1 Introduction [5 points]

- Group members: Philippe des Boscs, Zane Reeves, Julian Peres
- Colab link:
  Initial Visualizations
  Off implementation using off-the-shelf package
  SVD implementation derived from HW5 (no bias)
  SVD implementation with bias
- Piazza link: Piazza
- Division of labor: We, once again, adopted a split-and-meet strategy throughout the project to evenly and most effectively divide the workload amongst the members of our group.

  Sprint 0: Our strategy was to read the instructions on our own before actively starting the project to familiarize ourselves with the assignment and have the chance to individually brainstorm ideas.

  Sprint 1: Our group began with an initial in-person team meeting to distribute the work evenly, set up a progress timeline, and put together our different implementations of our SVD model from HW5. We were all on the same page regarding the different steps of the algorithm and agreed on a common version to set as our basis. Additionally, we derived the necessary gradients to later implement our SVD model implementation with the incorporation of a bias term.

  Sprint 2: Zane began by tackling the data processing necessary to extract the different entries for each initial visualization linked to the MovieLens Dataset. Philippe then used the data processing code to implement histograms using matplotlib for the required graphs. Some small modifications were needed in the code in relation to NumPy arrays and these changes were implemented.

  Sprint 3: Philippe and Zane began working on the SVD models on Saturday. Zane focused primarily on the implementation derived from HW5 while Philippe focused on the off-the-shelf implementation. Julian completed the code and data processing necessary for the final visualization.

  Sprint 4: Once done with the HW5 implementation, Zane began tackling the SVD implementation with bias based on the same format used for HW5 and using the expressions we derived as a group. Once done with the visualization code, Julian joined in on the off-the-shelf implementation of the SVD with Philippe to retrieve the necessary information from the trained SVD and graph it.

  Sprint 5: Philippe and Zane compared the first sets of visualizations and everyone worked to interpret the best results and finding testing strategies for our findings. Julian created visualizations for the off-the-shelf implementation. Philippe and Zane prepared the Piazza post.

  Victory lap: Final meeting of the group to reflect on performance, visualization comparisons, develop constructive criticism of our strategies, and present our last-minute findings. Philippe mostly worked on the final report summarizing findings and Julian worked on explaining the math.
- Packages used: We used the common packages used so far in the course such as NumPy, math, CSV, and pandas. Additionally, we used surprise for our off-the-shelf implementation of our SVD. Finally, our visualizations were made using (not-so-fancy but really convenient) matplotlib.

2 Basic Visualizations [20 points]

Discussion

Keep in mind that even though we kept a *bin* for *0* in our histograms, the ratings are only between *0.5* and *5* in increments of *0.5*. We felt like it kept a better visual representation of the data.

Analyzing our different visualizations, we notice that on average movies Figure 1 seem to have a low number of *bad reviews* meaning strictly less than 3. This makes sense to the members of our team as first, movies are a form of entertainment and pleasing by nature and second, most viewers base their initial movie selections on their own personal interests, making it a lot more likely that they will enjoy the movies they watch. Additionally, we notice that on average across all movies and all our visualizations (except for the 10 best movies in Figure 3, by a few), the most common rating for a movie is *4*. It makes sense as standards are often very high by nature because of how many great movies most people have watched across their life (siding away from a perfect *5*) but still very enjoyable.

Comparing the ratings made for the 10 most popular movies to the ratings made for the 10 best movies overall, we notice that they have a higher average proportion of high ratings of *5*, *4.5*, *4*. This makes sense as the average rating for each movie makes it part of the top 10 of our entire dataset. To compensate for this higher proportion of higher ratings, the proportion of lower ratings on average for those same movies is lower than for the 10 most popular movies. The overall proportion of *3* ratings is close to 1/3 in Figure 3 compared to Figure 2. An interesting thing we also notice is the difference in total ratings for both categories. There seems to be close to a factor *10* in favor of the 10 most popular movies (Figure 2) for the overall number of reviews than for the 10 best movies (Figure 3). This suggests that for the most part, the most popular movies are not the best movies.

For our genres, we selected Comedy (Figure 4), Mystery (Figure 5), and Children (Figure 6). We notice some significant differences regarding the distribution of the ratings with for example Comedy (Figure 4) and Children (Figure 6) having a higher concentration of ratings of *2.5*, *3* and *3.5* than mystery. We feel like this could make sense as Children and Comedy are broader categories than Mystery. Children's movies (Figure 6) group different genres and only refer to the overall "age" of the intended public but adults watch them as well and certain children may not like certain types of children's movies compared to other types. We can also question whether the ratings were made by children and parents sharing their children's experiences or if they were made by adults who watched those movies, in which case the ratings may be biased towards a lower grade. Regarding Comedy (Figure 4), we feel like it is a very hit-or-miss genre for movies as humor varies a lot depending on the viewers leading to an average rating (*3.5* in our case). On the other hand, because of how narrow the genre of Mystery (Figure 5) is, it tends to have a higher average rating with a very high concentration in the *4* mark and an overall higher proportion of higher grades. This is most likely due to our initial observation that most ratings across all movies are of *4* and the narrowness of the genre tends to have most movies of that category appeal to the population interested in Mystery movies. Overall, our team expected those trends and could make sense of them. We also bore in mind that Comedy (Figure 4) has a lot more reviews than Mystery (Figure 5), which itself also has more reviews than Children (Figure 6). The distribution of Children's movies may therefore not be very accurate overall because of the lack of data but its similarities to the distribution of Comedy suggest that it is still decently accurate.

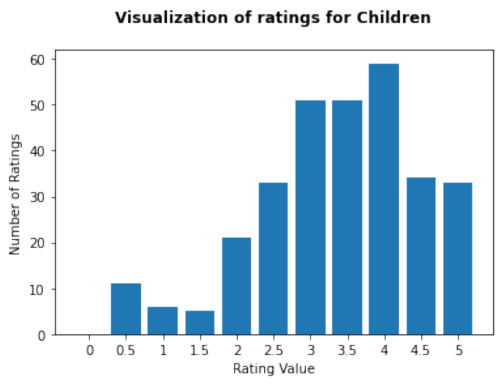

Figure 1: All ratings in the MovieLens Dataset

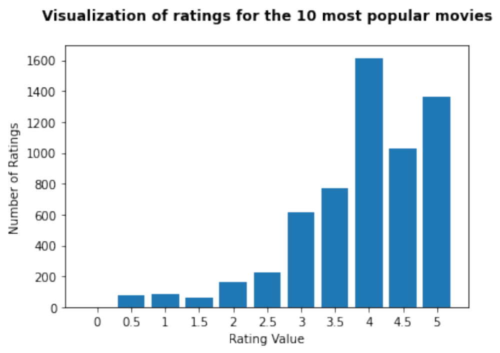

Figure 2: All ratings of the ten most popular movies by the number of ratings.

