User Adaptive Recommendation Interface

Kevin Yuan

Department of Computer Science, The College of Wooster

May 8, 2022

Contents

1 Abstract

Recommendation systems are present almost everywhere on the Internet. They act as tools that make browsing and content discovery much more efficient and effective. However, in the mainstream Internet, recommendation algorithms are usually hidden behind a black box implementation: the functionality and methodology of the algorithm are hidden away from users. The purpose of hiding the details of implementation are usually to simplify user experience. However, while the black box approach is very effective in many other technologies, it can be quite frustrating for users when applied to recommendation systems. The goal of this project is to design a product that gives users agency over their recommended content. In this case, the content recommended will be anime. Users of the product will be asked a personalized quiz about their preferences and the product will generate 10 anime recommendations based on three simple recommendation models.

2 Introduction

The internet is an information-rich environment, and consequently content discovery can be an arduous task for Internet users. There is too much content to view and not enough time to peruse through all of the available information. This is why the field of information retrieval (IR) has become an increasingly relevant and imperative study. Internet users need an efficient way to archive and find content online, hence there are many tools online that help users find and browse content more effectively. One such tool is recommenders: if computers could find and provide users with relevant content, personalized for users' needs, then that could save time for users. In fact, personalized recommendation systems are used everywhere on the internet. Not only can they increase user satisfaction, but good recommendations can also be profitable for businesses. For instance, e-commerce websites, like Amazon, analyze the items that the users buy and then recommend other items that the user could also like. The more accurate the recommendations, the more likely the user will purchase the goods.

Also, streaming platforms, like YouTube and Netflix, analyze a user's profile to recommend other videos or movies that they think the user might like. The more often users watch the recommended videos, then the more time they spend on the site, which equates to profit!

However, recommendation algorithms are a difficult topic to tackle. First of all, people have different preferences. Oftentimes, a person's preferences can be complicated to analyze and understand, and preferences are subjected to frequent change. There is too much variability and intricacy in user preferences that designing a single all-encompassing recommendation algorithm becomes a very challenging task. For instance, there are many different reasons why someone might like an item. Consider a theoretical user who likes the movie Skyfall. Suppose the user watched the movie Skyfall and wants to be recommended other similar content. But there are many different reasons why that person likes that particular movie. They might like the high ratings and might want to watch another critically acclaimed movie. Maybe they like spy action movies, or are fans of the James Bond franchise. Or maybe they really like Daniel Craig and just want to see more of Daniel Craig. A computer usually will not be able to determine the specific reason why that users like that movie unless they have enough data to recognize patterns in user preferences.

Generally, the consensus among researchers is to design complex and elaborate algorithms, borrowing high-technology such as machine-learning, to generate the most accurate recommendations for all users [5]. However, instead of trying to design a new or innovative algorithm that could revolutionize or improve recommendation accuracy, this project focuses on a different and more intuitive approach: giving users agency over their recommendations. What if users can choose or decide the content that they are recommended? The user can specify the type of recommendations they prefer, and the computer can apply a personalized algorithm that suits the user's input. Rather than trying to guess what the user wants, perhaps it would be more efficient to request to user input. It is not necessary to design an all-encompassing recommendation algorithm that addresses everyone's needs when the algorithms can be tailored to fit individual needs as specified.

Furthermore, the black box approach to recommendation algorithms can be irritating and negatively affect a user's experience. A black box approach involves hiding algorithm implementation and methods away from the user. The black box approach is popular because it encapsulates unnecessary and complicated details away from the user and also helps protect intellectual property. However, this approach can be very infuriating when applied to recommendation systems. It is sometimes common for users to get bombarded with recommendations that they do not want. And other times, users are not able to get the recommendations that they want. And so the uncertainty of how the recommendation algorithms work can cause confusion and vexation for users. Thus, giving users agency over the content they are recommended could potentially improve user experience, reduce computational power, and increase recommendation accuracy.

3 Recommendation models

The field of information retrieval deals with the problem of archiving and finding information. The three most popular IR methodologies are the vector space model, the inference network model, and the probabilistic model [9]. For recommender systems, the vector space model is most relevant: it involves converting an object (a document or user data) into a high-dimensional vector that can be numerically compared and analyzed [9]. This will be explained more in-depth later.

The most vital aspect of information retrieval systems is that they perform ranked retrieval, which means that the items are ranked (according to a quantitative metric) by importance or relevance [9]. Utilizing this feature makes it possible to generate recommendations for users by prioritizing the items with highest similarity and returning the top-N recommendations. However, it is important to note that not all recommender systems return top-N items. There are also systems that instead provide a predicted likeliness score of a certain item based on user preferences [7]. But the focus of this research will mainly be on recommendation systems that return top-N items. There are two basic types of recommendations: content-based models and collaborative filtering models.

