**Hybrid Recommendation Systems: Content or Collaboration, Which Comes First?**

Gilles G. Bouyer

Harrisburg University of Science and Technology

**Abstract**

This paper examines hybrid recommendation systems, mainly used in e-commerce for recommending movies to watch by companies like Netflix and Amazon. Hybrid systems mix two steps: a content recommendation step and a collaborative recommendation step that allow them to match products with demand based on product content and on evaluations or reviews that were provided by other customers. This project aims to explore the intricacies of a recommendation system and to determine if it is better to link content first followed by collaborative recommendations or to link collaborative recommendations first followed by content recommendations. For my study, I created a movie recommendation application that will learn from a MovieLens dataset, containing 10,000 movies reviewed by 138,000 users. In starting the application, a viewer enters the name of a movie she or he liked. One hundred movies are selected from the existing 10,000 in a first step, and a final five movie recommendations are issued in the second. This application forms the basis of my analysis. Using Python code in Jupyter Notebooks running on a Mac OS, I conducted an exploratory data analysis and compare these two-phase approaches: a) the collaborative-then-content approach with b) the content-then-collaborative approach. In introducing an error function similar to an F value to compare the two approaches, I find the collaborative-then-content approach is the superior of the two.

Keywords: Recommendation systems, content recommendations, collaborative recommendations, hybrid recommendation systems, cold-start problem.

# Introduction

With an ever-growing amount of information available online, consumers are looking for new ways to make sense of an overwhelming amount of data as they try to make informed decisions about what to buy online. Recommendation systems make these decisions easier as they propose a list of choices that have been prioritized for consumers based on various inputs. For example, companies like Netflix and Amazon that offer on-demand video streaming use such systems to offer tailored recommendations to their customers. As the product characteristics constantly evolve, these recommendation systems need to adapt rapidly to cope with the pace of technological evolution, consumer expectations, and the competition.

Recommendation systems are key to generating revenue for e-commerce companies. They are also an interesting topic for study by data scientists, who are in high demand as architects of these systems. These systems employ two basic algorithms: one for content recommendations and another for collaborative recommendations. Content refers to the characteristics of the products a company wants to sell, and the content algorithm groups products together based on these characteristics. Collaboration refers to matching the users’ opinions of a company’s product to consumers’ preferences. For example, a content recommendation system might recommend a movie from its database of movies similar to those the consumer watched before. A collaborative recommendation algorithm, on the other hand, matches viewers who watch movies to those who share similar tastes, and based on those, it recommends a new movie. Since no two viewers are exactly alike, it is challenging to find a recommendation system that can match each viewer’s tastes 100% of the time. Improving recommendation methodologies and measuring their efficiency is a data science task critical to e-commerce companies’ success.

## Literature Review

The body of knowledge and articles covering recommendation systems is extensive, so I have summarized the main studies with four key criteria in Table 1, which are the research problem, dataset, methodology, and measurement and interpretation of observations.

The first problem tackled in Table 1 deals with the speed of the recommendation systems. Market research has shown that when online, a consumer has little patience. While a recommendation may be prepared in advance, the recommender system needs to review all possible product recommendations for all of the clients based on a variety of factors, including product and client characteristics, preferences, and the recommendations’ historic validity. With 158 million customers in the third quarter of 2019 and approximately 5,600 movies and series in its catalogue, Netflix assigned serious computing resources to the task of creating individualized recommendations.[[1]](#footnote-1) Potential solutions include using a graph database (Patel & Dharwa, 2016) and focusing on a reduced set of data (Nayak, Mirajkar, Rokade, & Wadhwa, 2018) to make predictions. A graph database, made of nodes and links, will speed up the associations between components, while a relational database will spend time linking tables with JOIN functions to allow for more data. Another branch of recommendation systems focuses on optimizing a company’s revenue, such as when Greenstein-Messica and Rokach (2018) seek to maximize a customer’s Willingness to Pay (WTP). They consider product characteristics like prices, product discounts, sellers’ reputations, and other exogenous factors focusing on dollar amounts.

Too much, or too little, data is also problematic. Li, Wang, Cao, and He (2018) addressed the issue of too much data by mixing two linear regression models. Dang, Vinh, Tay, Zhang, Cong, and Li (2018) used hyperbolic distance to separate data space more efficiently, and Himel Uddin, Hossain, and Jang (2017) isolated a reduced set of five attributes (actors, directors, rating, genre, and year) to produce a movie recommendation system. Having no data is known as the *cold-start problem.* When an engine is cold, the car does not operate optimally, but as the engine runs, the car warms up and operates efficiently. A recommendation engine follows the same logic, with data being equivalent of the gas the engine needs to run. How can one determine which products users will choose when they log into a recommendation system for the first time and there is no data on what they like? Different researchers have devised different solutions to solve this issue. In their survey of recommender systems, Abbas, Zhang, and Khan (2015) applied five kinds of methodologies, ranging from fuzzy logic to an artificial neural network. Shriver (2018) introduced fuzzing techniques and Kula (2015) introduced sparse matrix representation to reduce the space needed to store the information while calculating predictions. Li et al. (2018) mixed data into different weighted regressions, so there were only a limited number of cases without information.

Data typically contain explanatory information, and their explanatory power can be measured to reduce the amount of data needed to produce suitable recommendations. Many of these studies are heuristic, and researchers select measurements depending on what they aim to improve. When trying to improve the recommender design, Nayak et al. (2018) focused on rating the system’s functionalities. When trying to improve the recommendation’s quality, Nilashi, Ibrahim, and Bagherifard (2017) measured its elasticity versus the parameters’ variations. When reviewing the validity of a choice, according to Abbas et al. (2015), Kula (2015), and Nilashi et al. (2017), the measures were precision, recall, F1 statistic, and their associated displays through a ROC curve or its equivalent. When reviewing the distance to a particular variable, several researchers used the root-mean-square error (RMSE) (Abbas et al., 2015; Greenstein-Messica & Rokach, 2018; Himel et al., 2017).