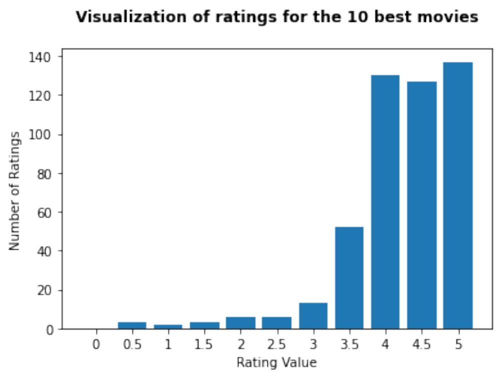

Figure 3: All ratings of the ten best movies by highest average rating.

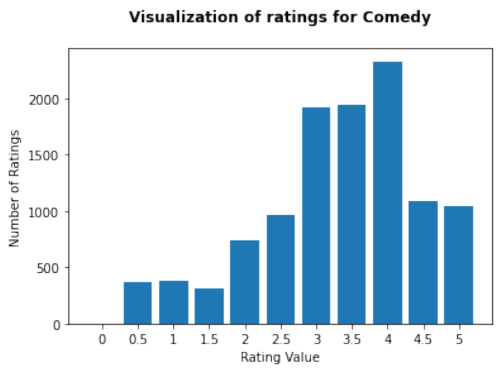

Figure 4: All ratings of movies from the genre **Comedy**.

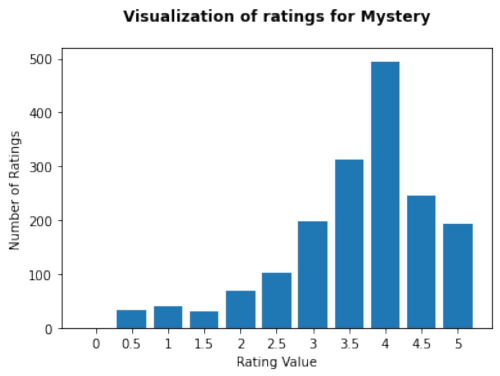

Figure 5: All ratings of movies from the genre **Mystery**.

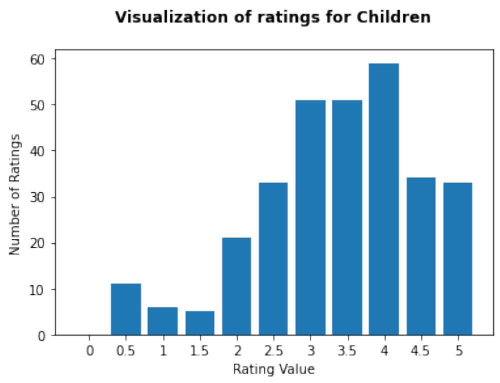

Figure 6: All ratings of movies from the genre **Children**.

3 Matrix Factorization Visualizations [60 points]

Matrix Factorization Methods

Your report should contain a section dedicated to matrix factorization methods. How do each of these methods work? Please be specific and include equations. How do they differ? How did they perform in comparison to one another on the test set? Can these methods' differences explain why they perform differently on the test set?

Without bias terms, the function we attempt to minimize is:

$$\underset{U,V}{\operatorname{argmin}} \frac{\lambda}{2}(\|U\|^2 + \|V\|^2) + \frac{1}{2}\sum_{i,j}(y_{i,j} - u_i^T v_j)^2$$

We determined our hyper-parameters by performing a very rudimentary grid search, starting with our step size (γ) and progressing through to regularization strength (λ), difference threshold, and number of epochs. We found that the following hyper-parameters minimized our error: $\lambda = .1$, epochs = 50, difference threshold = .001 and $\gamma = 3 * 10^{-3.5}$.

With bias terms, we attempt to minimize:

$$\underset{U,V,a,b}{\operatorname{argmin}} \frac{\lambda}{2}(\|U\|^2 + \|V\|^2 + \|a\|^2 + \|b\|^2) + \sum_{(i,j)\in S}((Y_{i,j} - \mu) - (u_i^T v_j + a_i + b_j))^2$$

where a_i and b_j are the bias terms for the user and item respectively, and u_i and v_j are the user and item factors with μ being the global model bias. The predictions are given by $\hat{r}_{ui}$ with the true prediction being r_{ui}. In order to solve for the minimum values, stochastic gradient descent is used for updating weights:

$$\begin{aligned}
a_i &\leftarrow a_i + \gamma(e_{ui} - \lambda a_i)\\
b_j &\leftarrow b_j + \gamma(e_{ui} - \lambda b_j)\\
u_i &\leftarrow u_i + \gamma(e_{ui} \cdot v_j - \lambda u_i)\\
v_j &\leftarrow v_j + \gamma(e_{ui} \cdot u_i - \lambda v_j)
\end{aligned}$$

We determined our hyper-parameters by performing a very rudimentary grid search, starting with our step size (γ) and progressing through to regularization strength (λ), difference threshold, and number of epochs. We found that the following hyper-parameters minimized our error:$\lambda = .1$, epochs = 50, difference threshold = .001 and $\gamma = 3 * 10^{-2}$.

Furthermore, the exact functionality of the algorithm that we implement is the same as the off-the-shelf model.

We chose to use scikit surprises' SGD method for our off the shelf implementation. This method solves the regularized squared error given by:

$$\sum_{r_{ui}\in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

where b_u and b_i are the bias terms for the user and item respectively, and p_u and q_i are the user and item factors. The predictions are given by $\hat{r}_{ui}$ with the true prediction being r_{ui}. In order to solve for the minimal values, stochastic gradient descent is used for updating weights:

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$$

$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u)$$

$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)$$

with $e_{ui} = r_{ui} - \hat{r}_{ui}$ (https://surprise.readthedocs.io/en/stable/matrix_factorization.html). This is run, in our case, for 100 epochs with 20 factors. In order to decrease over-fitting, we combined this algorithm with cross-validation with 10 folds, in order to have the 90/10 train-test split for the data. The algorithm was then fitted to a 90% grouping and the error was computed for the other 10% as validation. It was found that the validation accuracy for the training set hovered around 86% for the dataset. The q_i matrix that was computed from the training would be our V^T matrix.

For all of these implementations, a decomposition of the form $U\Sigma V^T$ was found. The found V matrix was transposed and its SVD was computed: $V = A\Sigma B^T$ (this time using np.linalg.SVD() not the SVD algorithm from surprise). The first two columns of A were taken as predictions since these should correspond to the best projections of the movies to a 2D space. These two columns were multiplied by the original matrix V in order to have a $2 \times N$ sized matrix for the movies (with N being the total number of movies). The column indexed by the movie id can then be taken as the 2-d representation of that movie, which can then be plotted as is seen below.

