CDS Language Analytics Assignment #3: Query Expansion With Word Embeddings

Description

This script utilizes word embeddings to find out how many songs by a given artist contain words similar to a given search word.

This code uses CodeCarbon to monitor the environmental effects of running this code. The effects of which can be found in the `out/emissions` folder

Setup

1. Make sure to have python and Git Bash installed!
2. Open a Git Bash terminal and use Git to download the repository:

```
git clone https://github.com/missingusername/cds-lang-git.git
```

3. Navigate to the project folder for this assignment:

```
cd cds-lang-git/assignment3
```

4. Before running the program, you first have to set up a virtual environment with the required dependencies. This can be done by simply running either `bash win_setup.sh` or `bash unix_setup.sh` depending on your system.
5. Before we can run the script, we first need the song data. The data can be found here, where you can download an `archive.zip` file. When downloaded, unzip the folder and place `Spotify Million Song Dataset_exported.csv` in the `/in` folder of the `assignment3` directory.
6. The script uses Argparse to parse commands to the script from the command line.

name	flag	description	required
artist	-a	What artist to search for	*REQUIRED
word	-w	What word to search for similar words for	*REQUIRED

name	flag	description	required
number	-n	How many similar words to find for the search word (default=10)	OPTIONAL
save	-s	whether to save a .csv file of the gathered data	OPTIONAL

To run the program, you can run the appropriate run script, again depending on your system. Here's an example of how to run the script with arguments:

```
bash win_run.sh -a metallica -w god -s
```

or

```
bash unix_run.sh -a abba -w love -s
```

The Code

The code works by first turning the data into a pandas dataframe, and filters it such that only songs by the specified artist remains.
We use gensim to load the model `glove-wiki-gigaword-50`, which is trained on 2B tweets (27B tokens, 1.2M vocabulary)
Then we generate a list of the 10 most similar words to the search word, using gensim "most_similar", as well as the search word itself.
Then, we simply go through every song by that artist, checking if any of the similar words appear in the songs text. We keep track of this in a dictionary, counting how many times each of the similar words occurs.
This all gets converted to its own dataframe, where we keep track of the song, if it contained any of the similar words, and how many times each similar word appeared in the text.
This is the dataframe that you can optionally have saved to the `/out` folder *(some output examples can also be found there)*.
Lastly, you get some stats about the data printed to the terminal, telling you how many of the songs featured similar words.

Takeaways from output

To look at the output of the code, we can use the printout provided at the end:

```
You searched for songs by: abba
Which contained words similar to: love
This returned the following similar words: dream, life, dreams, loves, me, my, mind, loving, wor

Of the 113 total songs by abba:
    106 songs contained words similar to love
    7 songs did not contain words similar to love
    This means 93.81% of abba songs contains words similar to love
```

From this printout we can see that it tells us what artist we searched for (`abba`), and what search word we used (`love`). We also get to see what similar words the models found for the search word (`dream, life, dreams, loves, me, my, mind, loving, wonder, soul, love`). We can also see how many songs that artist has in the dataset (`113`). It then tells us how many of the songs contained any of the similar words (`106`), as well as how many songs didn't (`7`). Lastly, we get the calculated percentage of how many songs contained similar words (`93.81%`).

Limitations & Improvements

The script takes a while to load the gensim model, and it does so every time you run the program. If one wanted to search for multiple artists and words, it might be more efficient to run the script recursively. Here, you could initialize the script and the model, and then enter what artist and word you want to search for, get the results, reset the gathered data, and enter another artist and search word.