3.1 Content-based models

Content-based models make recommendations solely based on the features of the items (price, color, genre, etc.). For instance, if a user likes comedy movies, then the model would prioritize recommending other comedy movies. The algorithm is designed to return items that have as many similar features as the items that the user likes. In order for this model to work, it is necessary to have some sort of data that can be used for feature analysis. The most basic form of data for content models are feature data sets or item descriptors. Item descriptors are more tricky because they usually need to be parsed through with some sort of processing technology. For instance, if the descriptors are summaries, then natural language processing will be necessary to classify the summaries. But these are not the only two ways to obtain information on items. There are also plenty of deep learning models, such as computer vision models, that can obtain or infer specific information about items when information is not explicitly provided [6].

3.2 Collaborative filtering models

Collaborative filtering models analyze user-item feedback to generate recommendations. There are two primary types of collaborative filtering approaches: memory-based and model-based.

Memory-based models use data that are stored as-is into memory [11], so recommendations are generated solely through processing the user ratings stored in memory. Memory-based models estimate the probability that a user will like a given item using either a user-based method or an item-based method [11]. A user-based method examines other similar users who have rated that particular item and returns the weighted average of the ratings of these other similar users [11]. The item-based approach is more vague because it can mean two different things. Either, it can refer to obtaining a weighted average of ratings of other similar items that a user has already rated [11], or it can refer to computing similarities between items based on how all other users rate those items [2].

The other approach to collaborative filtering is model-based: training a model with machine learning which can predict ratings of items that the test user has not yet rated [11]. There are three different ways to employ a model-based algorithm: Bayesian networks, clustering, and rule-based [11]. Bayesian networks employ probabilistic models [11]. Clustering is a classification problem that calculates the conditional probability of expected ratings based on probabilities of the user being in certain classes [11]. Finally, rule-based identifies associations between co-purchased items to recommend other strongly associated items [11].

3.3 Other models

Content-based and collaborative filtering models can be combined into hybrid models: advanced models which also employ deep learning to generate more accurate suggestions. For instance, in one particular project involving designing a recommendation system, researchers created a deep learning hybrid model where a recommendation supervisor learns from content-based filtering and collaborative filtering in order to generate suggestions [10]. Another significant recommendation approach is graph-based models that utilize transitive associations between users and items [11]. In fact, this is the approach used by YouTube to generate recommendations for a user's home page. The YouTube algorithm maps a set of seed videos, videos that the user likes, to other similar videos to create a related videos graph [3]. Then if a particular unwatched video in the related videos graph is reachable, through transitive closure, by all other videos in the seed set (by a given distance n), then that particular video is suggested to the user [3].

4 Research

The goal of this project is to design an interface that allows users to choose one out of three very basic recommendation algorithms. Since users are given agency over their recommendations, the algorithms should be simple and intuitive to the users, or else they will not understand what they are picking or choosing. The three algorithms are: a ranking recommender, a synopsis-based recommender, and an item-based recommender. All models will be implemented with python. The ranking recommender is a collaborative filtering model that returns a list of the top-N rated anime. The synopsis-based recommender is a content-based model that returns a list of the top-N anime that are most similar to the queried anime based on synopses. The final one is an item-based recommender, which is a collaborative-filtering model that returns other anime that are rated similarly by other users.

4.1 Anime data

The data used to generate recommendations is data that has been scraped from an online website called myanimelist.net and uploaded to kaggle.com, a repository of data sets. Myanimelist.net is an online anime and manga website, almost like IMDb but for anime instead of movies. While the website does not give users direct access to the media, it is a platform that allows users to create profiles and rate titles and discuss or recommend anime. Users can create a profile on the website and keep track of their watch list and ratings. Also, the website has a very comprehensive repository of almost all existing anime and manga titles with valuable descriptors and statistics for each, such as episode count, genres, anime studio and producer, plot synopsis, etc. The website also has an API that gives developers access to all this data. The dataset used for this research was scraped in 2020. The anime datasets will be converted into dataframes using the pandas library.