As meaningful, positive results are being reported, one also needs to observe the conditions under which these measurements were made and pay special attention to the data used. For hybrid movie recommendation systems, all studies use movie ratings and product data. For a movie, the product data will include its running time, genre, actors, directors, and set location. Kula (2015) used these data through the creation of tags applied to movies that provide its content characteristics. Another way to introduce content data is to use a movie ontology (Nilashi et al., 2017) that classifies all existing movies according to the semantic similarity found in their descriptions and scripts. Other studies (Abbas et al.,2015) used contextual data that answered the questions of who, what, how, where, when, and why. Another approach considered how one selects a movie through price, time, and immediacy. Greenstein-Messica and Rokach (2018) showed that people are more likely to pick a product when they can apply a discount because of who they are (e.g., being a long-time club member)—rather than a discount just for the product (movie). Hybrid recommendation methods can either follow a two-step approach (Abbas et al.,2015; Nayak et al., 2018; Nilashi et al., 2017; Patel & Dharwa, 2016) linking similar movies and viewer tastes, or include all data in a one-step calculation (Himel et al., 2017; Kula, 2015).

All studies will attempt to minimize the cost defined as the difference between predicted values and what was actually observed from a test set. Greenstein-Messica and Rokach (2018) use the Akaike Information Criterion (AIC) to evaluate the difference between predicted and observed values.

The other main algorithm researchers can use is *K* Nearby Neighbors (KNN), with *K* being the number of movies that are the closest to the movie of reference. Here, the distance formula differs as the focus is not on the closest matches, but rather on a number (*K)* of similar movies. Another way to group recommendations is using *K*-means algorithms (Himel et al., 2017) to create *K* groups of movies.

Nilashi et al. (2017) use Singular Value Decomposition (SVD) as it allows researchers to reduce the variables down to the ones that hold the most value (eigenvectors). SVD then identifies the factors (weights) that can be used to make the prediction. Neural networks are often used (Abbas et al.,2015), and they are very efficient. They combine multiple layers of matrices and apply linear algebra calculations to deduce multiple parameters (weights) to predict the result. The most difficult issue a neural network faces is the impossibility of explaining the reason for the decision. The high number of information layers makes it extremely hard to interpret what is happening at each network level. This is why this type of prediction is, therefore, rarely used for government-regulated activities (e.g., mortgage markets).

The following four papers are the ones that provided the basis for the techniques and ideas I used in this research. Kula (2015) provides methods and techniques to predict movie ratings for when only sparse information is available. These authors provided the ideas and tools on how best to measure the recommendation values: Nayak et al. (2018); Li et al. (2018); and Himel et al. (2017).

**Research Objective**

The mix of collaborative- and content-based filtering occurring in hybrid recommenders enables them to use product data and people’s preferences to provide a well-rounded recommendation. The study’s objective is to confirm my hypothesis of a best order for linking content and collaborative recommendations. I evaluate whether recommendations are better when the hybrid model starts with collaborative-based and then uses content-based filtering or when it starts with content-based first and use collaborative filtering afterwards. I seek to answer the question of which step should be run first in the hybrid recommendation improving suggestions and company bottom lines.

To determine the best order, I sought to build a hybrid movie recommender application from which I can develop my analysis, developed a new measurement to compare the two approaches, and calculate results to recommend which step should come first. Building my own recommender allows me to base my research on an application that works. I can explore various changes and measure if the change provides improvements. I use the underlying code of the application to compare the two scenarios.

The “tag genome” approach proposed by Kula (2015) provides the logic used to define the content. His method is inspired by DNA analysis, where genomes are associated with their components (polymorphisms). The frequency of words in the movie scripts and reviews are weighted and assigned for each movie. The movie ratings will be used to measure the collaborative aspect. Similar ratings on movies will form the collaborative base from which we can determine if a user should watch a movie she or he has not watched yet but others have seen.

I then quantitatively evaluate the results by creating my own measurement function mixing a content measurement and a collaborative measurement to compare the two original scenarios. Content will be evaluated using the average of the cosine similarities on movie tags with a variable I abbreviated as CosDis for cosine distance, and the collaboration’s strength will be evaluated by using an RMSE function comparing the actual ratings to the ones predicted for an individual. To predict the ratings, I will use the SVD technique to predict which rating a viewer would give to a movie, and then use the predicted and actual ratings to calculate the RMSE. (See Appendix A.) I will then introduce the F value, equal to CosDis x 1/RMSE, that mixes these collaborative and content measurements to compare the two scenarios. The findings will show that the larger the F value is, the better the model is. (See Appendices A and B for more information on cosine similarity.)

### Methodology

I used the MovieLens 20M Dataset that has 20 million movie ratings collected from 138,000 users on 10,000 movies.[[2]](#footnote-2) While the dataset had documentation, it was not enough to understand how to use the data in my study, so I explored it using Excel and Pandas to understand how best to use the set. (See Appendix D.)

**Application**

I built a movie recommender application, linking content then collaborative recommendations, using Python, Jupyter Notebooks, and Flask. Python has the necessary program libraries and the flexibility to model content and collaborative methodologies. Flask enables the application to run in a browser. Jupyter Notebooks facilitates data exploration and result analyses.

The application recommends five movies to a specific user, based on the user’s request to see a movie similar to one already watched. I started by identifying 100 movies that are similar to the movie selected by the user. Then the application predicted the ratings that the user would have given to those movies, based on ratings that other users with similar tastes have provided. Finally, the application recommended movies the viewer has not seen with the highest predicted ratings. The application setup is described in Appendix C, and the project repository, can be found at Github (see footnote 3 for the URL) with a video of the application in action.

