import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import re # For preprocessing

import pandas as pd # For data handling

from time import time # To time our operations

from collections import defaultdict # For word frequency

import spacy # For preprocessing

import logging # Setting up the loggings to monitor gensim

logging.basicConfig(format="%(levelname)s - %(asctime)s: %(message)s", datefmt= '%H:%M:%S', level=logging.INFO)

#automatically detect common phrases (bigrams) from a list of sentences.

from gensim.models.phrases import Phrases, Phraser

from gensim.models import Word2Vec

df = pd.read\_csv('../input/rickmorty-scripts/RickAndMortyScripts.csv')

df.shape

#to check missing values

df.isnull().sum()

#preprocess the data

brief\_cleaning = (re.sub("[^A-Za-z']+", ' ', str(row)).lower() for row in df['line'])

sent = [row.split() for row in df\_clean['clean']]

phrases = Phrases(sent, min\_count=30, progress\_per=10000)

bigram = Phraser(phrases)

print(bigram)

pre\_sentences = bigram[sent]

#build the model

w2v\_model = Word2Vec(min\_count=20,

window=2,

vector\_size=300,

sample=6e-5,

alpha=0.03,

min\_alpha=0.0007,

negative=20,

)

t = time()

w2v\_model.build\_vocab(sentences, progress\_per=10000)

print('Time to build vocab: {} mins'.format(round((time() - t) / 60, 2)))

t = time()

w2v\_model.train(sentences, total\_examples=w2v\_model.corpus\_count, epochs=30, report\_delay=1)

print('Time to train the model: {} mins'.format(round((time() - t) / 60, 2)))

# Most similar words

w2v\_model.wv.most\_similar(positive=["rick"])

# Most similar words

w2v\_model.wv.most\_similar('morty')

w2v\_model.wv.save\_word2vec\_format('output.txt')