4.2 Implementing a simple model

The first recommender is a ranking recommender model. The general premise is that the user might want to watch popular and highly rated titles. There are two variables that are relevant to this algorithm: the average rating and the number of ratings for each anime. While the anime dataset contains an average score for each anime, it is impractical to just rank the titles based off of that score alone. This is because a community rating that is composed of only a few reviews is not as reliable as a review that is made up of thousands of reviews. The more reviews, the more representative the resulting score is for the whole community. To deal with this, the first step is to remove titles that have less than a certain threshold of reviews. For this algorithm, the threshold is set to the 50th quantile, so only the top 50% most reviewed are included; the other 50% of titles are discarded [1]. Eliminating the majority of the titles from the dataset also improves the efficiency and speed of the algorithm.

Next, the scores need to be weighted by the number of ratings. IMDb top 250 used to employ a formula that accomplishes this goal, where v = number of ratings for anime x, R = rating for anime x, m = number of votes received by an anime in the nth percentile, and C = mean score of all anime ratings [1].

$$(v/(v+m) \times R) + (m/(m+v) \times C) \tag{1}$$

After applying the formula to calculate a new weighted score for each anime title, the top-N titles with the highest weighted-scores are returned.

4.3 Implementing a content-based model

The second recommender model that the user can select is a content-based model. The general premise of this model is that the user might want to watch titles that are similar in content to anime they liked. There are many different variables and methods to evaluate the similarity of different anime. Since the anime dataset contains a plot synopsis for each anime, the plot synopsis will be used as the metric of comparison.

For this model, only the plot synopsis will be used to compare the similarity between anime. Other aspects, like genre, are not going to be considered for the sake of simplicity. Natural language processing will be used to parse and analyze the anime synopsis. For this particular model, TF-IDF will be used to process the synopsis and cosine similarity will be used to calculate the similarity between them.

TF-IDF stands for term frequency - inverse document frequency. There are two main parts: the first part is term frequency and the second part is inverse document frequency.

The first part, term frequency, is an indicator for how often a term appears in a document [8]. There are many different ways to measure term frequency. The three most common methods are count, logarithmic, and boolean. The first method involves counting the number of appearances of the term. Logarithmic is similar, but it is just a log of the count [8]. The last method, boolean, involves setting the term frequency to 1 if the term exists in the document, and 0 if the term is not present [8].

The second part, inverse document frequency, deals with useless words: stop words, like "on, at, or, the, etc." that are not meaningful but still appear frequently in documents [8]. The idea is that if a word is a common occurrence among a corpus, then that word is probably not important. The formula for inverse document frequency is $log(N/dft)$, where N is the number of documents in the corpus and dft is the number of documents in which the term t occurs [8].

Then, TF and IDF are combined together by multiplying the two values to get a TF-IDF score [8]. The TF-IDF score is high when a term t appears many times within a document, low when the term appears a few times within a document, and low when the term is frequently occurring in a corpus.

TF-IDF scores can be used to generate a high-dimensional vector for each document, where each component of that vector corresponds to a TF-IDF score of a term. By using this method, document content can be interpreted mathematically. There are two simple ways of calculating the similarity between two vectors. One method is to calculate the magnitude of the vector difference. This method is problematic: even if both vectors have similar direction, if there is a great difference in magnitude between the two vectors, then the difference between the two vectors will also be great [8]. In other words, even if the two documents are very similar in content, if one document is much longer than the other, then the magnitude of the vector difference will also be large. So this methodology would not generate the desired results. Instead, the standard way of computing similarity between two vectors is to compute the cosine similarity, which is based off of the formula for calculating the angle between two vectors [8]. The higher the similarity score, the smaller the angle between the two vectors and consequently, the more similar the two documents are. So the most similar documents are those with high similarity values. Thus, the complete algorithm for the synopsis-recommender performs the following steps: the interface takes in a queried anime, generates a high-dimensional vector for that anime using TF-IDF, and then calculates cosine similarity scores with all other anime and returns the top 10 anime with the highest similarity scores [1].

