes forever to train three epochs. One way to speed up the training time is to improve the network adding “Convolutional” layer. Convolutional Neural Networks (CNN) come from image processing. They pass a “filter” over the data and calculate a higher-level representation. They have been shown to work surprisingly well for text, even though they have none of the sequence processing ability of LSTMs.

In [ ]:

```
def create_conv_model():
    model_conv = Sequential()
    model_conv.add(Embedding(vocabulary_size, 100, input_length=50))
    model_conv.add(Dropout(0.2))
    model_conv.add(Conv1D(64, 5, activation='relu'))
    model_conv.add(MaxPooling1D(pool_size=4))
    model_conv.add(LSTM(100))
    model_conv.add(Dense(1, activation='sigmoid'))
    model_conv.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model_conv
```

In [ ]:

```
model_conv = create_conv_model()
model_conv.fit(data, np.array(labels), validation_split=0.4, epochs = 3)
```

```
Epoch 1/3
94/94 [==============================] - 5s 33ms/step - loss: 0.5782 - accuracy: 0.6977 - val_loss: 0.4098
- val_accuracy: 0.8145
Epoch 2/3
94/94 [==============================] - 3s 28ms/step - loss: 0.2829 - accuracy: 0.8843 - val_loss: 0.3841
- val_accuracy: 0.8370
Epoch 3/3
94/94 [==============================] - 3s 29ms/step - loss: 0.1198 - accuracy: 0.9583 - val_loss: 0.4727
- val_accuracy: 0.8310
```

Out[ ]:

```
<keras.callbacks.History at 0x7f217c204970>
```

Save processed Data

In [ ]:

```
df_save = pd.DataFrame(data)
df_label = pd.DataFrame(np.array(labels))
```

In [ ]:

```
result = pd.concat([df_save, df_label], axis = 1)
```

In [ ]:

```
result.to_csv('train_dense_word_vectors.csv', index=False)
```

Use pre-trained Glove word embeddings

In this subsection, I want to 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, including 50, 100, 200 and 300 dimensions. I chose the 100-dimensional version. I also want to see the model behavior in case the learned word weights do not get updated. I, therefore, set the trainable attribute for the model to be False.

Get embeddings from Glove

In [ ]:

```
embeddings_index = dict()
f = open('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()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

In [ ]:

```
# create a weight matrix for words in training docs
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 model

I use the same model architecture with a convolutional layer on top of the LSTM layer.

In [ ]:

```
model_glove = Sequential()
```

```
model_glove.add(Embedding(vocabulary_size, 100, input_length=50, weights=[embedding_matrix], trainable=Fals
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'])
```

In [ ]:

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

```
Epoch 1/3
94/94 [==============================] - 3s 14ms/step - loss: 0.5872 - accuracy: 0.7040 - val_loss: 0.5706
- val_accuracy: 0.7035
Epoch 2/3
94/94 [==============================] - 1s 10ms/step - loss: 0.5095 - accuracy: 0.7563 - val_loss: 0.4965
- val_accuracy: 0.7750
Epoch 3/3
94/94 [==============================] - 1s 10ms/step - loss: 0.4850 - accuracy: 0.7733 - val_loss: 0.4907
- val_accuracy: 0.7645
```

Out[ ]:

```
<keras.callbacks.History at 0x7f20f18ccf70>
```

Word embedding visialization

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 in a great way, but here I just want to visualize the word relationship.

Get embedding weights from glove

In [ ]:

```
lstm_embds = model_lstm.layers[0].get_weights()[0]
```

In [ ]:

```
conv_embds = model_conv.layers[0].get_weights()[0]
```

In [ ]:

```
glove_emds = model_glove.layers[0].get_weights()[0]
```

Get word list

In [ ]:

```
word_list = []
for word, i in tokenizer.word_index.items():
    word_list.append(word)
```

