Building a Recommender System for Podcasts

Yamini Ananth, Katherine Wang, Abhiram Kolluri, Jafar Vohra

10 October 2022
APMA4903

Outline

Introduction & References	Who are we?
Recommender Systems	What are they? Why recommender systems?
Parts of a Modern RS	What constitutes a modern “Recommender System”?
Implementing Item-Based	Implementing Bag of Words, TF-IDF based podcast recs
Implementing User-Based	Recommending podcasts based on user preferences
Looking Ahead	Improving our model + where the field is headed

Who are we?

Yamini

- senior, seas, applied math + CS minor
- currently recruiting for data science roles

Kathy

- senior, cc, applied math + econ minor + premed
- biotech equity research + potentially med school or MD/MBA

Jafar

- Senior, SEAS, Applied Mathematics, Combined Plan
- Sports/Tech industry + data science, engineering roles

Abhiram

- senior, seas, applied math, combined plan
- healthcare/biotech data scientist + also recruiting for pm roles

What did we contribute?

Yamini

- Web scraping/data collection/data cleaning & pre-processing
- Code + slides for the Bag of Words, TF-IDF models, generating fake users, viewing model results, model improvement

Kathy

- Parts of a recommender system and pros/cons of a recommender system
- Math components of our model, ALS Matrix Factorization proof

Abhiram

- Types of recommender systems and overview of how they work
- Future trends in recommender systems

Jafar

- Collaborative Filtering code implementation and demonstration
- Pseudocode/Explanation for fake user generation and ALS Matrix Factorization
- Similarity calculation research and determination

Why do we individually care?

Yamini

- Can’t escape rec systems in day-to-day life, curious how they work
- Am a podcast enthusiast (favs: 99% Invisible, This American Life, Radiolab, Sidedoor)

Kathy

- Recommender systems are everywhere, interested in knowing how they mathematically work beyond the code
- Podcasts are long! Wondering how they might get recommended versus other content

Abhiram

- Interested in ML/DS applications that make people happier or healthier
- Podcast fan, favorites are: Lex Fridman Podcast, Duncan Trussell Family Hour, Hacks on Tap

Jafar

- Intrigued by the Netflix Prize competition
- Interested in ML/DS applications, want to broaden my perspective of ideas and projects worked on

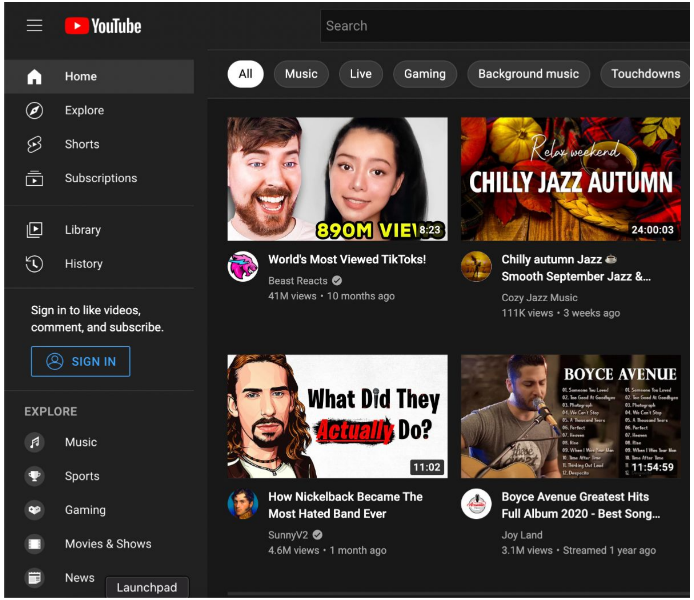
Primary References

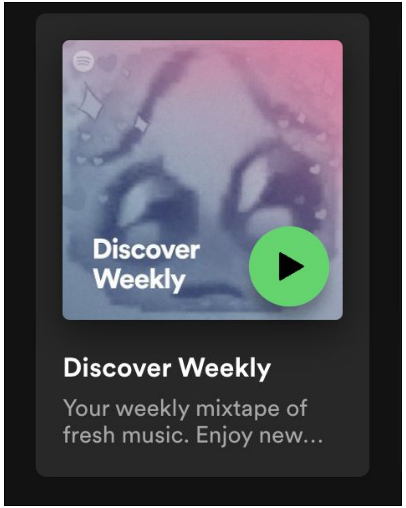
1. Covington, P., Adams, J., & Sargin, E. (2016, September). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 191-198).
2. Fethi Fkih, Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 9, 2022, Pages 7645-7669, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2021.09.014.
3. F.O. Isinkaye, Y.O. Folajimi, B.A. Ojokoh, Recommendation systems: Principles, methods and evaluation, Egyptian Informatics Journal, Volume 16, Issue 3, 2015, Pages 261-273, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2015.06.005.
4. Jones, Rosie, Zamanie, Hamin, et. al. (2021). Current Challenges and Future Directions in Podcast Information Access. SIGIR. 1. arXiv:2106.09227
5. Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John. (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of ACM World Wide Web Conference. 1. doi: 10.1145/371920.372071.

