Below is a link to the Google Colab file with the code to for steps 2-5 of this assignment as the Google News model is too large to run in IDLE:
https://colab.research.google.com/drive/1-9dThpK8Urud3zO3Ix1WqXTQayjmvXGX?usp=sharing .
The text files (wordsim, and movie reviews) need to be temporarily uploaded to Google Colab to run them. Those can be found here: https://drive.google.com/drive/folders/1z03PT4vaiD6H-5spJduPoFpSJLf_wolS?usp=sharing .

For step one of the assignment, I used the 'Santa Barbara Corpus of Spoken American English' found at https://www.linguistics.ucsb.edu/research/santa-barbara-corpus. I set up my model to run through each line of each document within the folder for training. I validated the model by running similarity queries. I printed the top 5 most similar words to "king." The words scored as most similar to "king" were "liver," "changing" and "slept." Parts of words that were scored as being in the top 5 most similar words to "king" were "fe" and "ki." These partial words could indicate an error in my method of training the model, or that words were broken down further into groups of letters when training the model. I also looked at the similarity scores of "king" to the words "queen," "prince," "horse," "car," "cake," and "man." The high similarity scores of "king" to "queen" (0.77) and "prince" (0.70) were expected, but "man" having a lower similarity score (0.08) compared to the scores for "cake" (0.53), "car" (0.48) and "horse" (0.21).

For step two of this assignment, I was not able to load the pre-trained Google News model using IDLE because my machine kept running out of memory. I was able to load the pre-trained model using Google Colab. To test eh model, I created a list of words that I associate with each of the following categories: people, places, and colors. From there, I created a function to run through the list of words in a category and find words similar to the category's words using the "most_similar" method from the Word2Vec model. This then printed a list of words that the model identified as similar to the category's words. I ran each category's list of words through this function. This helped to confirm the validity of the model, especially because some of the words that I had identified as being in the same category were listed as being similar by the Word2Vec model. Word2Vec considers these words similar because they have a high similarity score. This score is calculated by finding the cosine of the angle between the vectors of the words. Words with a cosine similarity score closer to 1 are more similar than words with a cosine similarity score closer to -1.

For step 3, I got a Spearman's rank correlation coefficient of about 0.7717. This means that there is a strong positive correlation between the human-annotated similarities of word pairs in the 'wordsim_similarity_goldstandard.txt' file and the similarities calculated by the large Word2Vec model. The model captures a large amount of the similarities quite accurately (as defined by the human-annotated similarities), but not all of them.

For step 4, I tested analogies using a for loop than ran through a list of analogies that I created. It displayed the word that the Word2Vec model got versus the word I expected the analogy to result in. Below are the results:

1. king + man + woman = teenage_girl (Expected: queen)
2. prince + man + princess = woman (Expected: woman)
3. Paris + France + Italy = Italians (Expected: Rome)
4. France + French + Italy = Italian (Expected: Italian)
5. dog + paw + human = employee_Laura_Althouse (Expected: hand)
6. good + bad + happy = unhappy (Expected: sad)
7. food + ate + drink = drank (Expected: drank)
8. nose + sneeze + mouth = sneezing (Expected: cough)
9. nose + smell + tongue = aroma (Expected: hear)

Only four of the analogies gave the expected result (analogies 2, 4, 6, and 7). The rest were quite far off from what I had expected.

Using Google Colab (as I did in steps 2-4 due to my laptop's memory limitations), I built a text classifier for the movie reviews dataset. I used the same method of preprocessing my data as I did in homework 3 (removing stop words, removing punctuation etc.) as well as tokenizing the text and making all the letters lowercase. I used the pre-trained Google News model to retrieve the word embeddings for each word in the reviews. From there, I split the data into training, development, and test sets. After training a logistic regression classifier on the training data, I got an accuracy of 76.94% on predicting the development data. After tuning the hyperparameters, I found that the default hyperparameters resulted in the highest accuracy. When I ran the classifier on the test data, I got an accuracy of 79.50%. This is slighter higher than the accuracy that I got using the Count Vectorizer in homework 3 (77.12%).