Scatter plot of first two components of TSNE

In [ ]:

```
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)
```

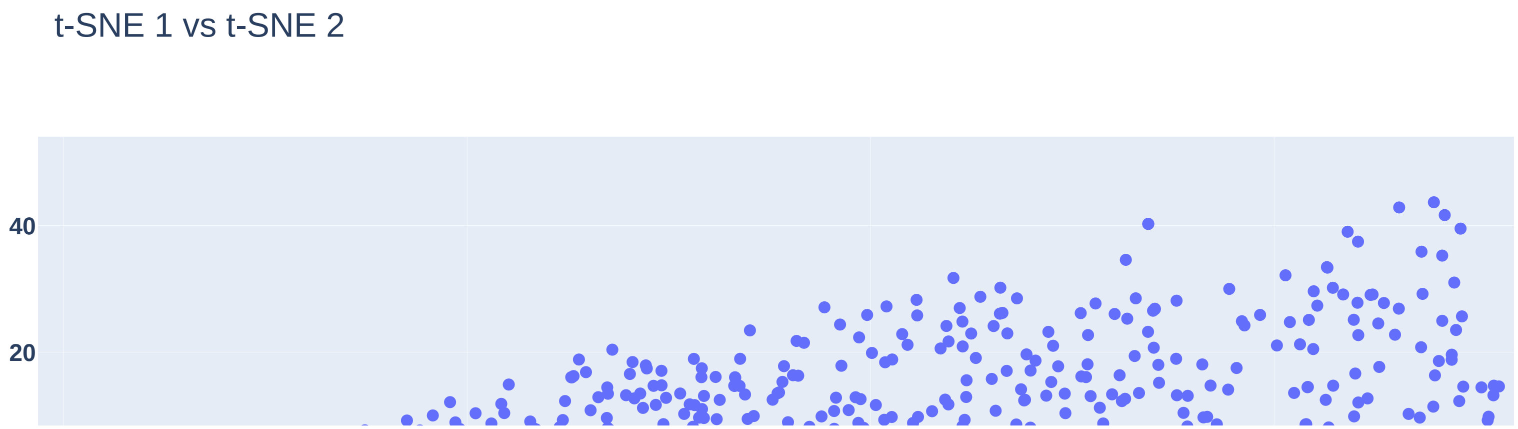
1. LSTM

In [ ]:

```
number_of_words = 2000
lstm_tsne_embds = TSNE(n_components=2).fit_transform(lstm_embds)
```

In [ ]:

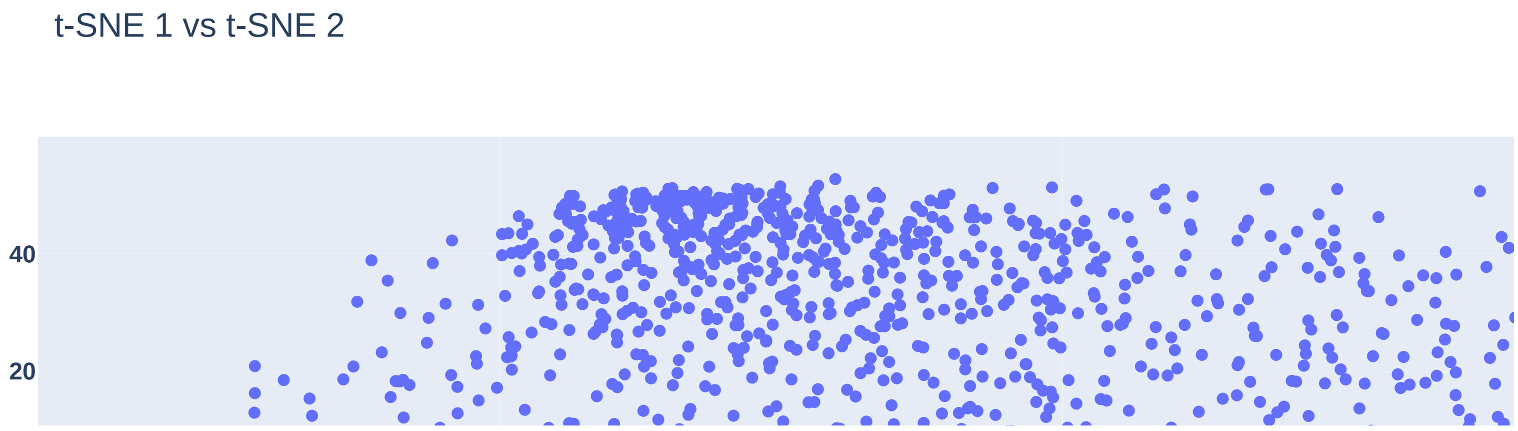
```
plot_words(lstm_tsne_embds, 0, number_of_words, 1)
```



