CIS 365 NLP Assignment

Dr. Denton Bobeldyk

For this lab/assignment you will be using Gensim's Word2Vec. Begin my installing the package using pip install:

pip install gensim

Your output should look something like this:

```
nlp $pip install gensim
Collecting gensim
  Downloading gensim-4.3.3-cp39-cp39-macosx_10_9_x86_64.whl (24.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.1/24.1 MB 17.0 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0,>=1.18.5 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from gensim) (1.22.0)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from gensim) (1.11.3)
Requirement already satisfied: smart-open>=1.8.1 in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from gensim) (7.0.5)
Requirement already satisfied: wrapt in /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from smart-open>=1.8.1->gensim) (1.15.0)
Installing collected packages: gensim
Successfully installed gensim-4.3.3
```

Include the following lines in your script for this assignment:

```
import gensim
from gensim.models import Word2Vec
from gensim.test.utils import datapath
from gensim.utils import tokenize
from gensim.models.word2vec import PathLineSentences
from gensim.utils import simple_preprocess
import os
import urllib.request
import zipfile
import random
```

Part One

Define a small corpus to start off, this should look like a few sentences:

```
corpus = [
        “This is a sample sentence you fill in”,
        “This is a another sample sentence you fill in”,
        “…”,
        “This is your last sample sentence you fill in”
]
```

Preprocess the corpus using the following command:

```
processed_corpus = [simple_preprocess(doc) for doc in corpus]
```

Next create a Word2Vec model:

```
model = Word2Vec(
        sentences=processed_corpus,
        vector_size=50
        window = 3
        min_count=1
        sg=0
        epochs = 10
)
```

Select a word from your corpus and find the words that are similar to it:

```
print(“Words similar to <your_word>:”)
print(model.wv.most_similar(‘<your_word>’, topn=5))
```

Report the word you selected along with the top 5 results returned in a report. Clearly label what you are reporting.

Select two words that you think should be similar to each other and report which 2 words you selected and what the similarity score was. You can use the ‘print(model.wv.similarity('this', 'fill')’ command to calculate and report this (replacing words ‘this’ and ‘fill’ with your words.

Experiment with some word vector arithmetic. NOTE: you may need to add some sentences to your corpus to get some interesting results for this section.

Your code should look something like:

```
try:
        print(model.wv.most_similar(positive=['this', 'fill'], negative=['is'], topn=1))
```

```
except KeyError:
        print("Error!!")
```

Where ‘this’, ‘fill’, ‘is’ are replaced by the words you had intended.

Part Two
Expand the corpus even more by downloading the Text8 dataset:

Include the following lines in your code:

```
dataset_url = "http://mattmahoney.net/dc/text8.zip"
dataset_path = "text8.zip"
corpus_file = "text8"

if not os.path.exists(corpus_file):
   print("Downloading the Text8 dataset...")
   urllib.request.urlretrieve(dataset_url, dataset_path)
   print("Unzipping the dataset...")
   with zipfile.ZipFile(dataset_path, "r") as zip_ref:
      zip_ref.extractall()
   print("Dataset downloaded and extracted.")
```

You can load and process the dataset using:

```
sentences=PathLineSentences(corpus_file)
```

Train the model using Word2Vec. Modify your existing implementation to accommodate for the new data.

Attempt to repeat the experiment we observed in the video using:

```
print("\nVector arithmetic: king - man + woman =")
try:
   print(model.wv.most_similar(positive=['king', 'woman'], negative=['man'], topn=1))
except KeyError:
   print("Word not in vocabulary!")
```

Report the result of your experiment.

The following code lets you visualize some words in a 2-dimensional space using PCA to reduce the number of feature vectors to 2. Select several words that may be interesting to report on, include those in the word list below and report on your observations (screenshot the output).

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

```
words = ['king', 'queen', 'man', 'woman', 'child', 'world', 'science', 'data', 'art', 'power'] #choose your own words here
word_vectors = [model.wv[word] for word in words if word in model.wv]

pca = PCA(n_components=2)
reduced_vectors = pca.fit_transform(word_vectors)

plt.figure(figsize=(10, 8))
for i, word in enumerate(words):
    if word in model.wv:
        plt.scatter(reduced_vectors[i, 0], reduced_vectors[i, 1])
        plt.text(reduced_vectors[i, 0] + 0.02, reduced_vectors[i, 1] + 0.02, word)
plt.title("Word2Vec Word Embeddings Visualization")
plt.show()
```

Part Three

For this part of the assignment, you will attempt to generate a meaningful sentence. In order to do so, a simple algorithm could be:

Select a seed word
Randomly select a word near the seed word
Continue until you reach a predefined length.

Experiment with some different algorithms. The completion criteria for this assignment is to generate any sentence however there is a chance for extra credit.

Extra Credit: The person/group that creates the best algorithm (as voted by the students in attendance at the final exam) will receive extra credit. How much extra credit will be determined by Dr. Bobeldyk based on how advanced/successful the algorithm is. It may be possible that more than 1 group will receive extra credit if sufficiently advanced algorithms have been provided.

Approved Language: Python

The assignment will be graded based on completion and demonstration of completion.
Hand-in:

1. Word/PDF document containing your 'report'
2. Screenshots of the execution/demonstration
3. Source code used to generate the above (please no zip files).