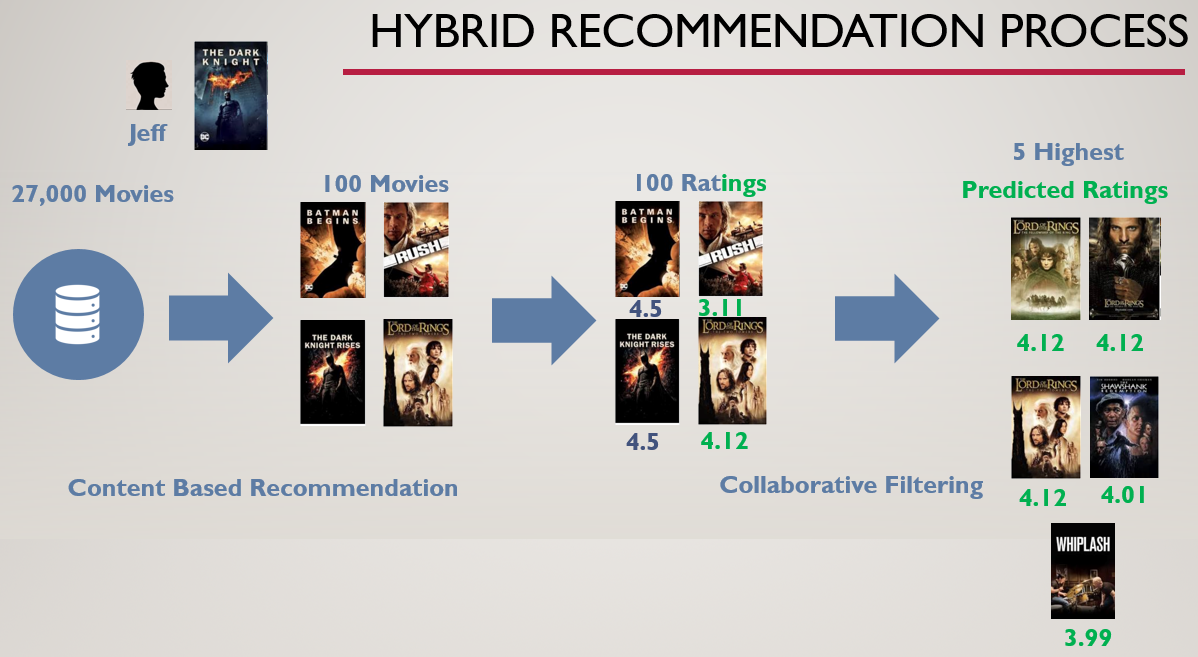
An example of the application’s UI appears below in which our user, Jeff, wants to watch a movie similar to *The Dark Knight*. Jeff enters the movie name, clicks on the yellow bar, and receives recommendations for five movies that he has not previously seen.



*Figure 1*. Hybrid Recommendation User Interface.

Typing “python main.py” at the Terminal prompt initiates the application and generates a link to enter into a browser.[[3]](#footnote-3)

The application process consists of a user looking for movies similar to ones already watched. The 100 movies closest to the movie of reference are selected and movie ratings for this viewer are predicted if the viewer has not watched them. The top five ratings are used to identify the suggested movies. Figure 2 depicts this process.



*Figure 2.* Overall process flow of a content-collaborative-based movie recommender.

#### Statistical Methods

I used the two recommendation models: Content recommendation and collaborative recommendation, and ran them one after another. Scenario 1 (S1) was content followed by collaboration, and Scenario 2 (S2) was collaboration followed by content.

Collaborative recommendations will provide the ability to predict which rating the viewer would give to the available movies based on the knowledge of what people with the same tastes gave to those same movies. I used Kula’s function (2015)—the surprise library—to calculate the predicted ratings. Once the ratings are predicted, the application will recommend the highest-rated movies.

#### Algorithm

The algorithm mixes data preparation, content recommendations, and collaborative recommendations, as well as displays results. I prepared the data by transforming tables into matrices and dataframes. Content recommendation is about creating a matrix that measures the content between movies. I did this by calculating the cosine distance between movies. Collaborative recommendation is about giving a possible ranking a use would have given to a movie based on her or his previous choice and other viewers’ previous choices. I did this by using SVD to approximate the values missing in the rating matrix. (See Appendix A for the algorithm.)

#### Measurement

After proposing a list of movies, I produced the RMSE, providing the error of prediction between the ratings predicted and the actual ratings. This measures the collaborative aspect of the recommendation.

Content recommendation is evaluated by using the cosine similarities between recommended movies and the movie of reference. The best value of CosDis would be 1 if the movies were perfectly aligned with a zero angle, resulting to a cos(0°) = 1 value. Meanwhile, the best value of the RMSE would be zero if there was no difference in preferences between the observed and predicted movie ratings. I mixed the two, creating an F value that divides the cosine by the RMSE.

The higher the F value, the better the model performs.

**Results and Measurement**

In Scenario 1 (S1), I linked content then collaborative recommendations. In Scenario 2 (S2), I linked collaborative then content recommendations. (S1 is depicted in Figure 2.) For S2, I predicted the ratings for all 10,000 movies in the database for one viewer. I then isolated the top 100 ratings and recommended the five movies that have the highest cosine similarity (representing the same movie characteristics) with the movie of reference. The top 100 sets were different in S1 and S2, as S1 contained the movies closest to the reference movie in terms of content, while S2 contained the movies closest to Jeff’s preferences. Clicking on this [link](https://github.com/gillesbouyer/ANLY699_Project/blob/master/01_content_tag_recommender.ipynb) provides access to the code used to perform the analysis and to create the main Python programs. (See Appendix C.)

I found that linking content then collaboration has an RMSE of 0.225 and linking collaboration then content has an RMSE of 0.057. As these reflect an average distance between predicted and actual values, one can estimate that the smaller the RMSE is, the better the prediction. While RMSE reflects the collaborative aspect, where ratings are predicted through collaboration, the CosDis between the first 100 movies and the movie of reference shows how close the recommended movies are to the movie of reference. This is expressed as:

F = CosDis x 1/RMSE

In my experiment where one viewer, Jeff, is looking for a movie similar to one he has seen, I found that FCollaboration\_Content = 8 and FContent\_Collaboration = 4. This implies that the collaboration-then-content model is superior to the content-then-collaboration model. These values can be found by clicking on the file 01\_content\_tag\_recommender.ipynb of the Github repository [github.com/gillesbouyer/ANLY699\_Project](https://github.com/gillesbouyer/ANLY699_Project) and scrolling to the end of the file.