2. CNN + LSTM

```
In [ ]: conv_tsne_embds = TSNE(n_components=2).fit_transform(conv_embds)
```

```
In [ ]: plot_words(conv_tsne_embds, 0, number_of_words, 1)
```



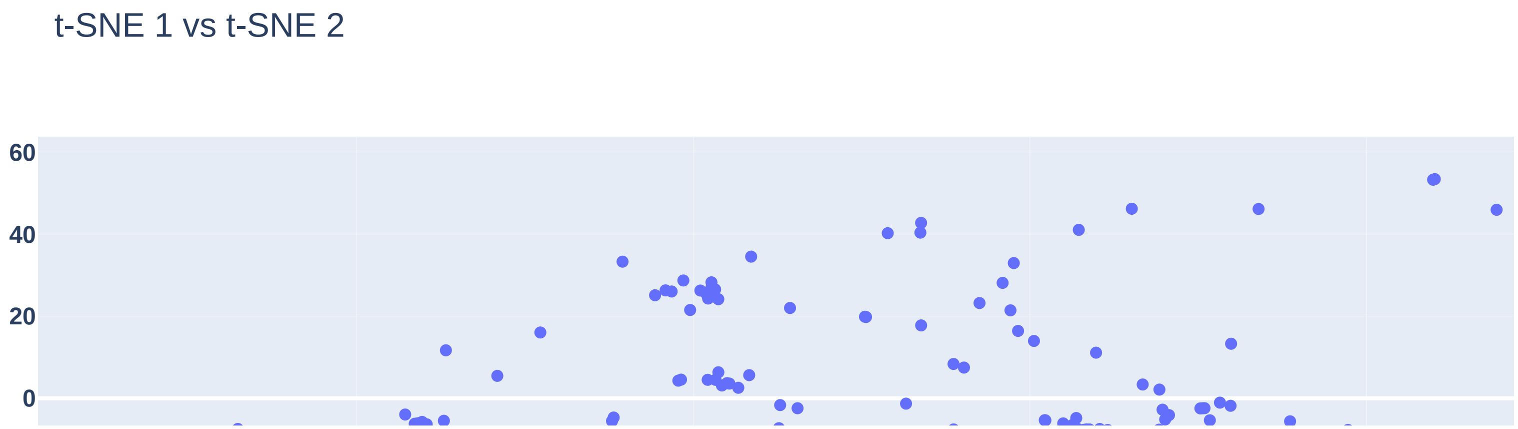
3. Glove

In [ ]:

```
glove_tsne_embds = TSNE(n_components=2).fit_transform(glove_emds)
```

In [ ]:

```
plot_words(glove_tsne_embds, 0, number_of_words, 1)
```


t-SNE 1 vs t-SNE 2

Word Embeddings from Word2Vec

In this subsection, I use word2vec to create word embeddings from the review comments. Word2vec is one algorithm for learning a word embedding from a text corpus.