Performance of Each Model

Method	RMSE Accuracy
HW5 SVD w/o bias	.8247
HW5 SVD w/ bias	.7194
Scikit surprise	.8989

The difference in RMSE can be explained through implicit model training by the grid search process and model optimization through hyperparameter tuning. When performing grid-search we selected hyper-parameters such that our MSE for our validation set is minimized, thus causing implicit training and tuning. Grid-search was used to tune both our HW5 SVD algorithm and our new SVD algorithm with bias. We simply used the same learning rate, number of epochs, and difference threshold for our Off-The-Shelf model as our SVD model with bias as they both rely on the same foundational math. Because of this, we didn't explicitly tune to the implementation that the Off-The-Shelf model uses. Thus, explaining why our Off-The-Shelf has a higher MSE than all other models. Additionally, since the SVD with the bias terms has a stronger ability to fit our data through modeling bias in data, we derive a smaller MSE than our stock HW5 SVD.

Looking at the resulting accuracy, we see that the out-of-the-box implementation had a higher test accuracy than the HW5 implementation with biases. This can likely be attributed to the use of k-fold cross-validation in the off-the-shelf implementation, which serves to have a stronger model that has less over-fitting to the training set, which is why we see that it has a better test accuracy.

Visualizations

Visualization Discussion

We made some pretty interesting observations graphing our different matrix factorization visualizations. The overall trend that we noticed was that most similar movie types/movies from the same series were clustered together. We saw this phenomenon through James Bond movies (Figures 7 13), Avengers movies (Figure 19), Lord of the Ring movies (Figures 8 14 20) etc. We notice some significant patterns along certain axes in our visualizations such as the themes of the movies (love, action, comedy, children) as can be seen in our 10 selected movies from different genres (Figures 7 13 19). one of the main examples is the grouping along the x-axis of movies like Fifty Shades of Grey, Pride, and Prejudice, and Cinderella. We similarly observe the same y-axis-based pattern in some of those graphs based on the maturity level of the movies with a range going from family-friendly movies to movies with very violent and sexual content.

When considering the best-rated movies (Figures 9 15 21) and the most popular ones (Figures 8 14 20), we notice some interesting differences in their visualizations through our different matrix factorization techniques. First, we notice that they are not identical to each other. This makes sense as the clustering and axis factors can differ from one factorization technique to another. We still notice the grouping of similar movies such as, once again the Lord of the Rings which are always close to Kill Bill and Monsters Inc, for an unknown reason. (Figures 8 14 20). The Monsters Inc, Finding Nemo, and Shrek movies in the most popular movies are often not too far away from each other, forming a line but do not show any clustering as significant as the other movie series we have mentioned. It could be explained by the fact that Shrek is not from the same studio (Dreamworks vs Pixar/Disney). For the 10 best movies (Figures 9 15 21), we find the movies to be much more spread out in our visualizations which makes sense to our group as the movies have very different genres, age appropriateness, and producers behind them. Their differences could explain the lack of clustering.

The visualizations of our three genres are not very consistent across our three different SVDs. Considering the Mystery genre (Figures 23 17 11), we notice fewer discrepancies between our HW5 SVD and our bias SVD (Figures **??** 17) than with our SVD implementation using an off-the-shelf package. Our group was not expecting as many discrepancies over our genres between our SVD implementations. A lot of it is associated with randomness and the fact that the movies are more difficulty differentiable by nature as they are all part of the same genre and therefore tend to appeal to the same population. We notice this most significantly in the Children's genre (Figures 24 18 12) where our visualizations are very different creating different clusters. A common observation we make between those graphs is the separation of Shrek 2 from the rest of the movies. There are not a lot of similarities which makes sense, once again because of randomness, and since, through data analysis, we notice that the ratings of the children's movies are very similar one to another. Finally, looking a the comedy genre, we once again have some pretty significant discrepancies, we notice in our HW5 implementation (Figure 10) that the Jump Street movies are together whereas they do not form a cluster in our off-the-shelf implementation (Figure 22). In the case of comedy, HW5 and the bias implementations are actually much more similar to each other which would make sense since they follow similar coding strategies compared to the off-the-shelf and should lead to closer performance. We notice this through the distance of movies like 17 again and 24 Hour Party People located along the edges of our graphs and far from other movies (Figures 10 16). We therefore overall notice some closer similarities between genres with a higher overall number of ratings like Mystery and Comedy than for Children, which has a lot fewer reviews.

Visualization Plots

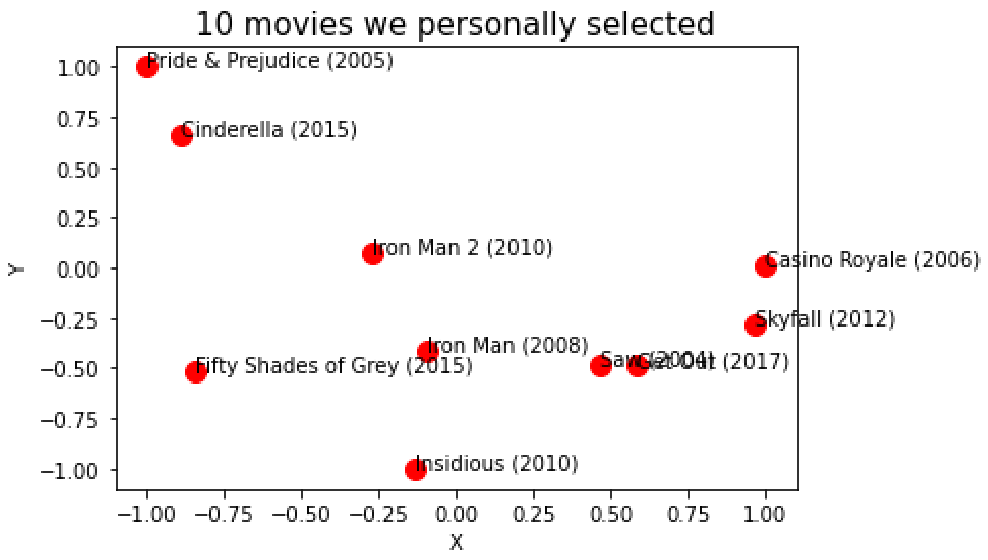

Figure 7: **HW5 A:** Any ten movies of your choice from the MovieLens dataset.

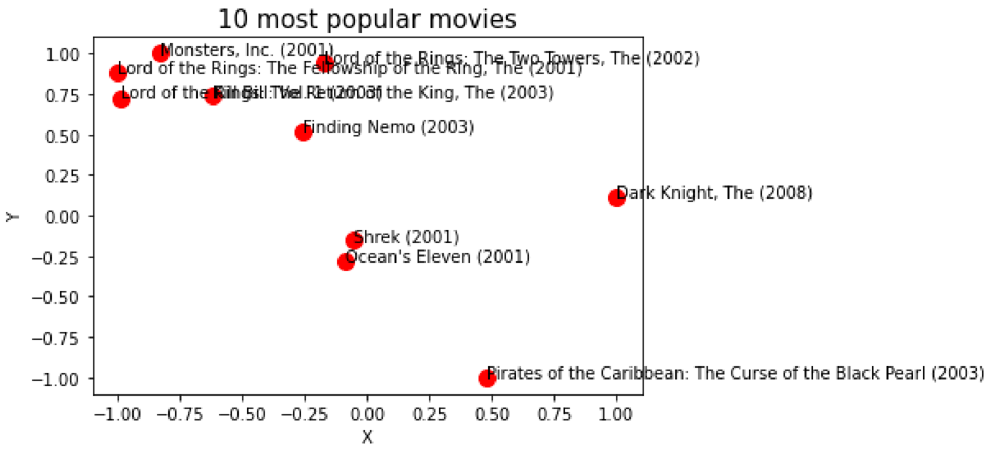

Figure 8: **HW5 B:** The ten most popular movies (movies which have received the most ratings).