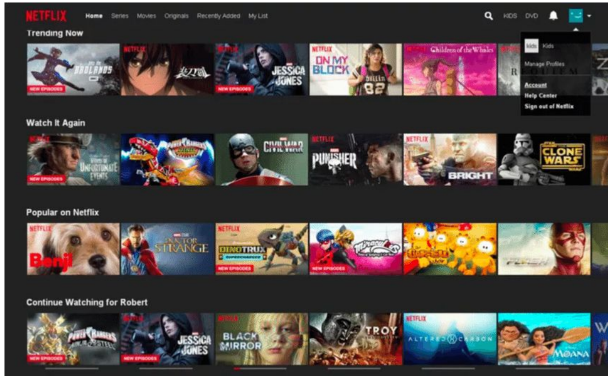
Outline

Introduction & References	Who we are, motivation, & references
Recommender Systems	What are they? Where do we see them?
Parts of a Modern RS	What constitutes a modern “Recommender System”?
Implementing Item-Based	Implementing Bag of Words, TF-IDF based podcast recs
Implementing User-Based	Recommending podcasts based on user preferences
Looking Ahead	Improving our model + where the field is headed

Recommender systems are everywhere!







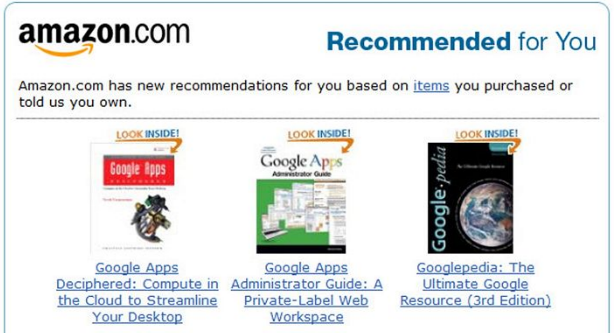


Image Source: youtube.com, spotify.com, netflix.com, amazon.com

What is a Recommender System?

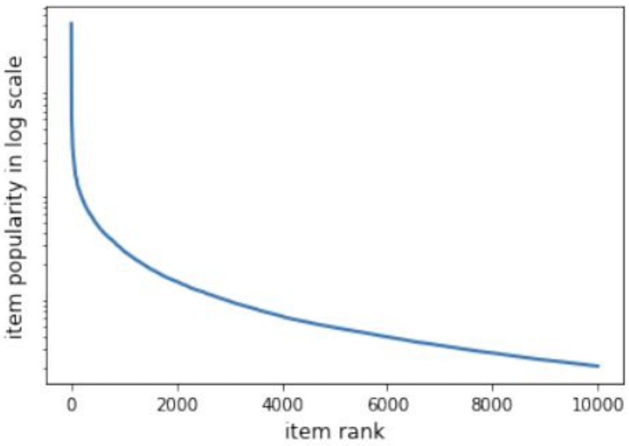
A **recommender system** is an

- Information filtering algorithm
- Leverages user profiles & item metadata to predict items a user might like
- Particularly useful when there is an overwhelming number of items in a service

… in a tech product, that means it can help:

- Predict the rating a particular user would give to an item
- Differentiate a product from its competitors, even if they share the same item space

How can we recommend podcasts?



Popularity bias (Jones et al)

- Podcasts are a commitment!
- Long tail problem: top 10% shows get ~95% of listeners
- Goal: recommend podcasts based on **how they relate** to podcasts you & listeners like you enjoy, rather than how objectively popular they are!

Image source - Jones et al https://arxiv.org/pdf/2106.09227.pdf

Outline

Introduction & References	Who we are, motivation, & references
Recommender Systems	What are they? Where do we see them?
Parts of a Modern RS	What constitutes a modern “Recommender System”?
Implementing Item-Based	Implementing Bag of Words, TF-IDF based podcast recs
Implementing User-Based	Recommending podcasts based on user preferences
Looking Ahead	Improving our model + where the field is headed

Parts of a Large-Scale Recommender System (Covington, P. et. al)

1. Candidate Generation
2. Scoring
3. Reranking

Candidate generation

- Given a query, the system generates a set of relevant candidates out of a larger dataset

- 2 common approaches
  1) **User-based** (Collaborative Filtering)
  2) **Item-based** (Content-based Filtering)

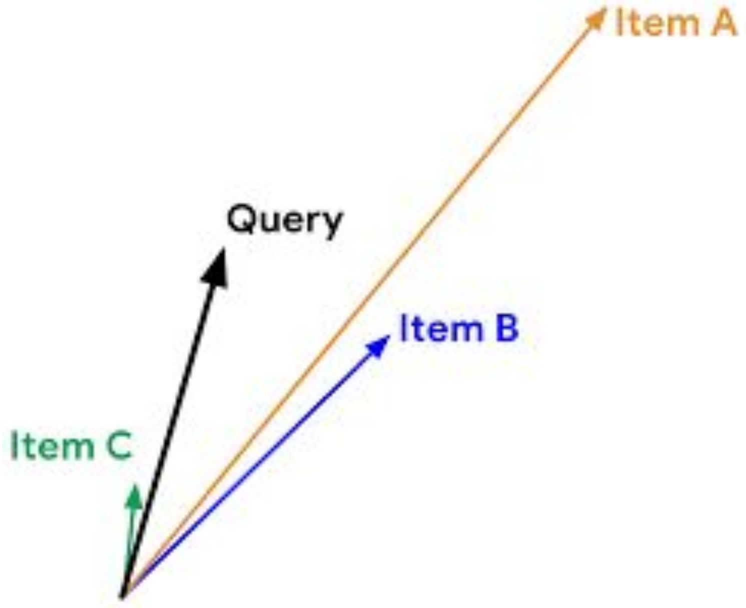


Image Source: https://developers.google.com/machine-learning/recommendation/overview/candidate-generation

Item-based/Content-based Filtering

- **Content-based Filtering** uses the content itself, how similar it is to other content, and other metadata to make recommendations
- They recommend a set of items that are comparable to the ones which the user liked in the past.
- **Example:** If user A watches two cute cat videos, then the system can recommend cute animal videos to that user.