#### Conclusion

Recommender systems and their associated algorithms offer a rich and dynamic body of knowledge that is critical to e-commerce sales. The two mix the objective aspect of comparing content with the subjective aspect of comparing viewers’ tastes and preferences. Yet, is it possible to determine a better order for collaborative and content recommenders to optimize results? I explored this issue by creating a hybrid recommender using GroupLens’ MovieLens dataset and showed how to improve recommendations from a hybrid system I built. I compared two algorithms (linking the content-collaborative method and then linking the collaborative-content method) using RMSE to measure the content aspect and CosDis for the collaborative aspect. In my experiment where one viewer, Jeff, is looking for a movie similar to one he has seen, I find that linking content then collaboration has an RMSE of 0.225 and linking collaboration then content has an RMSE of 0.057. As these reflect an average distance between predicted and actual values, one can estimate that the smaller the RMSE is, the better the prediction. This, therefore, favors the path of collaboration first instead of content first. As RMSE does not reflect the content aspect, I created the average cosine distance (CosDis) to show how close the recommended movies are to the movie of reference and introduced the F value, expressed as:

F = CosDis x 1/RMSE

In my experiment where Jeff is looking for a movie similar to one he has already seen, I show that FCollaboration\_Content = 8 and FContent\_Collaboration = 4, giving the edge to collaboration first, content second.

Future studies, with sufficient cloud computing power, could confirm this result by repeating the experiment for all 10,000 movies and 138,000 viewers to test all possible combinations of this dataset.

Still, this modest experiment reveals how a collaboration-then-content hybrid recommendation system is more accurate than a content-then-collaboration system when recommending movies. This result will likely vary depending on the industry, the products, the data available and the resources allocated to the research. It is a key area of interest for any e-commerce company as it provides some new considerations for how these companies can improve, and monetize, the choices offered to their customers.

**References**

Abbas, A., Zhang, L., Khan, S. (2015). A survey on context-aware recommender systems based on computational intelligence techniques. *Computing Journal, 97*(7), 667–690*.* Retrieved from *﻿*http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.2640&rep=rep1&type=pdf

Dang, T., Vinh, Q., Tay, Y., Zhang, S., Cong, G., & Li, X.-L. (2018). Hyperbolic recommender systems. Retrieved from ﻿https://arxiv.org/pdf/1809.01703.pdf

Greenstein-Messica, A., & Rokach, L. (2018). Personal price aware multi-seller recommender system: Evidence from eBay. *Knowledge-Based Systems,* 150(June): 14–26. ﻿ Retrieved from <https://doi.org/10.1016/j.knosys.2018.02.026>

Himel, T., Uddin, M., Hossain, M., & Jang, Y. (2017). Weight-based movie recommendation system using K-means algorithm. *2017 International Conference on Information and Communication Technology Convergence (ICTC),* 1302–1306. ﻿ Retrieved from http://ieeexplore.ieee.org/document/8190928/

Kula, M. (2015). Metadata embeddings for user and item cold-start recommendations. Retrieved from <https://arxiv.org/pdf/1507.08439.pdf>

Li, C., Wang, Z., Cao, S., & He, L. (2018). WLRRS: A new recommendation system based on weighted linear regression models R. *Computers and Electrical Engineering*, *66*, 40–47. [doi.org/10.1016/j.compeleceng.2018.02.005](https://doi.org/10.1016/j.compeleceng.2018.02.005)

Nayak, R., Mirajkar, A., Rokade, J., & Wadhwa, G. (2018). Hybrid recommendation system for movies. *International Research Journal of Engineering and Technology, 5*(3): 12–17. *﻿* Retreived from www.irjet.net

Nilashi, M., Ibrahim, O., & Bagherifard, K. (2017). A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. *Expert Systems with Applications, 97,* 507–520. Retrieved from https://doi.org/10.1016/j.eswa.2017.09.058

Patel, A., & Dharwa, J. (2016). An integrated hybrid recommendation model using graph database. *2016 International Conference on ICT in Business Industry & Government (ICTBIG), 1*–*5.* Retrieved from http://ieeexplore.ieee.org/document/7892680/

Shriver, David (2018). Assessing the quality and stability of recommender systems. *University of Nebraska-Lincoln, Computer Science and Engineering: Thesis, Dissertation.* Retrieved from https://digitalcommons.unl.edu/computerscidiss/147