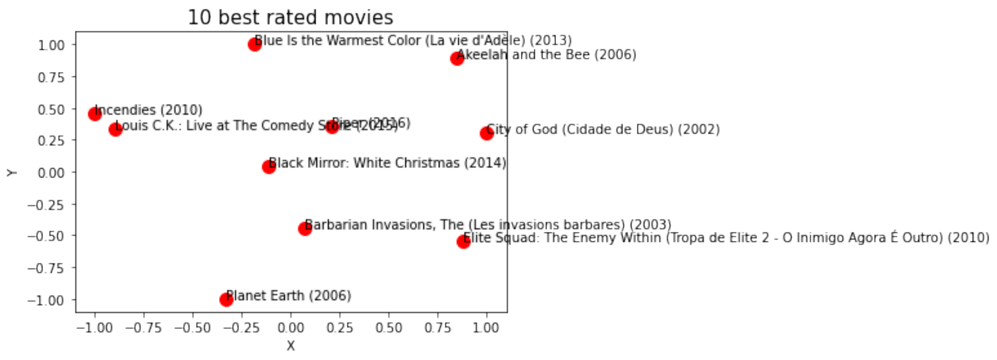

Figure 9: **HW5 C:** The ten best movies (movies with the highest average ratings).

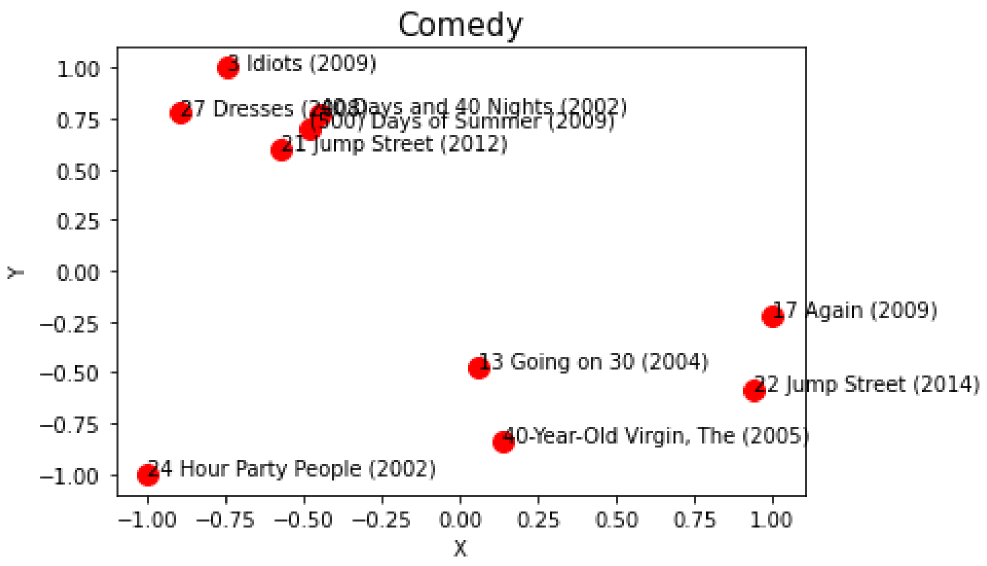

Figure 10: **HW5 D1:** Ten movies from the genre **Comedy**.

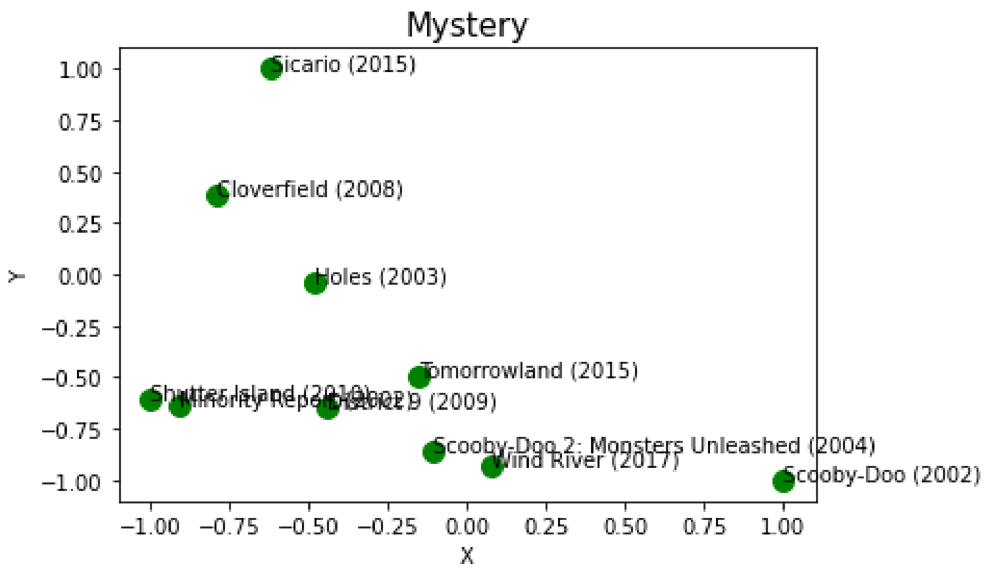

Figure 11: **HW5 D2:** Ten movies from the genre **Mystery**.

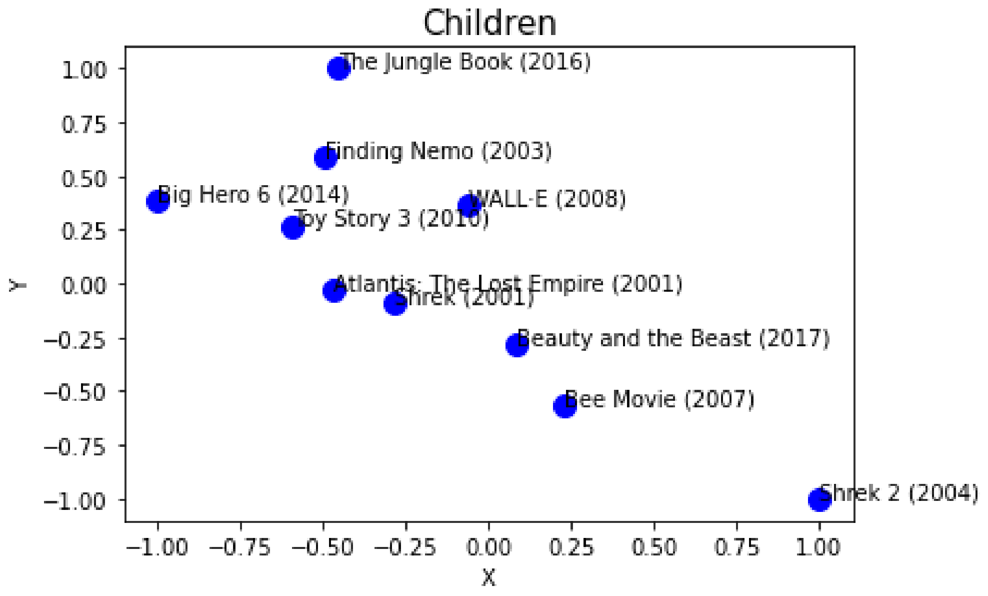

Figure 12: **HW5 D3:** Ten movies from the genre **Children**.

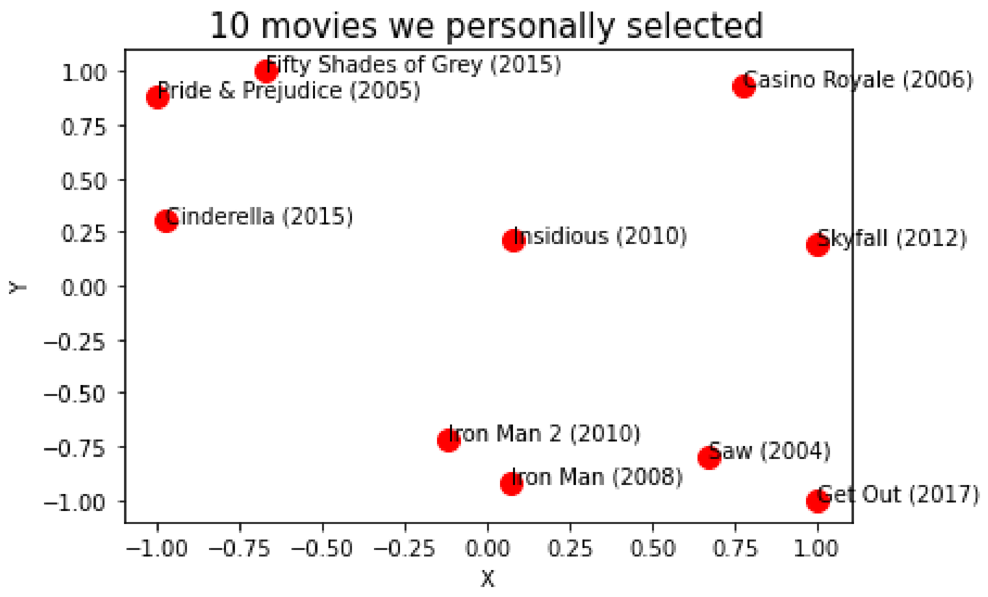

Figure 13: **Bias A:** Any ten movies of your choice from the MovieLens dataset.

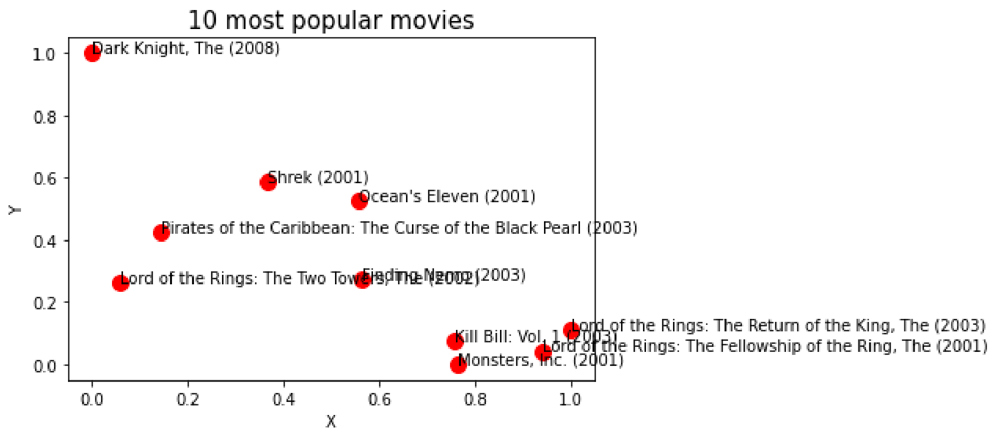

Figure 14: **Bias B:** The ten most popular movies (movies which have received the most ratings).

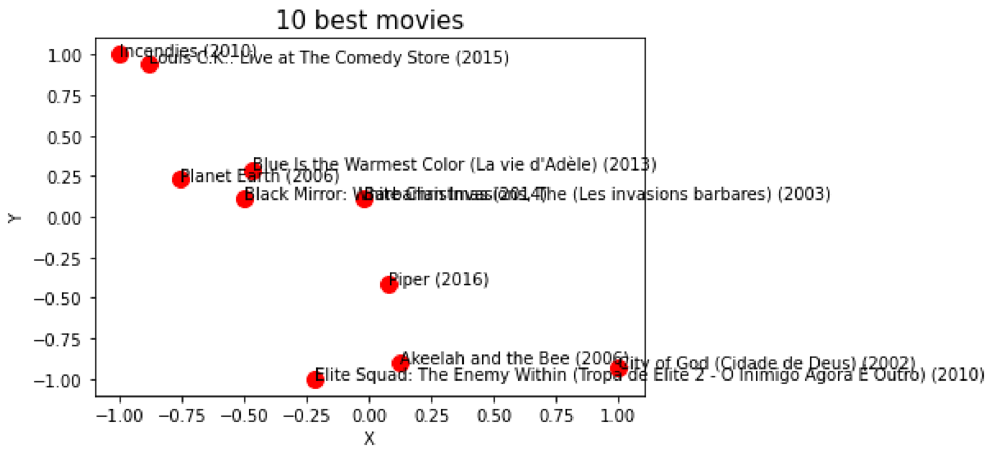

Figure 15: **Bias C:** The ten best movies (movies with the highest average ratings).

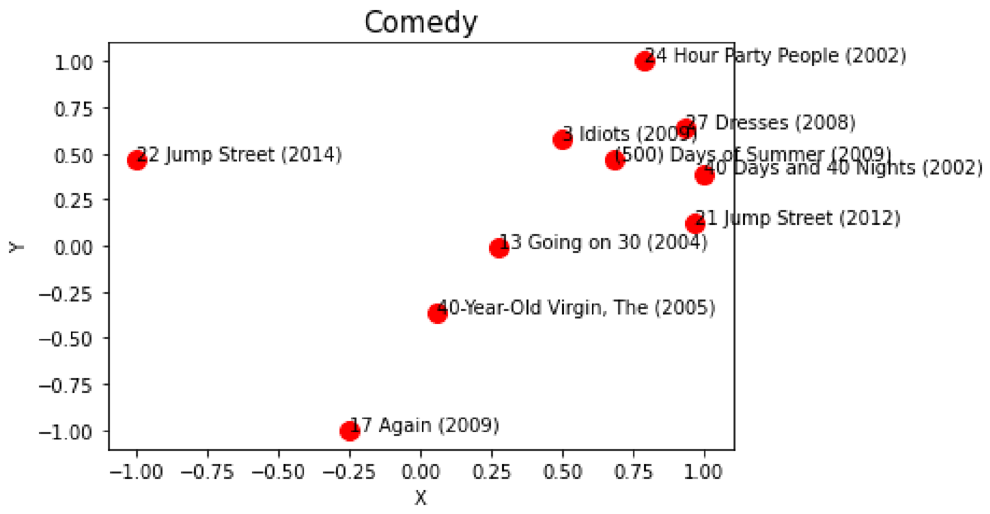

Figure 16: **Bias D1:** Ten movies from the genre **Comedy**.

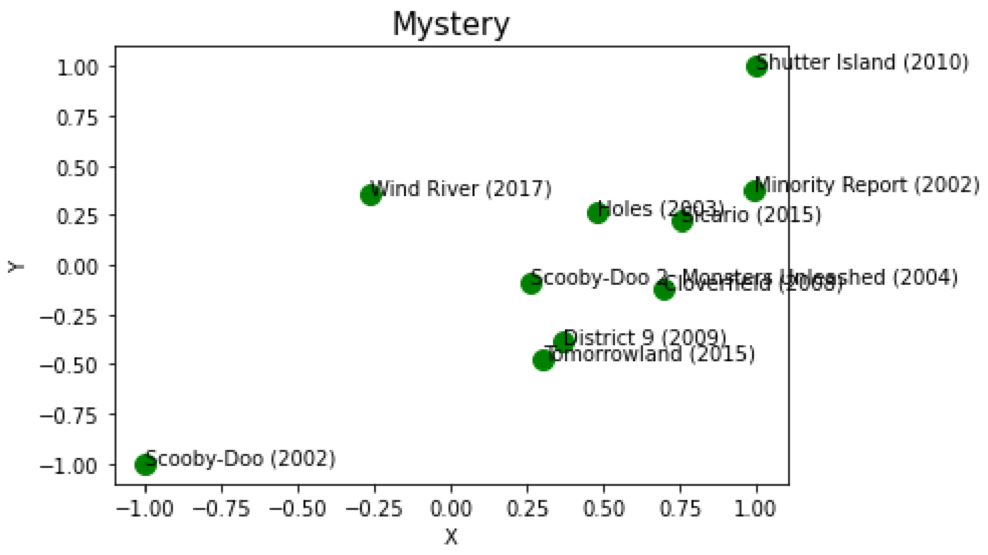

Figure 17: **Bias D2:** Ten movies from the genre **Mystery**.

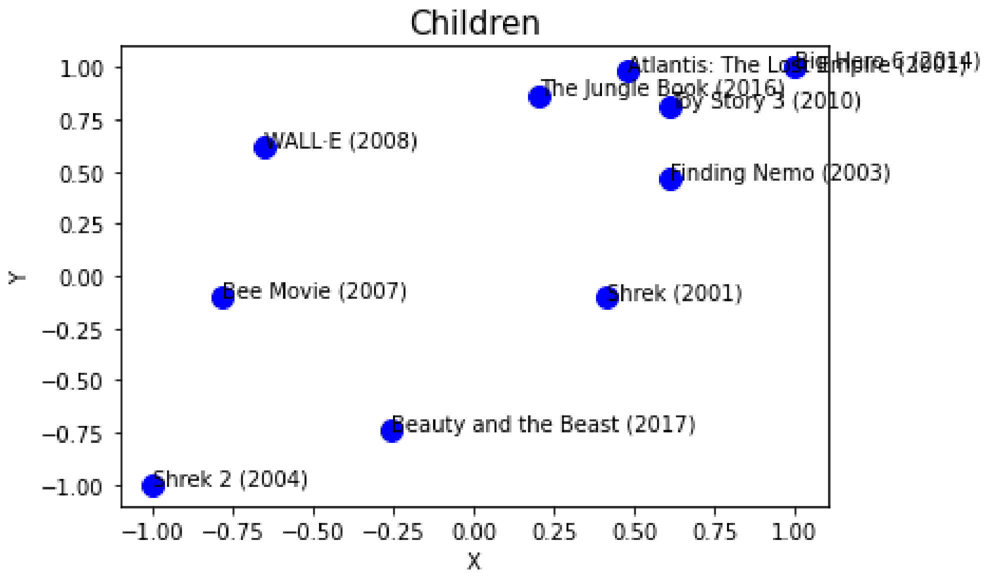

Figure 18: **Bias D3:** Ten movies from the genre **Children**.

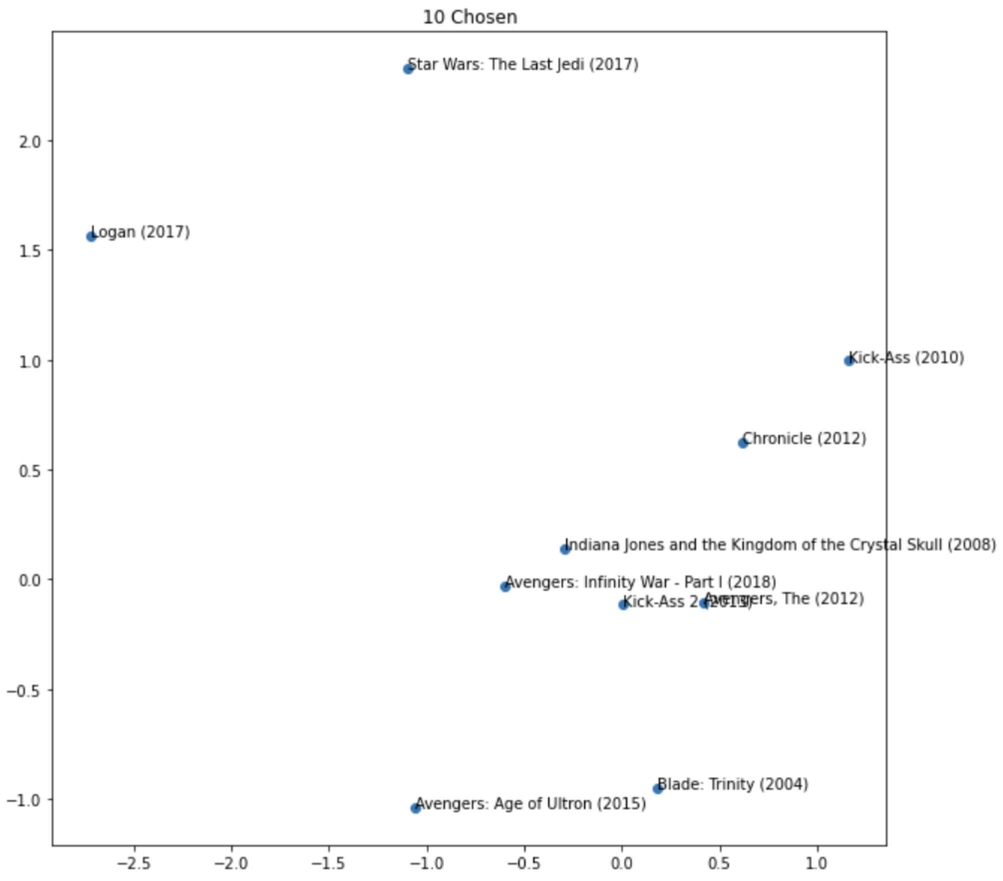

Figure 19: **COTS A:** Any ten movies of your choice from the MovieLens dataset.

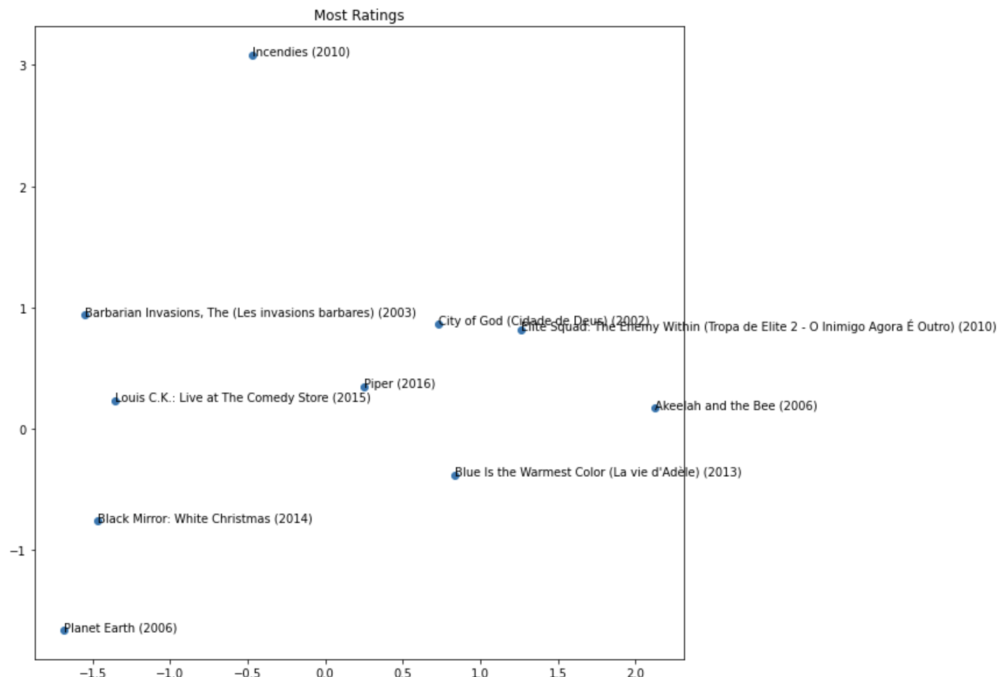

Figure 20: **COTS B:** The ten most popular movies (movies which have received the most ratings).

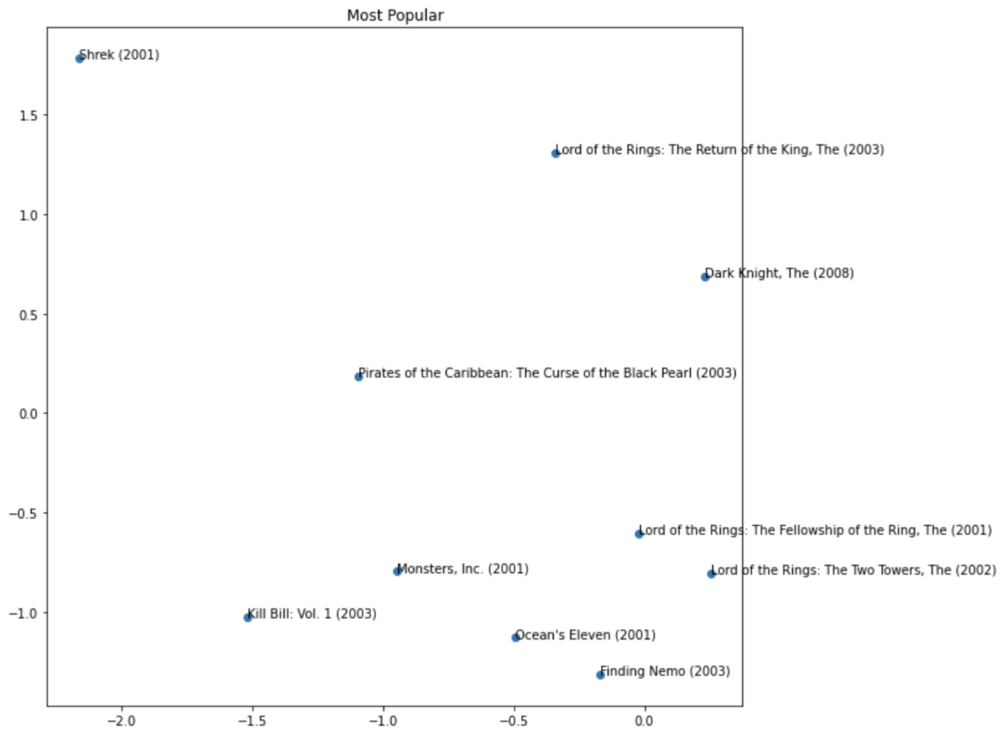

Figure 21: **COTS C:** The ten best movies (movies with the highest average ratings).