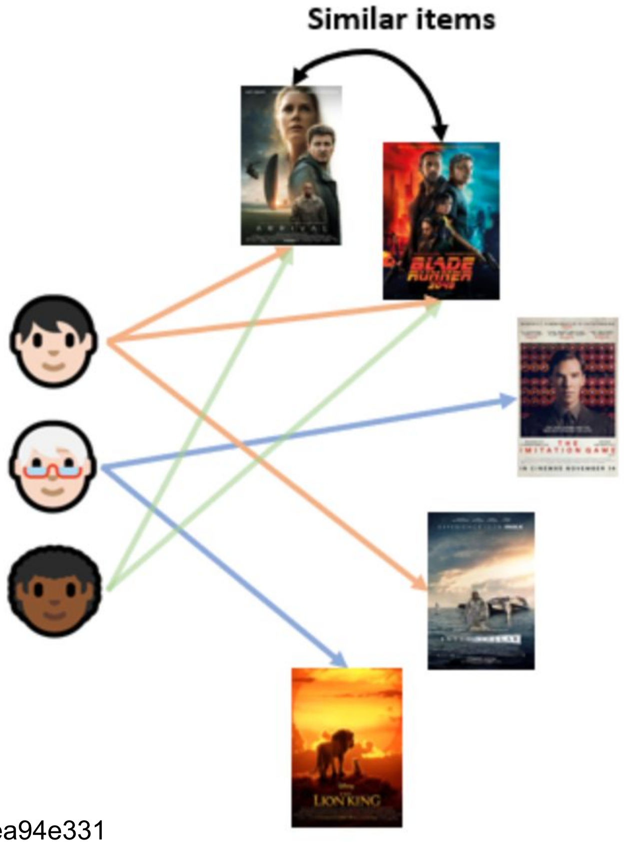


Image Source: https://towardsdatascience.com/how-does-collaborative-filtering-work-da56ea94e331

Pros and Cons of Item-based/Content-based Filtering

PROS

- Since the recommendations are specific to the item, the model doesn't need data about other users
  - This sometimes makes it easier to scale to more users
- The model can capture the specific interests of a user, and recommend niche items specific to the user

CONS

- The model can only make recommendations based on the existing interests of the user
- There tends to be a lower diversity of recommendations compared to user-based collaborative filtering

Overall, there also tends to be a low computational cost of finding neighbors in the similarity matrix.

User-based/Collaborative Filtering

- **Collaborative Filtering** recommends items by identifying other users with similar tastes
    - It uses their opinion to recommend items to the active user.
- **Example:** If user A is similar to user B, and user B likes video 1, then the system can recommend video 1 to user A (even if user A hasn't seen any videos similar to video 1).

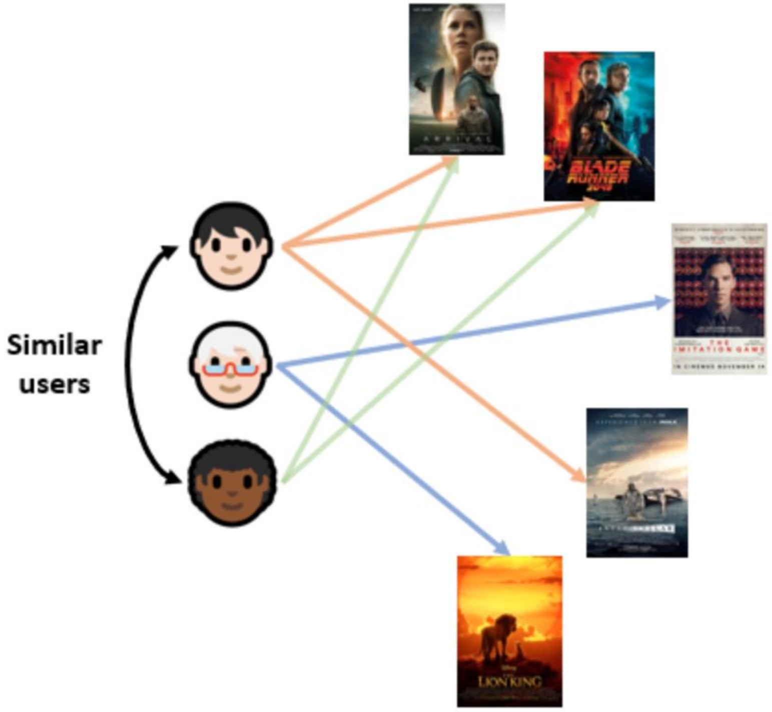


Image source: https://towardsdatascience.com/how-does-collaborative-filtering-work-da56ea94e331

Pros and Cons of User-based/Collaborative Filtering

PROS	CONS

• Higher diversity in recommendations
• Can help users discover new interests
    ○ In isolation, the system may not know the user is interested in a given item, but the model might still recommend it since similar users are interested in it
	• Usually more users than recommendations
	• User data is required to build profiles
	• **Cold start problem**
	    ○ There may be no to little information on a new user's preferences, meaning there's nothing to compare with

System doesn't need contextual features, only a feedback matrix for the bare minimum case! No domain knowledge needed.

Determining Similarity

We want to determine how similar two vector representations of data are!

