CS410 Course Project
Movie Genre Analysis
Arda Bedoyan

Overview:

This program is meant to help users determine the genre of a movie. It works by taking in as input a movie script and then printing out up to three genres that the movie falls under. The code is based on an input document's similarity to documents in an already established collection of documents.

The program can be used to determine the genre of a new movie or, at the very least, determine the top movies that a new script is similar to. This can be used to make interesting observations and help to understand how sometimes very different movies can actually have quite similar scripts. Creative writing and film studies can further find ways to analyze scripts based on this similarity feature.

Getting Started:

The program uses the Gensim toolkit (https://radimrehurek.com/gensim/index.html)

Gensim runs on Linux, Windows and Mac OS X, and should run on any other platform that supports Python and NumPy. The documentation says that Gensim is tested with Python 3.6, 3.7, and 3.8. I used Python 3.8.7 on a Windows computer.

Begin by running `pip install --upgrade gensim` in your terminal. If you encounter any trouble, please check what version of Python you have installed. I encountered some errors installing Gensim when running on Python 3.11, and therefore downloaded Python 3.8. Afterwards, the errors stopped and I was successfully able to install Gensim. Once installation is done, no further setup needs to be done.

How the Code Works:

The program works by first initializing a training set of movie scripts. This training set includes 15 movies across five genres. The five I chose for this project were Action, Comedy, Drama, Romance, and Horror. These genres were chosen so as not to have too many options. Additionally, it can be argued that most other genres are combinations or sub-genres of these five genres. The training set of movies included the movies in the table below.

Genre	Movie 1	Movie 2	Movie 3

Action	The Terminator	Die Hard	The Dark Knight
Comedy	Dumb and Dumber	Bean	Bridesmaids
Drama	Forrest Gump	Good Will Hunting	The Godfather
Romance	Titanic	Pretty Woman	Notting Hill
Horror	Get Out	The Conjuring	The Shining

I chose these 15 movies because I thought they exemplified their corresponding genre well. Additionally, not every movie script can be found online, so I had to eliminate some chosen movies whose scripts could not be found. The scripts for these movies in addition to movies that I ran through the program to determine their genres were saved into text files. If you would like to run the program for a movie that does not have a corresponding text file saved in this repository, please copy and paste the script into a text file and name the file the same name as you will enter into the program. If you notice that the program gives you some errors when trying to read in the text files, please change the encoding type of the text file from UTF-8 to ANSI.

Please import the necessary libraries below.

```
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
from collections import defaultdict
from gensim import corpora
from gensim import models
from gensim import similarities
from pathlib import Path
```

The genre_analysis() function is where the analysis truly takes place. It reads in all of the movie script from the training set, removes common words, and tokenizes the documents. It then removes words that only occur once. A corpus is created that consists of all of the pre-processed scripts.

The Vector Space Model algorithm used in this code is Latent Semantic Indexing (LSI), sometimes referred to as Latent Semantic Analysis (LSA). The LSI code I implemented was based off of the code from the Gensim documentation (https://tinyurl.com/2vrvvejf). LSI transforms documents from either a bag-of-words or Tf-Idf-weighted space, into a latent space of a lower dimensionality. According to Gensim documentation, dimensionality of 200–500 is recommended as the gold standard. I chose to use 300, which worked equally well as 200, 400, and 500. However 300 worked much better than a smaller number of features, such as 2-5.

After the model is run, the results are sorted and the genres corresponding to the top three movies most similar to the inputted movie script are gathered. After de-duplicating any repeated genres, the program prints out the results of the genre analysis and adds the new genres to not only the training set but also a new dictionary of results that contains the movie title and its top genres.

Evaluation:
During the course of this project, I learned just how difficult it is to determine the genre of a movie. Genres are subjective and what one person may classify as a drama, another might classify as a romance film. Furthermore, there does not exist a single movie that is just one genre completely. Movies are a blend of many genres, many of which were not included in this project. Genres also have sub-genres, so two horror movies might not be deemed similar at all. Whereas one movie can be a psychological horror, another can be a slasher horror film and share little in common with the former.

Therefore, the program will only work as well as the training set. I used a relatively small training set of only 15 movies. For a program that can be used more widely, the training set should be much larger and many more genres should be included. This takes great human effort to initialize the program, but should ultimately lead to better results and less human correction in the future.

Due to the nature of such a program, the computer can make mistakes. Therefore, I added a feature that allows the human user to verify whether the results of the program are in line with what they expected. If not, then they can manually enter the top two genres that they associate with the chosen movie and update the movie results that way.

After the code was initially completed, I ran 10 additional movie scripts to check that the results were in line with what I expected. After changing the number of features to 300, I was satisfied that the program was generally getting the main genre of an inputted movie correct. For example, it was correctly classifying *The Dark Knight Rises* as a drama and an action movie, and it classified *Ghost* as a romance and a drama. However, certain movies were incorrectly classified under the horror genre, such as *Legally Blonde* and *The Hangover*. I believe this has to do with the difficulty of classifying a horror script for a computer. This is why the option to update the computer results existed. Again, more scripts as part of the training set would improve the performance of the program.

Ultimately, the program did what I expected it to. I took as input a movie script and printed out its top genres based on that script's similarity to other movies. The program

does not perform perfectly, but as with all new endeavors, it will take time and various iterations to improve and advance.

Test Set	Expected Genre	Output num_topics = 300		
The Dark Knight Rises	Action	Action	Drama	Romance
Kill Bill	Action	Romance	Drama	
Legally Blonde	Comedy	Horror	Action	Romance
The Hangover	Comedy	Horror	Drama	Action
My Best Friend's Wedding	Romance	Drama	Comedy	
Ghost	Romance	Romance	Drama	
The Exorcist	Horror	Horror	Drama	Romance
A Quiet Place	Horror	Romance	Horror	
The Shawshank Redemption	Drama	Horror	Action	Drama
Schindler's List	Drama	Action	Horror	

Where to Go From Here:
I had originally hoped to be able to create a website that would act as a movie genre generator and script database. As I researched different methods to approach my project, I realized that time restrictions would not permit me to be able to create the front-facing application that I had initially envisioned. However, I think this remains as an interesting next step and may reapproach the topic as a side project in the future.

Team Contributions:
This was completed as a solo project.

Sources:
https://radimrehurek.com/gensim/auto_examples/core/run_topics_and_transformations.html#sphx-glr-auto-examples-core-run-topics-and-transformations-py

https://radimrehurek.com/gensim/auto_examples/core/run_similarity_queries.html#sphx-glr-auto-examples-core-run-similarity-queries-py

https://en.wikipedia.org/wiki/Latent_semantic_analysis#Latent_semantic_indexing