**Tables**

Table 1

Literature Review Summary

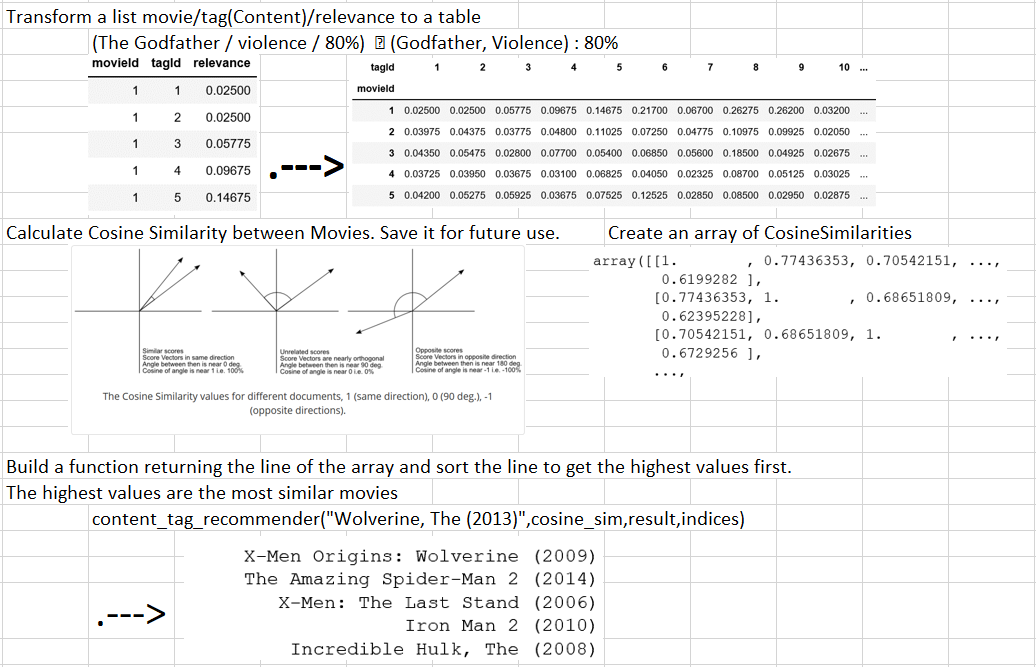
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Literature | Problem | Data | Techniques | Measurements |
| Abbas: Context-aware | Cold-start problem, sparsity | Time, location, ratings, and product | Many (ANN) | MSE, Precision, recall |
| Dang: Hyperbolic | Too much choice | People, items, and density | Hyperbolic distance. BPR triple loss | Hit rate, nDCG |
| Greenstein:  eBay | Optimize willingness to pay | Products, seller reputation, and promotion (Y/N) | Matrix Factorization, regularization | Lowest AIC, Kolmogorov-Smirnov |
| Himel: Weight *K*-Means | What’s the relevant data | Ratings and products | K-Mean | RMSE, MSE |
| Kula: LightFM | Cold-start problem | Ratings and tags | Weighted gradient descent | ROC curve |
| Li: Weighted regression | Massive information | Users, items, and ratings | Linear regression | RMSE, MAE |
| Nayak: Hybrid movie | Speed and accuracy | Ratings and product data | Cosine *KNN* | Functionalities |
| Nilashi: Ontology | Improve recommendations | Ontology, Ratings | SVD | Precision and recall |
| Patel: Graph database | Speed | People, products, and ratings | Graph SQL | Speed |
| Shriver: Quality | Goodness of fit | People, products, and ratings | Fuzzing | Elasticity of change |

**Appendix A**

**Content and Collaborative Pseudo-Algorithm**

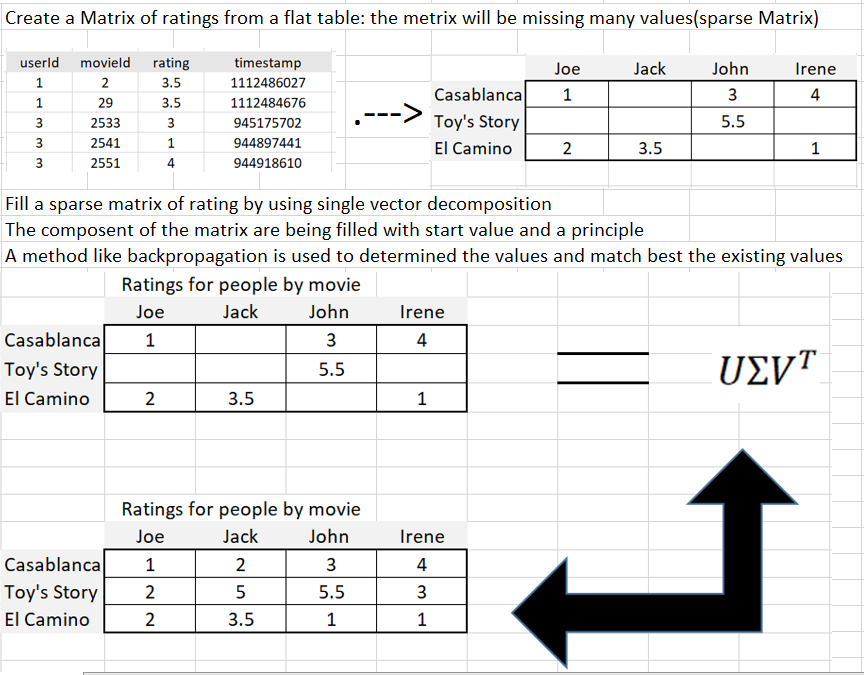
The recommendation part of the algorithm is as follows. Content collaboration creates a matrix comparing each movie to all the others. The matrix is (n,n) with a diagonal of ones and the values in each row k are equal to the values in column k. Each tag has a number (genome tag table gives the correspondence between the tag number and the name of the tag), each movie has values for all the tags, and the cosine distance between all these characteristics provide content comparison between movies. One selects a line of the cosine matrix corresponding to the movie and ranks the cosines from the largest to the smallest to determine the five most similar movies.

Content-Based Recommendation:



**Building a Collaborative-Based Recommendation**

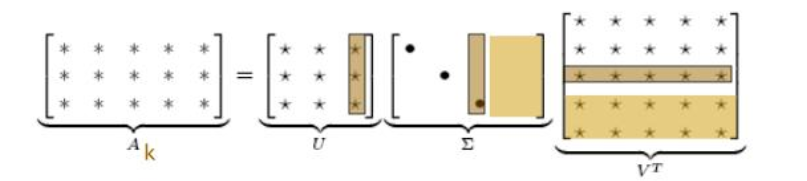
Collaborative recommendation is about predicting the ranking a person would have given to a movie based on his previous choice and other people’s previous choices. I populate a matrix with the values of the ratings given to movies. The matrix is sparse as most people have not rated all 10,000 movies. I then decompose the matrix in 3 matrices, initialize the missing values to a small number, and conduct back-and-forth calculations (decomposition 🡨🡪 multiplication) until reaching an optimal result when calculating the existing values of the matrix. All the other values filled are the predictions.



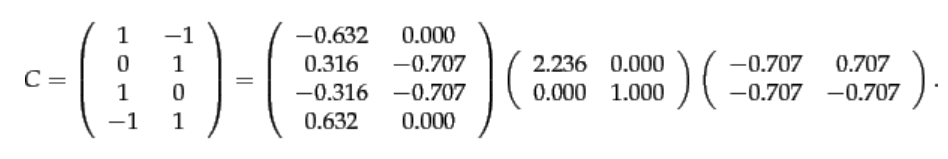
**Singular Value Decomposition**

Singular Value Decomposition (SVD) can be viewed as a transformation of a matrix tracking information common to people and movies (the rating) into three matrices. The first matrix, **U**, tracks information between movies and a transformed set of people. This set is sometimes called principal components. The second matrix, ∑ is a diagonal matrix (zero values except the diagonal) that represent the importance these components have component by component, the first value on the upper right being the highest of them. The third matrix **VT** is tracking the relationship between principal components and the people.

Examples of how a matrix can be calculated by multiplying U, ∑,and **VT** appears below:



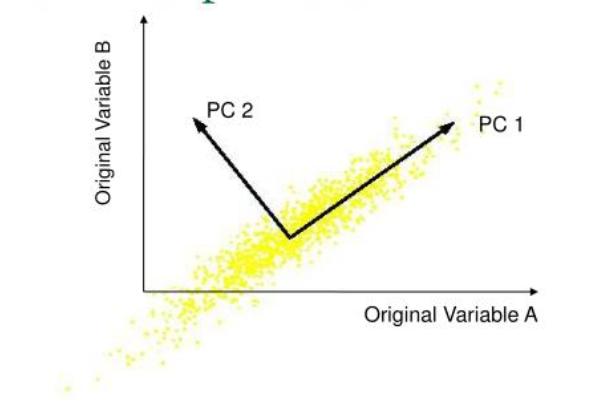
<https://www.slideserve.com/braden/eigen-decomposition-and-singular-value-decomposition>



<https://nlp.stanford.edu/IR-book/html/htmledition/term-document-matrices-and-singular-value-decompositions-1.html#eqn:svd>

**Principal Components**

The idea is to find another way to represent something that allows to differentiate the values observed in a better way. One can imagine a cloud of points representing movies based on their profits and ratings (assuming that ratings can be negative). One sees the yellow/gray cloud of points. Replacing the variables A and B by Principal Components 1 and 2 characterizes and differentiates the movies better. These two dimensions represent what the matrix ∑ contains.



**Appendix B**

**Cosine Similarity**

We are using cosine similarity to calculate how different two concepts are. We saw in Appendix A that this is linked to the alignment of the concepts. The more aligned they are, the more the angle between the two concepts will trend toward 0, the more the cosine of the angle will trend to 1 (cos(0o) = 1). Therefore, the higher the cosine similarity between the two concepts, the more they match. Consider the following three movies: *Daddy’s Home* (2015), *Instant Family* (2018), and *Invincible* (2006). Actor Mark Wahlberg stars in all three movies. In *Daddy’s Home*, he co-stars with Will Ferrell, so I can assign “Wahlberg” a coefficient of 0.75 on *Daddy’s Home*, and 0.95 on the two other movies where he is the main star. *Invincible* is an action movie, so I can assign it a .6 weight for “Action.” The two others are family movies, so I give them the following weights respectively 0.96 on *Daddy’s Home* and 0.9 on *Instant Family*. I can then represent these movies with the following coefficients for the “Action,” “Wahlberg,” and “Family” tags:

*Daddy’s Home*: “Action”: 0.2, “Wahlberg”: 0.75, “Family”: 0.9

*Instant Family*: “Action”: 0.3, “Wahlberg”: 0.95, “Family”: 0.96

*Invincible*: “Action”: 0.8, “Wahlberg”: 0.95, “Family”: 0.4

The formula of the cosine similarities of two vectors (x1, y1, z1) and (x2, y2, z2) is:

The higher the similarity, the closer the two vectors are.

In applying this to these movies, I then calculate the cosine similarity between *Daddy’s Home* and *Instant Family* as 0.99:

Between *Daddy’s Home* and *Invincible*, it is 0.79:

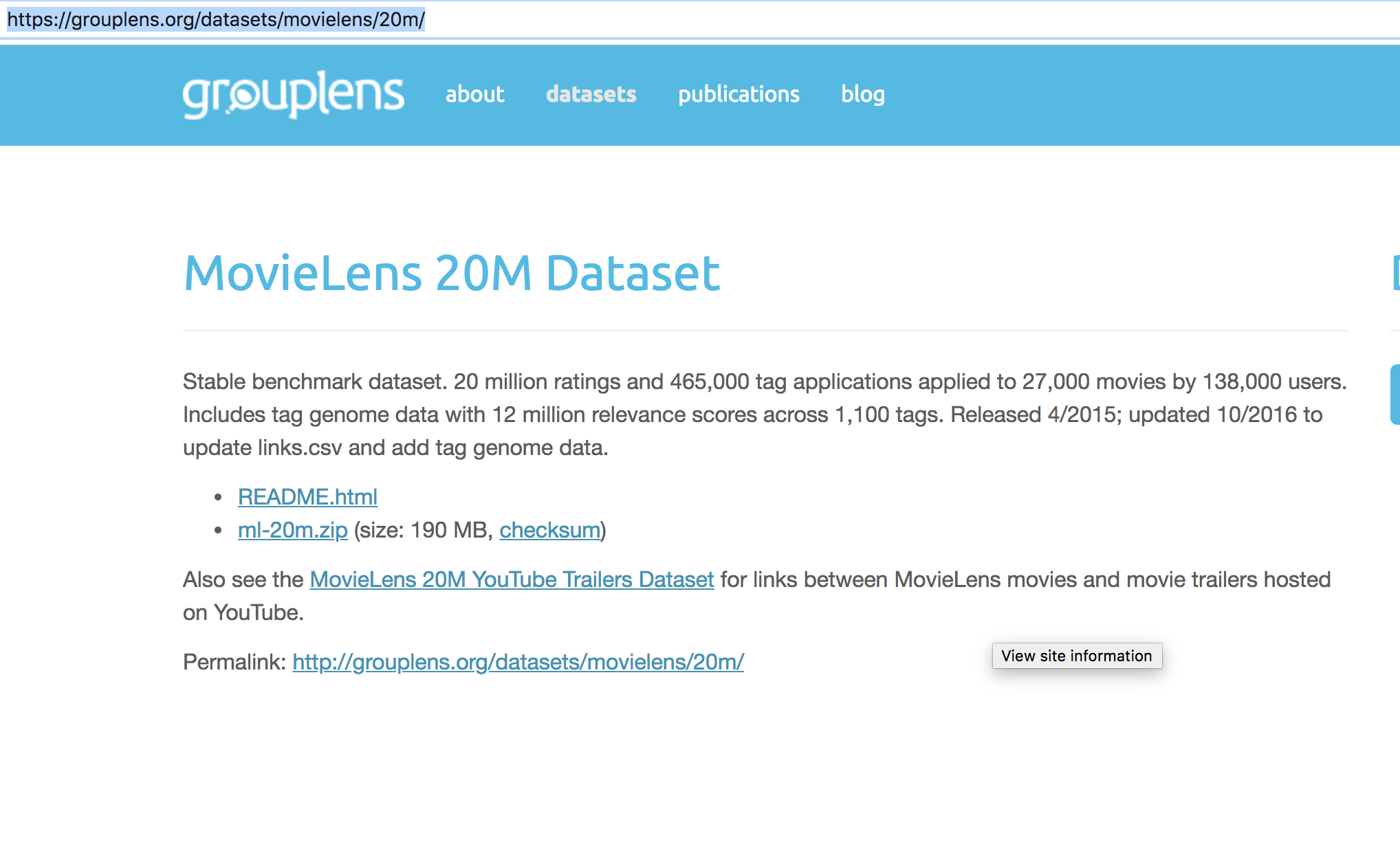
Between *Instant Family* and *Invincible,* it is 0.84:

The highest similarity found is 0.99 for *Instant Family* and *Daddy’s Home*. These two are the most similar movies.

**Appendix C**

**Application Set-Up**

I describe the data used in this study in more detail below, as well as how to view it. The dataset is contained in a zip file located at: <https://grouplens.org/datasets/movielens/20m/>



Once downloaded and opened, one sees the following files:



-*Ratings* file contains the 20 million ratings given by 138,000 viewers to 10,381 movies (even if it says 27,000 movies in the dataset description). Ratings range from 0.5 to 5 by ½ steps. This is used to do the collaborative part.

-*Genome-tags* file contains the name of 1,100 tags (e.g., afterlife, Africa, England, hotel, Robert Downey, Jr.) and their id.

-*Genome-scores* file contains the movieid and the tagid and the relevance of the tagid to the movie.

For content, I used the following files, applying 1,100 tags 465,000 times to 27,000 movies

-*Movies* file contains the movieid and the title of the movies.

-*Links* file links movies to the IMDB and the movie database (TMDB).

-*Tags* file shows which tags were given to which movie by which viewer.

Files need to be put in the data folder of the application. Running the two notebooks generates additional files as follows in the data folder:



-*Cosine* file contains the distance between movies.

-*Svdpred* file contains the weights used to predict the ratings.

-*Contentreco* file contains the 100 movies selected during content filtering.

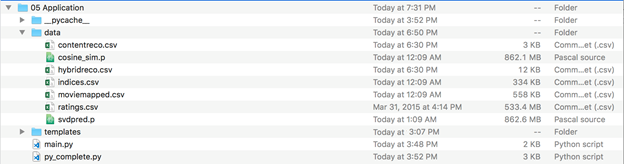
-*Hybridreco* file contains the subset of 100 movies that have not been rated and ranked by predicted rate. What is returned in the application is the first five records.

-*Indice* is a table title-index.

-*Movietagged* movie/tag.

-*Moviemapped*: index – movieid –Title. Index goes from 0 to 10380. Movieid from 1 to 131170.

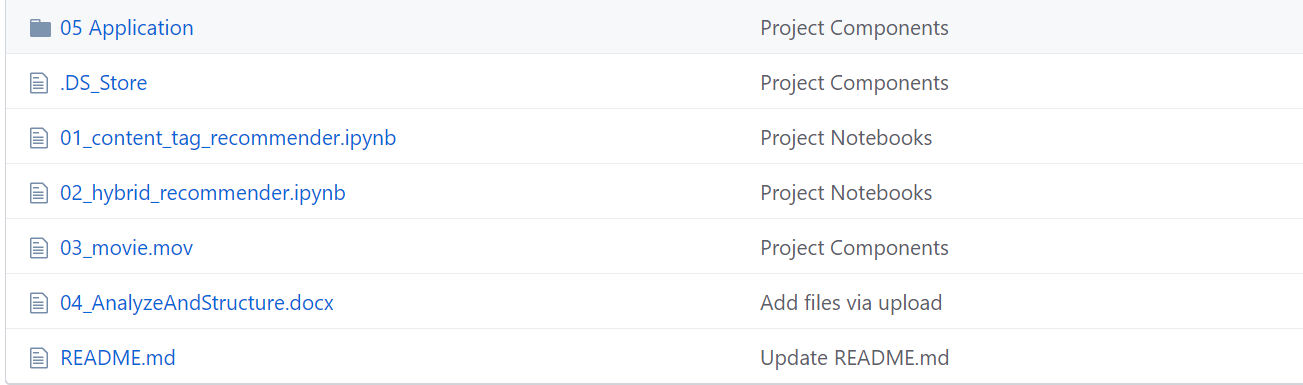
These additional files are placed in the data folder used by the application:



Overall, the project folder looks like:



And the Github repository looks like, 04\_AnalyseAndStructure.docx being this document.

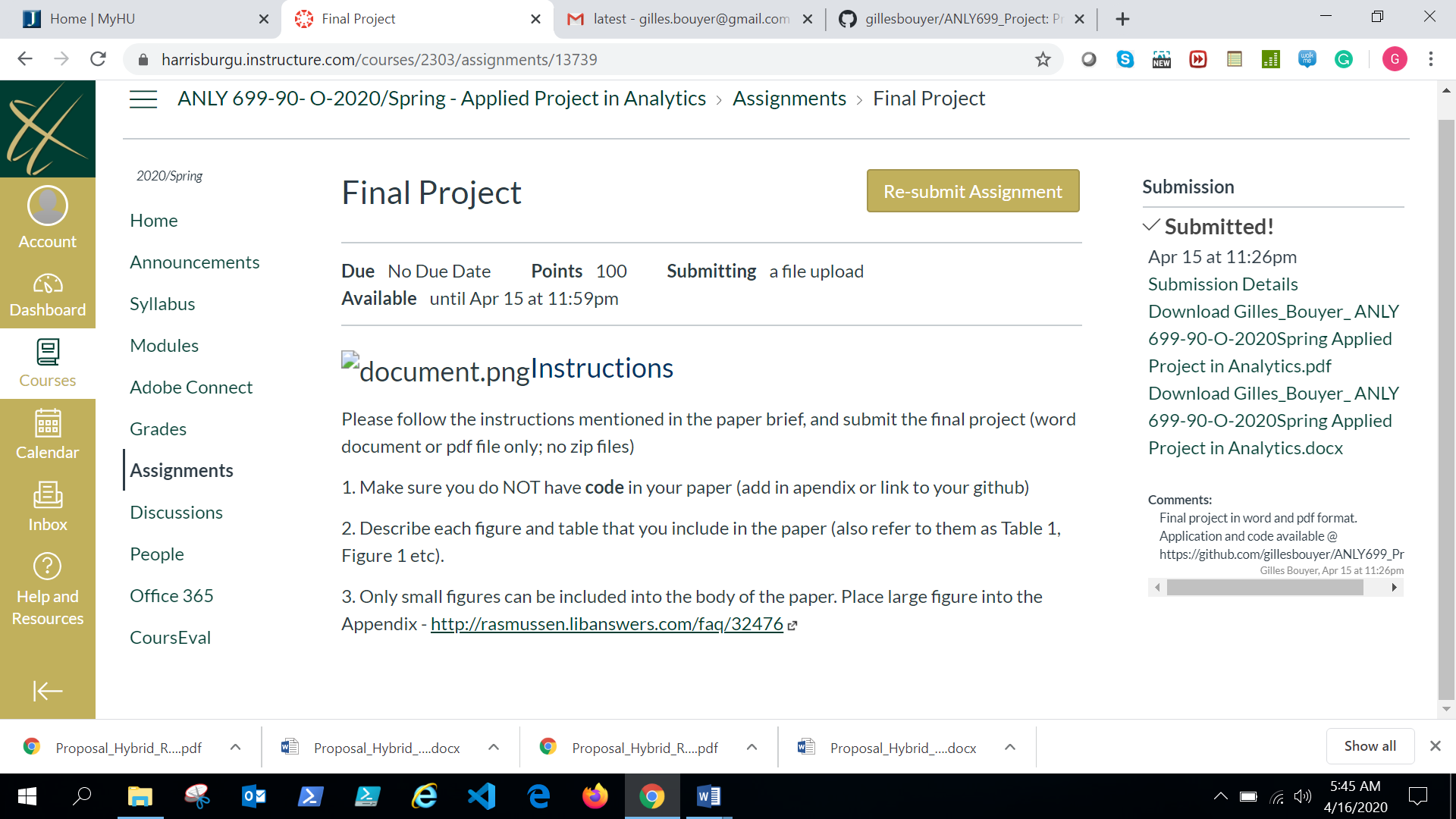


**Appendix D**

**MovieLens Data Set**

The dataset has 1,100 tags applied to the 27,000 movies for a total of 465,000 tags. The rating file is a table structured as follows: user id, movie id, rating, and timestamp—an example being 1, 2, 3.5, and 1112486027. The ratings range from 1 to 5 in intervals of .5. The timestamp is in GMT format. The Tag file is structured as follows: user id, movie id, tag, and timestamp—an example being: 18, 4141, Mark Waters, and 1240597180. In this case “Mark Waters” is the tag, associated by viewer 18 to movie 4141 at the GMT time 1240597180. This is, however, not the most important content contained in the dataset; rather, it is the genome-score.csv file that associates a tag to a movie with a weight. Its structure is movie id, tag id, and relevance—for example 1036, 519, and 0.603. The relevance has been calculated in advance with the Tag Genome project (Kula et al., 2015), which is a very good marker in this case of the relevance of tag 519 on movie 1036. The relation tag id and tag can be found in the file genome-tags.csv with the easy structure tag id and tag. (The 519 tag id represents the tag “honest” in that file.)

Two additional files are provided: movies.csv contains movie id, title, and genre; links.csv contains the links to review sites. A movie example is *Toy Story* (1995), Adventure|Animation|Children|Comedy|Fantasy, and an example of this is 1, 114709, 862, which means that the Internet Movie Database (IMDB) review of *Toy Story,* movie 1, can be found at <http://www.imdb.com/title/tt0114709/> and the moviedb review can be found at <https://www.themoviedb.org>/movie/862-toy-story. The structure of the link file is movie id, imdb id, tmdb id. In the context of recommendations, the best data source for establishing collaboration is found in ratings.csv, and the best content information is found in the genome-score.csv. Additional information on collaboration is in tags.csv that provides the values of the tags for each movie. Additional content information is embedded in the movie file where the genres are found and in the link file. The link file contains the codes (e.g., tt0114709) to access the reviews of the movie that one can then use to create more tags.



1. See https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide/ [↑](#footnote-ref-1)
2. MovieLens (<https://movielens.org>) is a web-based movie recommendation system. It was created in 1997 by GroupLens Research (<https://grouplens.org>), part of the University of Minnesota’s Department of Science and Engineering. Its goal is to centralize research on personalized recommendations. GroupLens originally created a commercial venture, and Amazon used its technology in its first recommender system. The dataset is retrievable at <http://files.grouplens.org/datasets/movielens/ml-20m.zip>. [↑](#footnote-ref-2)
3. Readers can go to [github.com/gillesbouyer/ANLY699\_Project](https://github.com/gillesbouyer/ANLY699_Project), and copy and paste “Dark Knight, The (2008)” to replicate the result displayed here. [↑](#footnote-ref-3)