4.4 Implementing a collaborative filtering model

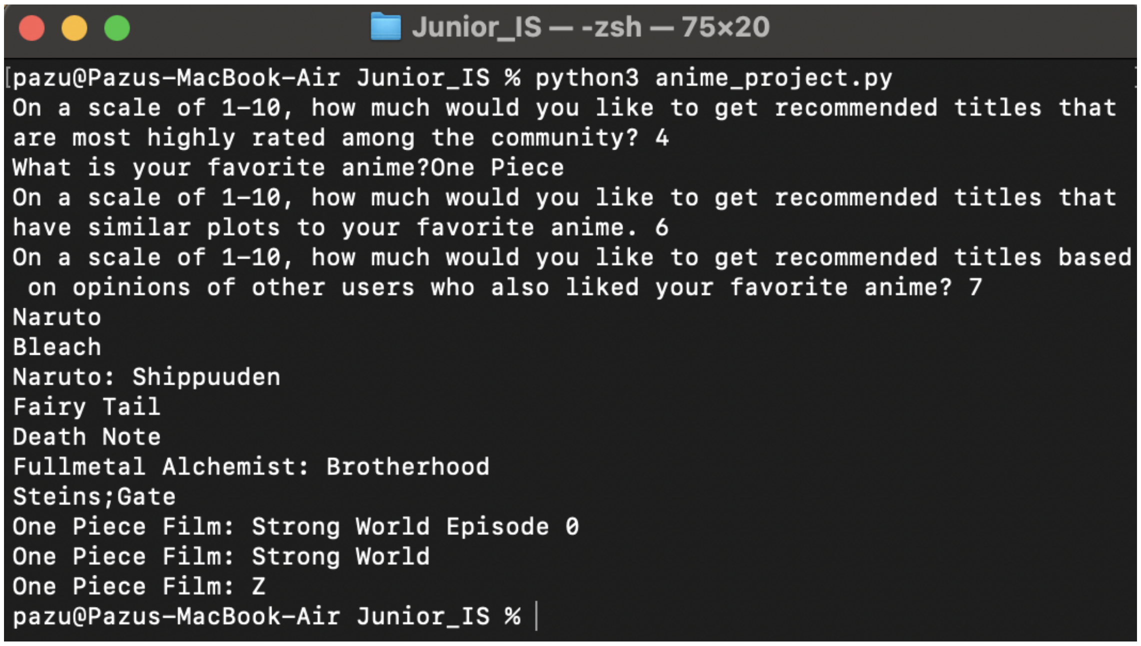
The last model that the user can choose from is an item-based collaborative filtering model. The algorithm works similarly to the content-based model since it also involves creating a high-dimensional vector. Each anime is converted into a vector, where each component represents a user and the corresponding user rating. If two vectors are close to each other (using a similarity measure such as cosine similarity or euclidean distance), then that means that they are rated similarly by all people: the same people who liked one anime also liked the other, and the same people who disliked one anime also disliked the other. By combining all vectors together, a user-item matrix is created. Since the matrix is very large, the matrix is condensed into a sparse matrix to save space [4]. So the complete item-based algorithm works as follows: the user is queried for their favorite anime, Tthen a K-nearest neighbors function, using cosine similarity, returns the n-closest anime to the user's favorite anime [4].

4.5 Discussion of product

A personalized quiz is given to the user to properly gauge their interests. The quiz is composed of four total questions: (1) On a scale of 1-10, how much would you like to get recommended titles that are most highly rated among the community? (2) What is your favorite anime? (3) On a scale of 1-10, how much would you like to get recommended titles that have a similar plot to your favorite anime? (4) On a scale of 1-10, how much would you like to get recommended titles based on opinions of other users who also liked your favorite anime?

The first, third, and fourth question gauges a user's potential interest in the ranking, synopsis-based, and item-based algorithms respectively. The second question is the query that is used for the synopsis-based and item-based algorithms. Because the user's preference inputs are in the form of ordinal variables, it is possible to weigh a user's preference for each algorithm in order to return a weighted mix of titles generated from all three algorithms.

The product was implemented in the command prompt interface. The UI is a simple set of input and output: the product prints out the questions to the terminal and the users type their answers into the terminal. After the product processes the input, it prints the recommendations onto the screen. [Fig 1.] shows an example where a user uses the product. Here, the user answers 4, 6, 7 for first, third, and fourth question respectively. And the user also specifies that their favorite anime is One Piece. In return, the product prints out ten titles, two titles from the ranking recommender, three titles from the synopsis recommender, and five titles from the item-based recommender.



```
[pazu@Pazus-MacBook-Air Junior_IS % python3 anime_project.py
On a scale of 1-10, how much would you like to get recommended titles that
are most highly rated among the community? 4
What is your favorite anime?One Piece
On a scale of 1-10, how much would you like to get recommended titles that
have similar plots to your favorite anime. 6
On a scale of 1-10, how much would you like to get recommended titles based
 on opinions of other users who also liked your favorite anime? 7
Naruto
Bleach
Naruto: Shippuuden
Fairy Tail
Death Note
Fullmetal Alchemist: Brotherhood
Steins;Gate
One Piece Film: Strong World Episode 0
One Piece Film: Strong World
One Piece Film: Z
pazu@Pazus-MacBook-Air Junior_IS %
```

Figure 1: User interface of product

4.6 Limitations and future works

There are many flaws in this product. In fact, each of the algorithms have at least one major flaw. The main issue with the ranking algorithm is that the recommendations generated are completely static: the top-N recommended titles are always the same top-N titles with the highest weighted score. This means that no matter who uses this algorithm, the output will always be the same, and that makes it a very inflexible model. A possible solution to counter this flaw is to incorporate the weighted rating aspects in the other two recommendation algorithms instead of having it as a stand-alone algorithm.