- A similarity measure $s: E \times E \to R$ takes a pair of embeddings, returns a scalar measuring their similarity.
- Given a query $q \in E$, the system looks for items $x \in E$ that are close to q – with high similarity $s(q,x)$.
- Select the k most similar items to our query item as part of our candidate pool

Examples of Similarity Measures

Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

- Calculates cosine
- Depends on angle, not magnitude
- Great for non-normalized vectors & high-dim data!

Pearson Correlation Coefficient

$$PCC\left(u, v\right)$$

$$\frac{\sum_{i \in I_{uv}} \left(r_{ui} - \overline{r_u}\right)\left(r_{vi} - \overline{r_v}\right)}{\sqrt{\sum_{i \in I_{uv}} \left(r_{ui} - \overline{r_u}\right)^2}\sqrt{\sum_{i \in I_{uv}} \left(r_{vi} - \overline{r_v}\right)^2}}$$

- Classic linear correlation between two datasets
- Correlation sign determined by regression slope
- Ratio of the means of each dataset

Euclidean Distance

$$s(q, x) = \|q - x\|$$

$$= \left[\sum_{i=1}^{d} (q_i - x_i)^2\right]^{\frac{1}{2}}$$

- Smaller distance means higher similarity
- Euclidean distance will not be effective in deciding which vectors are similar to each other if same norm

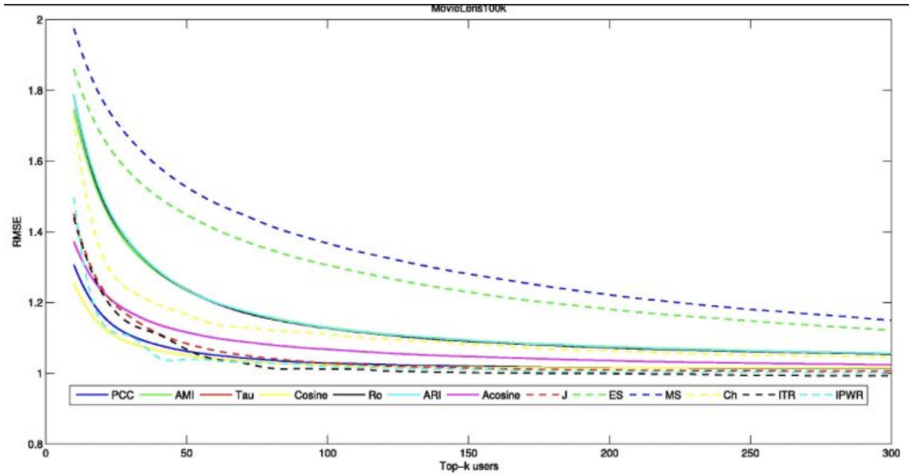
Jaccard Coefficient

$$J\left(u, v\right) = \frac{|u \cap v|}{|u \cup v|}$$

- Ratio of the number of elements shared between two sets to the total elements
- Good for cases where duplication does not matter

Efficacy of Cosine Similarity

Image Source: Fkih, F. et. al. Similarity measures for Collaborative Filtering-based Recommender Systems (fig. 8a)



- Fkih et. al found combining multiple other methods can yield marginally better results in the long tail
- Cosine similarity attains optimum in a few top-k items
- Cosine similarity = powerful metric with a simple & fast implementation

Cosine similarity is powerful, yet simple!

Parts of a Large-Scale Recommender System

1. Candidate Generation
2. Scoring
3. Reranking

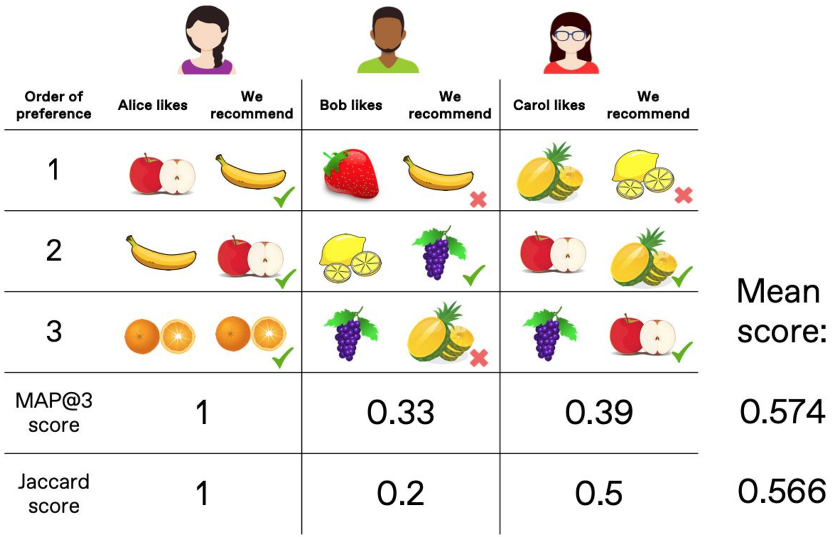
Scoring

After candidate generation, another model scores and ranks the generated candidates to select the set of items to display.

- The recommendation system may have **multiple candidate generators** from different sources
- The system **scores** and **ranks** candidates by a single model

Why the Candidate Generator is not used to score

- Some systems rely on **multiple candidate generators.**
- With a smaller pool of candidates, the system can afford to use **more features** and a more complex model that may better capture context.



Source:
https://www.kdnuggets.com%2F2020%2F02%2Fcomparing-apples-oranges-bananas.html&psig=AOvVaw1wE_I1HcagC1-lMYbwandr&ust=1664913243743000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCMjMk6brxPoCFQAAAAAdAAAAABAN

Parts of a Large-Scale Recommender System

1. Candidate Generation
2. Scoring
3. Reranking

Reranking

In the final stage of a recommendation system, the system can re-rank the candidates to consider additional criteria or constraints (like a new click, new item, etc).

- There are many ways that the system can choose to re-rank candidates:
  - Filtering to remove some candidates
  - Modifying the score as a function of certain variables
- The model can also be adapted by training it for freshness, diversity, or fairness

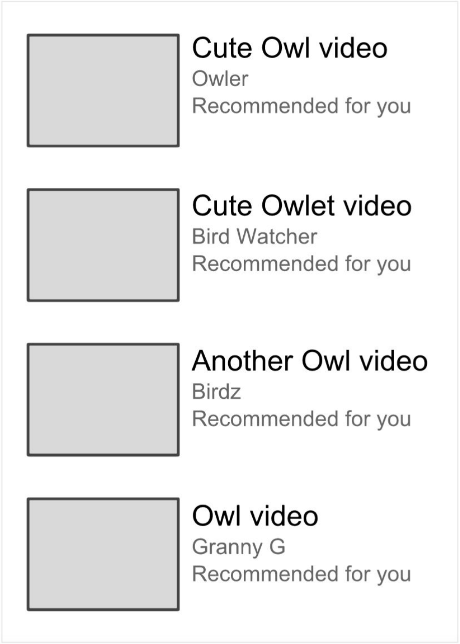


Image source: https://developers.google.com/machine-learning/recommendation/dnn/re-ranking

Outline

Introduction & References	Who we are, motivation, & references
Recommender Systems	What are they? Where do we see them?
Parts of a Modern RS	What constitutes a “Recommender System”?
Implementing Item-Based	Implementing Bag of Words, TF-IDF based podcast recs
Implementing User-Based	Recommending podcasts based on user preferences
Looking Ahead	Improving our model + where the field is headed

Podcast Recommender Implementation Overview

Goal

Given one podcast you like, can we recommend similar podcasts out of the ~4300 top podcasts?

Idea 1
(Item-based filtering)

If you like one podcast, you'll like other podcasts that are similar

Idea 2
(Collaborative filtering)

Users with similar tastes may like similar podcasts

Check out our code here!

Steps to Create Item-Based Podcast Recommender System

1. Collect data
2. Pre-process data
3. Create bag-of-words representation of podcasts
4. Create TF-IDF representation of podcasts
5. Compare results

Step 1: Data Collection

- Scraped metadata on top 300 podcasts across top 15 categories on Apple Podcasts
- ~4300 total, after accounting for duplicates across categories

Data Collected

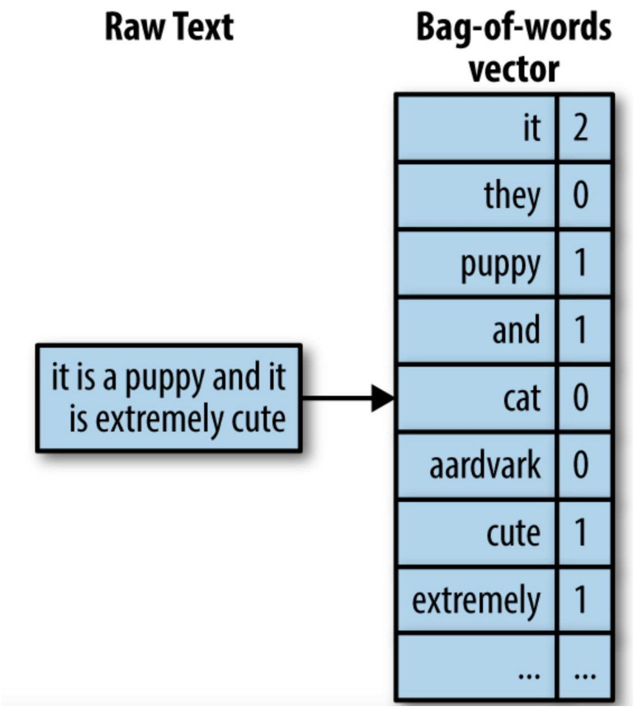
- Title (text)
- Producer (text)
- Description (text)
- 6 Recent Episode Titles (text)
- 6 Recent Episode Descriptions (text)

Step 2: Pre-processing data

Task	Example
● Filtered out URLs, special characters	● No more “https://” or “&%*(#”
● Tokenized (separated each word into its own string)	● “I love the economy” → [“i”, “love”, “the”, “economy”]
● Removed stop-words	● Remove ‘a’, ‘my’, ‘your’, ‘the’, etc
● Lemmatized	● ‘economy’, ‘economical’ and ‘economist’ → ‘econ’
“Explaining the economy! Subscribe at https://podcasts.apple.com/…”	[‘explain’, ‘econ’, subscribe’]

Step 3: Creating the Bag-of-Words Model

- **Ignores the order of the words** and only considers each word’s **frequency** in a given piece of data
- The “bag of words” is a **fixed-length vector**–the length of the vocabulary of known words–where each entry of the vector denotes a count of that word
- Treats all words **independently**



Step 4: Creating a TF-IDF Model

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

- Gives each word in the text an **assigned weight**
- The frequency of a term in a document is calculated (Term Frequency) and is **penalized** by that same term appearing in every other document
- Each word’s TF-IDF relevance is normalized, so frequency(word) = probability(word)

Step 5: Use NLP Models to Generate Results

Generate Vectors

Generate embeddings of podcast text using nlp technique

Calculate Similarity

Compute cosine similarity between each of the podcasts in our dataset

Find Similar Items

For a given item, check the similarity matrix and identify the podcasts with the n highest cosine similarities

Overlap between the models' recommendations?

```
Recs for The Daily:
    Recommended by both tf-idf and cv:
        The Daily 202's Big Idea
        Impeachment Inquiry: Updates from The Washington Post
        The 11th Hour with Brian Williams
        The Takeaway
        Article II: Inside Impeachment
        Impeachment: A Daily Podcast
    Uniqely recommended by tf-idf:
    Uniqely recommended by cv:
```

Total agreement between models

Is divergence between models a signal that the recommendations are not high quality?

```
Recs for The Joe Rogan Experience:
    Recommended by both tf-idf and cv:
    Uniqely recommended by tf-idf:
        Jordan Peterson Interviews & Speeches
        Revisionist History
        Ari Shaffir's Skeptic Tank
        MILLION DOLLAR LIFE LESSONS
        The Horror of Dolores Roach
        Malcolm Gladwell, Revisionist History: Special Event
    Uniqely recommended by cv:
        3 Books With Neil Pasricha
        The Creative Penn Podcast For Writers
        1001 Heroes, Legends, Histories & Mysteries Podcast
        The Ground Up Show
        1001 Classic Short Stories & Tales
        1001 Stories For The Road
```

Total divergence between models

Step 5: Compare Results

```
Recommendations for The Daily:
	Impeachment Inquiry: Updates from The Washington Post
	Impeachment: A Daily Podcast
	The Takeaway
	Article II: Inside Impeachment
	The Daily 202's Big Idea
	The 11th Hour with Brian Williams
Recommendations for Murder, etc.:
	Criminology
	Murderville
	Unsolved Murders: True Crime Stories
	Murder Minute
	Don't Talk to Strangers
	True Crime All The Time Unsolved
Recommendations for This American Life:
	The Stoop Storytelling Series
	The Story Home Children's Audio Stories
	Spooky Boo's Scary Story Time
	The Story Behind
	This is the Gospel Podcast
	1001 Heroes, Legends, Histories & Mysteries Podcast
Recommendations for Call Her Daddy:
	Stiff Socks
	Two Judgey Girls
	NAKED with Catt Sadler
	Slay Girl Slay
	Hot Marriage. Cool Parents.
	Safe For Work
Recommendations for The Joe Rogan Experience:
	The Creative Penn Podcast For Writers
	1001 Classic Short Stories & Tales
	3 Books With Neil Pasricha
	The Ground Up Show
	1001 Stories For The Road
	1001 Heroes, Legends, Histories & Mysteries Podcast
```

Bag of Words results

```
Recommendations for The Daily:
	Impeachment Inquiry: Updates from The Washington Post
	The 11th Hour with Brian Williams
	The Daily 202's Big Idea
	Article II: Inside Impeachment
	Impeachment: A Daily Podcast
	The Takeaway
Recommendations for Murder, etc.:
	Murder Minute
	Criminology
	Murderville
	Unsolved Murders: True Crime Stories
	Don't Talk to Strangers
	True Crime All The Time Unsolved
Recommendations for This American Life:
	Experimental Brewing
	1A
	Through the Looking Glass: A LOST Retrospective
	The Grave Talks | Haunted, Paranormal & Supernatural
	Darkness Prevails Podcast | TRUE Horror Stories
	BeerSmith Home and Beer Brewing Podcast
Recommendations for Call Her Daddy:
	hey, girl.
	Girls Night with Stephanie May Wilson
	Stiff Socks
	Fierce Girls
	Becoming Something with Jonathan Pokluda
	Two Judgey Girls
Recommendations for The Joe Rogan Experience:
	MILLION DOLLAR LIFE LESSONS
	Malcolm Gladwell, Revisionist History: Special Event
	The Horror of Dolores Roach
	Jordan Peterson Interviews & Speeches
	Revisionist History
	Ari Shaffir's Skeptic Tank
```

TF-IDF results

At first glance, results seem reasonable!

Outline

Introduction & References	Who we are, motivation, & references
Recommender Systems	What are they? Where do we see them?
Parts of a Modern RS	What constitutes a “Recommender System”?
Implementing Item-Based	Implementing Bag of Words, TF-IDF based podcast recs
Implementing User-Based	Recommending podcasts based on user preferences
Looking Ahead	Improving our model + where the field is headed

Steps to Create User-Based Podcast Recommender

1. Create fake user data
2. Implement a collaborative filtering algorithm - Alternating Least Squares Matrix Factorization - to train & test a model
3. Generate recommendations
4. Validate results

Step 1: Generating Fake User Ratings

```
def generate_user_ratings(users_count):
    for idx, user in users_count:
        ratings = list
        reviewed = set
        quantity_rated = random integer between 5 and 20
        for i in quantity_rated:
            podcast = random podcast from all podcasts
            while (podcast in reviewed):
                Add podcast to reviewed
                rating = random integer between 1 and 5
                Append to ratings list
        create user dataframe with podcasts and ratings
        append to user_ratings
    return dataframe of user_ratings
```

Assumptions:

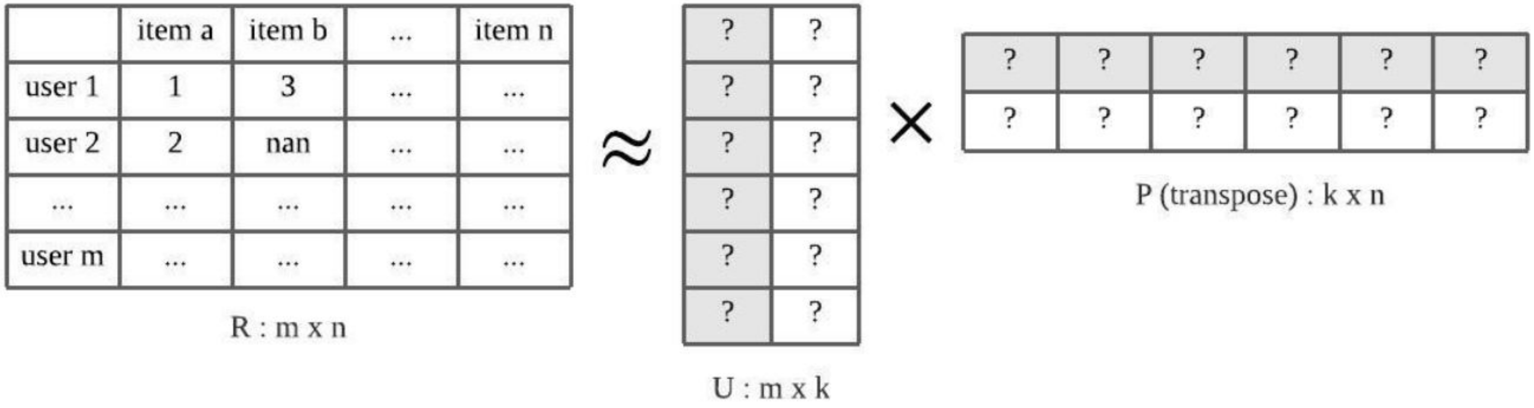
- User selects podcasts to rate independently
- User selects ratings independently
- Users actually rate in a range from 1-5
- User can only have 1 rating per podcast

Step 2: Applying Matrix Factorization

- Each user can be described by *k latent features*.
  - Ex) Feature 1 = how much each user likes news podcasts.
- Each podcast can also be described *k* latent features.
  - Ex) feature 1= how “newsy” is this podcast?
- Multiply each feature of the user by the corresponding feature of the podcast and add everything together, this will be a good approximation for the rating the user would give that podcast.

Step 2: Alternate Least Squares Matrix Factorization

The goal of matrix factorization is to separate the utility matrix into the **user latent matrix** and the **product latent matrix**, such that **R = U x P**



In Alternative Least Square (ALS), the factorization model is optimized in an iterative process

How this works: 1) Defining the Objective Function

$$RMSE = \sqrt{(real - prediction)^2/n}$$

where real = R, prediction = U*P^T

- The objective function is defined using the loss function- we chose RMSE
- Measure of how spread out the residuals are
- Tells you how concentrated the data is around the line of best fit

How this works: 2) Latent Factors

Assume there are **m** users and **n** items, R = m * n, U = m * k, P = n * k, where k is the latent factors

$$
\begin{aligned}
loss &= min(real - prediction)^2 \\
&= min(R - U * P^T)^2 \\
&= min \sum_{x,y} (R_{x,y} - U_x * P_y^T)^2
\end{aligned}
$$

…then add the l2 norm to our objective function to avoid overfitting

$$loss = min \sum_{x,y} (R_{x,y} - U_x * P_y^T)^2 + \lambda(\|U\|^2 + \|P\|^2)$$

How this works: 3) Partial Differentiation

Take the partial derivative with respect to U and P

- By fixing one, we can optimize the other one
- Iteratively alternate the latent matrix U and P to optimize the utility matrix factorization.

$$\frac{\partial loss}{\partial U} = 0$$

$$= \frac{\partial}{\partial U} \sum_{x,y} (R_{x,y} - U_x * P_y^T)^2 + \lambda(\|U\|^2 + \|P\|^2) = 0$$

$$= -2 \sum_{x,y} (R_{x,y} - U_x * P_y^T) P_y + 2\lambda U_x = 0$$

$$= -(R_x - U_x^T P^T) P + \lambda U_x^T = 0$$

$$= U_x^T = R_x P (P^T P + \lambda I)^{-1}$$

$$\frac{\partial loss}{\partial P} = 0$$

$$= \frac{\partial}{\partial P} \sum_{x,y} (R_{x,y} - U_x * P_y^T)^2 + \lambda(\|U\|^2 + \|P\|^2) = 0$$

$$= P_y^T = R_y U (U^T U + \lambda I)^{-1}$$

Step 3: Implementing ALS Matrix Factorization for Collaborative Filtering

- Create a train/test split of 0.7/0.3
- Set the maximum number of ALS iterations=10
- Set *k*-latent-factors between 10-100
- Run 3-fold cross-validation
- Identify which *k* produced least error in test data (k=100)
- Use the best model to generate predictions for the whole dataset

Step 4: View Results

```
User #17 Profile:
5 reviews
Top 5 shows:
       FreshRN
       Impeachment: A Daily Podcast
       6 Minute Grammar
       CBS Evening News -- Full Audio
       RadioWest
                ..
We recommend the following:
       Dad Tired
       Accidental Tech Podcast
       Castology
```

```
User #15 Profile:
20 reviews
Top 5 shows:
       Global News Podcast
       The G Club
       The Podcast History Of Our World
       Lunch Therapy
       The Remnant with Jonah Goldberg
                ..
We recommend the following:
       Pod Save the People
       Peace Of Mind with Bhi Bhiman
       The Curbsiders Internal Medicine Podcast
```

Not bad!!!

