1. **Reading the text file content and creating a list.**

def get\_phrase\_list():

content = list(map(lambda x: x.strip("\n"),open("phrase\_file.txt","r").readlines()))

return content

1. **Reading and cleaning the csv file data by iterating over its rows**.

def read\_csv\_file(csv\_file):

sentence\_list = []

data\_Frame = pd.read\_csv(csv\_file)

for index , row in data\_Frame.iterrows():

sent = row["Before-Cleaning-Sentences"]

sent = sent.replace('\n',' ').replace('\t',' ')

sent = re.sub( '\s+', ' ', sent).strip()

sent\_list = sent\_tokenize(sent)

for new\_sent in sent\_list:

punctuation\_character = '"#%&\'()\*+,/<=>[\\]^\_`{|}~'

translator = str.maketrans('', '',punctuation\_character)

new\_sent = new\_sent.translate(translator)

sentence\_list.append(new\_sent)

return sentence\_list

1. **Creating sqllite db and and appending the data in it.**

def sql\_content():

"""

Get finance document corpus...

"""

sql\_content = []

conn = sqlite3.connect('C:\\Users\\G753903\\Downloads\\finance\_corpus.db')

cur = conn.cursor()

all\_tables = cur.execute("SELECT name FROM sqlite\_master WHERE type='table';").fetchall()

for table in all\_tables:

rows = cur.execute("SELECT \* FROM %s " %(table[0])).fetchall()

for row in rows:

index, label , sent1 , sent2 = row

sql\_word\_tokenization1 = word\_tokenize(sent1)

sql\_word\_tokenization2 = word\_tokenize(sent2)

total\_word\_tokenization = sql\_word\_tokenization1 + sql\_word\_tokenization2

row\_words = []

for word in total\_word\_tokenization:

if len(word) >3 and word not in stop\_words:

row\_words.append(word)

sql\_content.append(row\_words)

return sql\_content

1. Training the word2vec model by reading the content from a csv file and saving its output in a pickle file.

def train\_word2\_vec():

"""

this function train word2vec model on 10-K document

"""

new\_sentence\_list = []

content = read\_csv\_file()

for line in content:

line = line.strip('\n').strip().lower()

line\_tagged = TextBlob(line).tags

new\_filtered\_token = []

for word\_tagged in line\_tagged:

word , tag = word\_tagged

if word not in stop\_words and len(word)>3:

new\_filtered\_token.append(word)

if new\_filtered\_token:

new\_sentence\_list.append(new\_filtered\_token)

print ("----------Training for Word2vec Model ----------")

model = Word2Vec(new\_sentence\_list,window=7, min\_count =1,size=128,sg=0)

words = list(model.wv.vocab)

print ("----------Load Word 2 Vec Unigram model ------")

with open('word2\_vec.pickle', 'wb') as handle:

pickle.dump(model, handle, protocol=pickle.HIGHEST\_PROTOCOL)

print ("Model Load Successfully....")

if \_\_name\_\_ == "\_\_main\_\_":

train\_word2\_vec()

1. **Read Config file**

import configparser

def read\_configfile(configfile="config.ini"):

Config = configparser.ConfigParser()

Config.read(configfile)

Input\_corpus\_path = Config.get('SectionTwo','input\_10k')

return Input\_corpus\_path