In [ ]:

```
%%capture
%pip install gensim
```

In [ ]:
```
from gensim.models import Word2Vec
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/deepcode/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[ ]:
```
True
```

Tokenize the reviews coments.

In [ ]:
```
df['tokenized'] = df.apply(lambda row : nltk.word_tokenize(row['text']), axis=1)
```

In [ ]:
```
df.head()
```

Out[ ]:

	stars	text	tokenized
0	3	decid eat here awar go take hour begin end tri...	[decid, eat, here, awar, go, take, hour, begin...
1	5	i have taken lot spin class year noth compar c...	[i, have, taken, lot, spin, class, year, noth,...
2	3	famili diner buffet eclect assort : larg chick...	[famili, diner, buffet, eclect, assort, :, lar...
3	5	wow ! yummi differ delici favorit lamb curri k...	[wow, !, yummi, differ, delici, favorit, lamb,...
4	4	cute interior owner gave tour upcom patio roof...	[cute, interior, owner, gave, tour, upcom, pat...

Train word2vec model

In [ ]:
```
model_w2v = Word2Vec(df['tokenized'], vector_size=100)
```

In [ ]:
```
X = model_w2v[model_w2v.wv]
```

Plot Word Vectors Using PCA

In [ ]:
```
from sklearn.decomposition import TruncatedSVD
```

In [ ]:
```
tsvd = TruncatedSVD(n_components=5, n_iter=10)
result = tsvd.fit_transform(model_w2v.wv.vectors)
```

```
In [ ]: result.shape
```

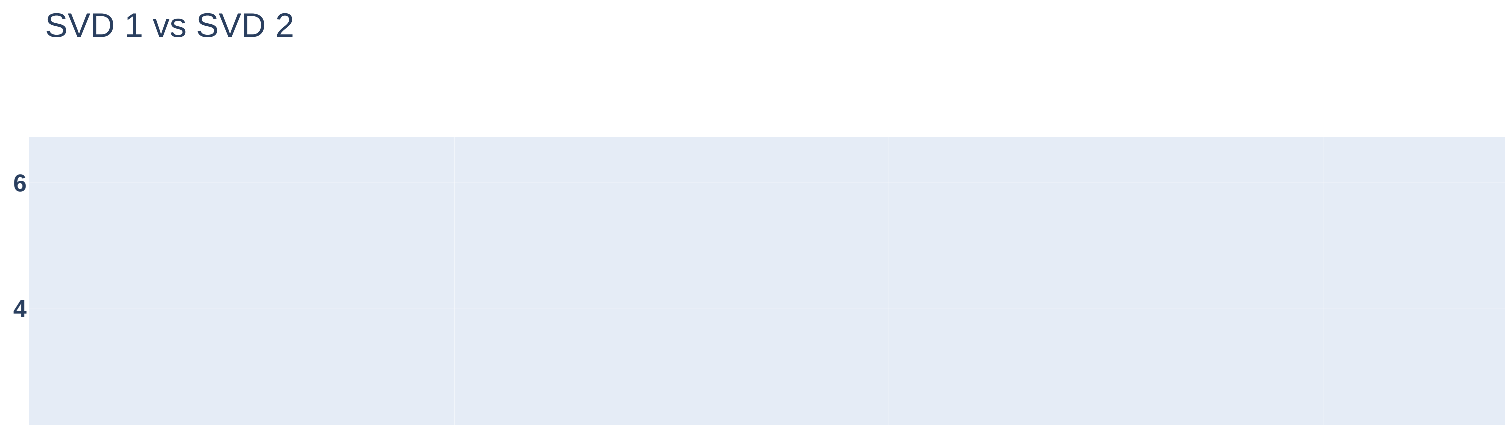
```
Out[ ]: (4114, 5)
```

```
In [ ]: tsvd_word_list = []
        #words = list(model_w2v.wv)
        words = model_w2v.wv.index_to_key
        for i, word in enumerate(words):
            tsvd_word_list.append(word)

        trace = go.Scatter(
            x = result[0:number_of_words, 0],
            y = result[0:number_of_words, 1],
            mode = 'markers',
            text= tsvd_word_list[0:number_of_words]
        )

        layout = dict(title= 'SVD 1 vs SVD 2',
                      yaxis = dict(title='SVD 2'),
                      xaxis = dict(title='SVD 1'),
                      hovermode= 'closest')

        fig = dict(data = [trace], layout= layout)
        py.iplot(fig)
```



In [ ]: