Text Analytics HW2

1. Lab
I’ve built three word2vec models in which I only changed three parameters and kept all others as default values:

sg: defines the training algorithm. By default (sg=0), CBOW is used. Otherwise (sg=1), skip-gram is employed.
workers: use this many worker threads to train the model
window: maximum distance between the current and predicted word within a sentence.

Model 1: sg=0, window=5, workers=5
Model 2: sg=0, window=50, workers=5
Model 3: sg=1, window=50, workers=5

Code please refer to:
https://github.com/MSIA/zzm7646_msia_text_analytics_2020/blob/homework2/HW2.py

And normalized text output:
https://github.com/MSIA/zzm7646_msia_text_analytics_2020/blob/homework2/output.txt

I chose 10 words and manually reviewing their closest neighbors in terms of cosine similarity to compare the embedding of the three models. Please the full results at:
https://github.com/MSIA/zzm7646_msia_text_analytics_2020/blob/homework2/HW2.ipynb

In general, all three models performed well that the top 5 closest words looked reasonable to me. For some words, I found that skip-gram gave better results than CBOW, see the “chicago” example below:

Model1

chicago
(nyc, 0.8111834526062012)
(hawaii, 0.7611942291259766)
(seattle, 0.7306854128837585)
(boston, 0.7283211946487427)
(cali, 0.7216947078704834)

Model2

chicago
(nyc, 0.6978552937507629)
(ohio, 0.6540114879608154)
(brooklyn, 0.651384711265564)
(coast, 0.6509166359901428)
(pittsburgh, 0.644057035446167)

Model3

chicago
(giordano, 0.7606273889541626)
(coast, 0.7211610078811646)
(giordanos, 0.7184209823608398)
(pizzerias, 0.7082206606864929)
(pittsburgh, 0.7080279588699341)

2. Comparison of word2vec, Bert, and Elmo

	word2vec	**Bert**	**Elmo**
learning model details	Word2vec can utilize either CBOW or continuous skip-gram to produce a distributed representation of words. In CBOW architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction. In the skip-gram architecture, the model uses the current word to predict the surrounding window of context words and weighs nearby context words more heavily than more distant context words	BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer.	Elmo is a deep contextualized word representation that models both complex characteristics of word use (e.g., syntax and semantics) and how these uses vary across linguistic contexts (i.e., to model polysemy). These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pretrained on a large text corpus.
summary of word context approaches	encodes a word fixed representation into a single vector (order of words in sentences is not considered)	sentences or multiple sentence into a single class vector or multiple kind of contextualized word vectors (position of the word matters)	encodes a word in context into a set of vectors (corresponding to various layers, position of the word matters)
corpus size requirements	large text corpus	small text corpus	large text corpus
computational requirements	use negative sampling and subsampling to speed up the computation	expensive on cloud computational costs	less expensive on cloud computational costs
implementation	Python: genism	Python: keras_bert, pytorch-pretrained-bert	Python: pytorch-fast-elmo
date of publication	2013	2018	2018
number of google scholar citations	23k	10k	4k