The main flaw of the content-based model is that the TF-IDF scores also factor in the names of the protagonists. It views names as unique words, and therefore, if the same name appears in two different synopses, then the content-based model interprets these two anime as being very similar, even if they are completely unrelated. For instance, consider a test user who likes the anime Bleach. Bleach is an action and fantasy based anime about a protagonist named Ichigo who fights against corrupt spirits named Hollows. Out of the top-10 recommendations from the content-based model, 3 titles are completely unrelated to

Bleach. One is a magical girl anime, another is an idol girl anime, and the last is a romance slice-of-life anime: all have a female lead with the name Ichigo. One possible way to fix this will be to find a way to include names as stop-words that gets filtered out. Or, another solution will be to filter the anime recommendations based on genre first, so that completely unrelated recommendations will not be processed by the content-based model.

The flaw for the user-based model is that the running time is too long. The original dataset, composed of around 100 million rows, is reduced to less than a third, and the algorithm still takes around two minutes to finish computing. Fixing this requires other methods or ways to reduce the amount of computations needed. A potential solution is to perform more thorough filtering of the dataset beforehand to reduce the number of computations.

5 Conclusion

The primary goal of this research project is to see if it is possible to design a recommendation product that can adapt to user input. By asking users a personalized quiz, the product gives users a feeling of control and agency over their recommendations, which alleviates the common issue of black box recommendations. There are three main algorithms that are implemented in this product: a ranking algorithm, a synopsis-based algorithm, and an item-based algorithm. The gist of all of these algorithms are relatively straightforward for users to understand; this is important because users need to be able to understand the basic inner-workings before being able to select the right choices that best reflect their preferences.

There are a lot more areas to further develop the product. For instance, the product can be more robust if the product can also recommend based off of other features, such as anime studios, voice actors, or even number of episodes. Also, the product can be further personalized if users can input multiple favorite anime instead of just one.

Moving forward, the goal is to turn this product into a full-stack online application. Instead of retrieving data from an old anime dataset, this full-stack application will scrape data directly from the myanimelist.net site. Also, this site will allow users to create profiles and save their recommendations. And hopefully, the user interface will be more engaging and appealing than a command interface.

References

[1] Python Recommender Systems: Content Based & Collaborative Filtering Recommendation Engines. `https://www.datacamp.com/community/tutorials/recommender-systems-python`, May 2020.

[2] Item-based Collaborative Filtering. `https://www.analyticsvidhya.com/blog/2021/05/item-based-collaborative-filtering-build-your-own-recommender-system/`, May 2021.

[3] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 293–296, New York, NY, USA, September 2010. Association for Computing Machinery.

[4] Everydaycodings. Anime recommendation system: Collaborative method. `https://everydaycodings.medium.com/anime-recommendation-system-collaborative-method-ca3e84ee41a0`, February 2022.

[5] Bei Hui, Lizong Zhang, Xue Zhou, Xiao Wen, and Yuhui Nian. Personalized recommendation system based on knowledge embedding and historical behavior. *Applied Intelligence*, 52(1):954–966, January 2022.

[6] Crossing Minds. What are the top recommendation engine algorithms used nowadays? `https://itnext.io/what-are-the-top-recommendation-engine-algorithms-used-nowadays-646f588ce639`, March 2020.

[7] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international*

conference on World Wide Web, WWW '01, pages 285–295, New York, NY, USA, April 2001. Association for Computing Machinery.

[8] H Schutze, CD Manning, and P Raghavan. Scoring term weighting and the vector space model. In *Introduction to information retrieval*, pages 118–119. Cambridge Univ. Press, 2008.

[9] Amit Singhal. Modern Information Retrieval: A Brief Overview. *IEEE Data Eng. Bull*, 24:35–43, 2001.

[10] Gabriele Sottocornola, Fabio Stella, Markus Zanker, and Francesco Canonaco. Towards a deep learning model for hybrid recommendation. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 1260–1264, New York, NY, USA, August 2017. Association for Computing Machinery.

[11] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 501–508, New York, NY, USA, August 2006. Association for Computing Machinery.