Text Analytics, HW
Louis-Charles Genereux
Github: https://github.com/MSIA/lgo4950_msia_text_analytics_2021/tree/homework2/HW2%20submission

Note to grader:

I have run the lab section of this homework (preprocessing) on two different corpa:

1. Pride and Prejudice by Jane Austen (just 1 book)
2. 20 Newsgroups corpus, training set only (mostly containing emails referencing news articles)

While normalization and tokenization steps are similar in both cases (removing punctuation, converting to lower case, removing digits), there are some subtleties. In the case of the book, I remove chapter names. In the case of the email exchanges, I remove the from/to names and e ail addresses as well as the subject line.

I ran word2vec on both corpa and obtained better results in case 2 (likely due to a considerably larger body of text). Therefore, I will use the Newsgroups corpus for word2vec.

My full methodology can be consulted in the Jupyter. Also, I have converted my full analysis pipeline of the Newsgroup corpus into a .py file. Additionally, I have written a unit test of my preprocessing pipeline in a separate .py file.

My final submission includes: the Jupyter notebook, its pdf version, py files (executable script and unit test), the cleaned corpus (txt) and a csv file studying different model results.

To run the .py file, use the following commands:
Python question_1.py

```
(base) [14:18:51] louisgenereux:HW2 git:(main) $ python question_1.py
- File paths to all corpus elements collected
- All elements of corpus added to list
- All corpus preprocessed and added to text file
- All words of corpus added to list of list
- READY FOR WORD2VEC MODELING!
Created 9 skipgram models with embeddings [50,100,200] and window size [3,5,7]
Created 9 CBOW models with embeddings [50,100,200] and window size [3,5,7]
Nearest neighbors generated for 18 models
Nearest neighbor results saved to csv
```

To run the test, run the following commands:
pytest question_1_test.p

```
(base) [14:36:58] louisgenereux:HW2 git:(main) $ pytest question_1_test.py
============================= test session starts ==============================
platform darwin -- Python 3.8.3, pytest-5.4.2, py-1.10.0, pluggy-0.13.1
rootdir: /Users/louisgenereux/Desktop/Term 4/Text_analytics/HW2
collected 1 item

question_1_test.py .

============================== 1 passed in 2.06s ===============================
(base) [14:40:32] louisgenereux:HW2 git:(main) $
```

Question 1:

I have decided to experiment with 3 different word2vec parameters:

* **Model**: skipgram or CBOW
* **Number of embeddings:** the number of dimensions used to convert each word to vectors. Its standard value is 100, I also test 50 and 200
* **Window size:** the total number of words before and after a given word that are considered in modelling. Its standard value is 5, I also test 3 and 7

Experimenting with different combinations of parameters generates 2*3*3 = 18 model variations. I evaluate them by generating the 3 closest neighbours from a cosine similarity perspective for 10 words: ['government', 'army', 'happy', 'anger', 'pride', 'wealth', 'overwhelming', 'education','family','computer']. I then compare the outputs across these models.

I started by comparing the standard **skipgram** vs **CBOW** models (with 100 embeddings and window size of 5). While nearest neighbours generated by both models were generally similar across words, I noticed that **skipgram** generated slightly higher quality (more general and descriptive) nearest neighbours. In the basic **skipgram**, the 3 nearest neighbours for 'government' are [soviet, fascist, genocide] as opposed to [soviet, genocide, greek] for **CBOW**. If I did not know what a government is, I would find more context/topics to explore under **skipgram** than under **CBOW** ('greek' would not help me make a dictionary search that can help me understand the concept of government as much as 'fascist'). Another example where basic **skipgram** outperforms **CBOW** is with the word 'pride', generating [husband, heart, mother] rather than [husband, heart, choosing]; 'choosing' in this case is an apparent error from CBOW.

I generally found that higher embeddings numbers provide more subtleties in nearest neighbours. Comparing nearest neighbours for of word "wealth" through **skipgram** models with 50 vs 100 embeddings (and window size of 5), I notice that 100 embeddings' nearest neighbours ([element, feeding, individuality]) alludes to the theme of individuality, which was not raised with only 50 embeddings ([element, allegation, feeding]).

When experimenting with the window size setting, the small changes in parameter value I have tested do not generate materially different results (other than longer training time when window size is increased).

Question 2:

	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding	Distributed Representations of Words and Phrases and their Compositionality
Date of publication	* 11 Oct 2018 * (references other paper)	* 2013 * (referenced by BERT paper)
Number of citations	23,347 citations	29,944 citations
Summary of findings	This paper builds on the Elmo framework (which evaluated word context from a bi-directional as opposed to just left to right) This paper introuces: * An **architectural improvement** versus Elmo (which concatenated results from LTSMs after they had been trained on only one direction. Bert uses a bidirectional transformer. * **Masked language model**: where random words are masked (with the objective of predicting the word given the left and right context) * **Next sentence prediction**: (in training, 50% of the actual next sentences are maintained, 50% are randomized) * Experiments demonstrated that both bi-directionality and next sentence prediction improves predictions across tasks (e.g., determine if next sentence is entailment/contradiction/ neutral vs first, compare equivalence of sentences),	This paper presents an extension of the Skip-gram model proposed by Mikolov, which made great strides because it did not rely on dense matrix multiplications. This paper introduces: * **Subsampling of common words** like “a”, “the” etc. (can accelerate training, improve representation of less common words) * Improvement of softmax activation function with **hierarchical softmax** (less nodes evaluated for probability distribution) * **NEG**: an improvement over hierarchical softmax (learning only from high quality vectors, essentially updating weights only for subsets of words) * Approach to identifying **idiomatic phrases** (e.g., “Chicago Tribune” vs just “Chicago” and Tribune) Additionally, this paper shows that **vector additions/subtractions** can result in identifying the most appropriate word
New contributions to NLP field	* New model architecture (BERT representations are jointly conditioned on both left and right context in all layers) * Next sentence prediction (can provide great value in the context of question answering) * Adaptative learning rate (when compared to GPT models)	* Subsampling of frequent words (increasing speed and improving significance of infrequent words * Negative sampling (NEG) and a formula for identifying negative samples * Formation of idiomatic phrases by gradually decreasing threshold for acceptance of phrase
Data required	~ 3B words (from Books and wikipedia) * Cannot shuffle sentences (because next sentence is required in training)	~30 billion words (faster training despite larger corpus, because of Skip-gram architecture)