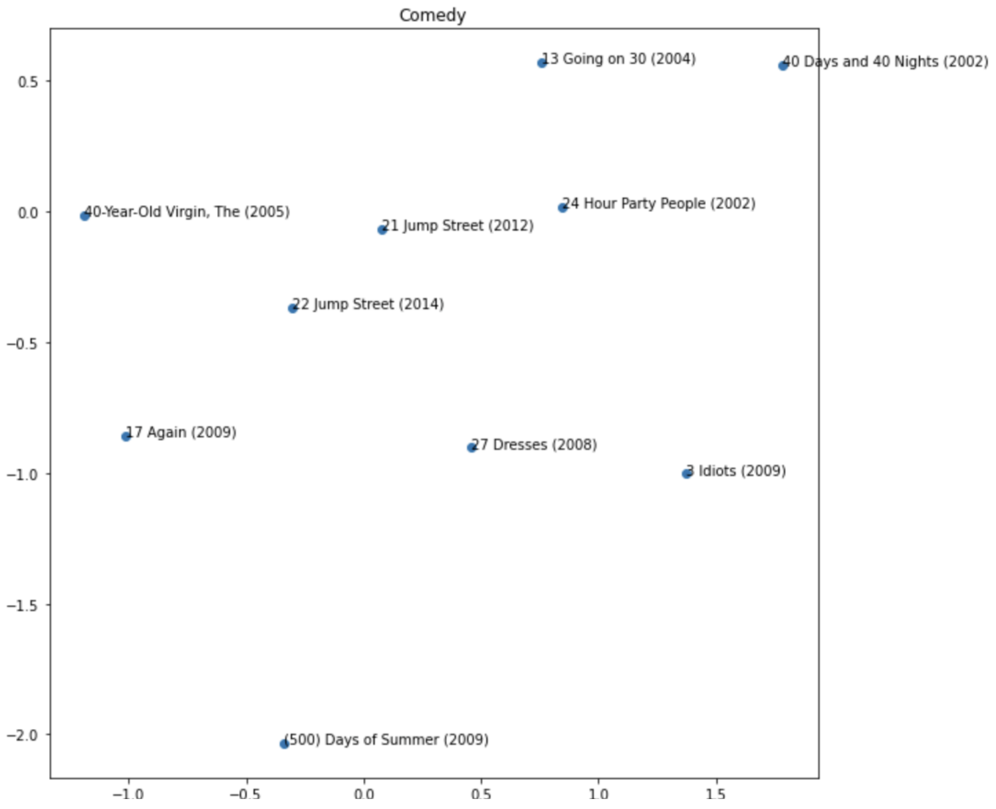

Figure 22: **COTS D1:** Ten movies from the genre [Comedy].

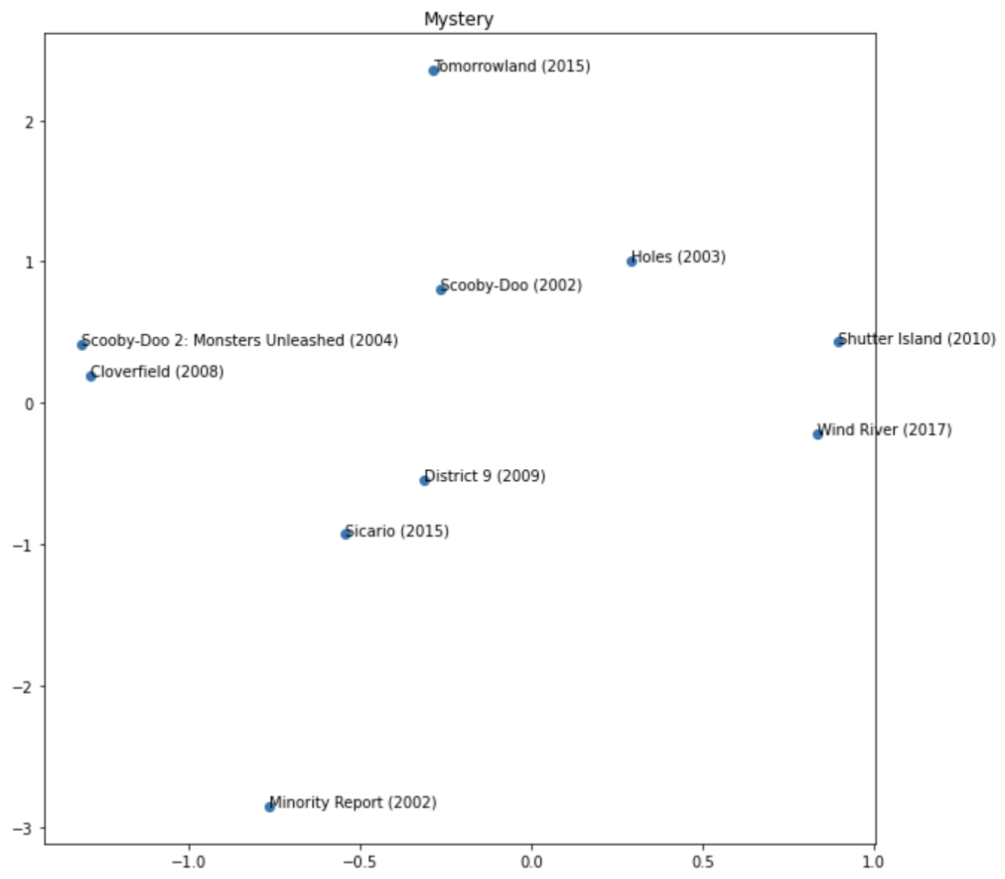

Figure 23: **COTS D2:** Ten movies from the genre [Mystery Movies].

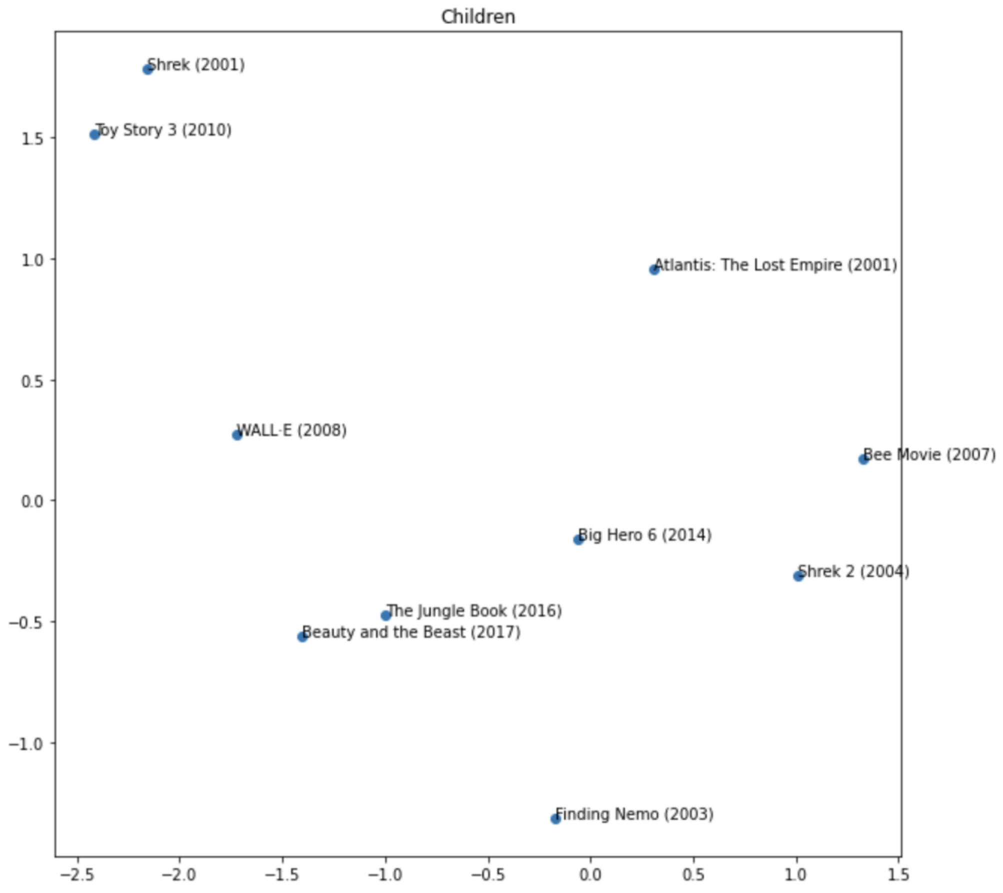

Figure 24: **COTS D3:** Ten movies from the genre [Children's].