Outline

Introduction & References	Who we are, motivation, & references
Recommender Systems	What are they? Where do we see them?
Parts of a RS	What constitutes a “Recommender System”?
Implementing Item-Based	Implementing Bag of Words, TF-IDF based podcast recs
Implementing User-Based	Recommending podcasts based on user preferences
Looking Ahead	Improving our model + where the field is headed

How this recommender system can be improved

- Exploring different similarity methods
- Full podcast episode transcripts in the dataset
- Using a more complex embedding mode and/or custom embedding model
- Real user data/non-naive fake user data
- Combining multiple models into a hybrid recommendation system

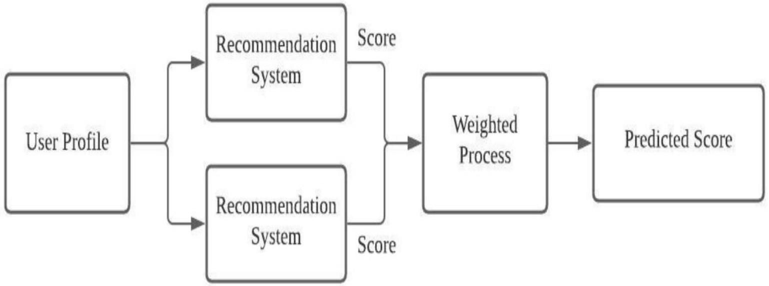


Figure 1. Weighted Hybrid Recommendation System Image by author

Future Trends with Recommender Systems

- Using **implicit feedback** from users such as listen duration, pause/plays, shares, etc. often produces higher quality results
- Adding time-weighting, as newer/fresher content may be more popular in the short term
- Using personal/demographic data, like location
- Exploring tradeoff between user security/privacy and accurate recommendations

Image Source: https://www.heritageconsultants.com/the-future-is-loading/

Future trends: Community-Based Collaborative Filtering

- **Tracking user-community behavior**
  - Who people interact with can give a lot of information on their preferences
- Utilizing social network graphs to find areas of shared interests have promising results

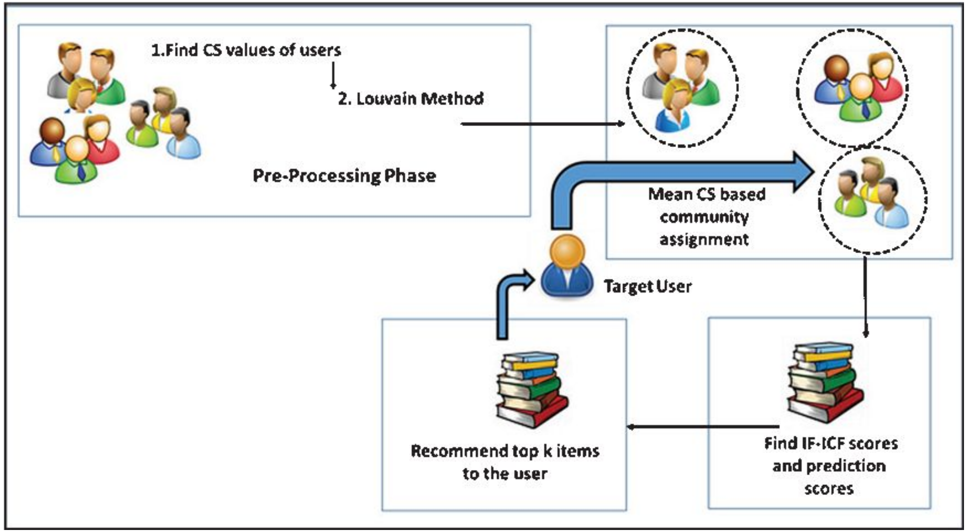


Image source:
https%3A%2F%2Fwww.semanticscholar.org%2Fpaper%2FCCFRS-Community-based-Collaborative-Filtering-Sharma-Bedi%2Fdb30b9ce2619ee7e200cf220689d072ed41549c2&psig=AOvVaw2iSw3XHPYEd9d8yd0_iyv_&ust=1665074129445000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCIDXqtLCyfoCFQAAAAAdAAAAABAE

Issues with Recommender Systems to fix in the future

- **Echo chambers:** systems that show users similar content to what they already like
- **Polarization:** recommendation systems can reinforce group identity and promote conflict between group
- Negative externalities on society
- Hyper-personalized recommendation systems can make users feel like their privacy is being violated

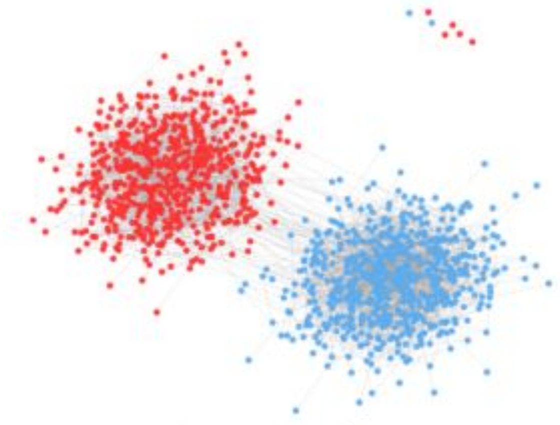


Image Source: https://commons.wikimedia.org/wiki/File:Assortative_Case_of_the_SBM.png

Mitigating social impacts of recommender systems

- Utilizing methods to decide which news stories should be shown to a user
  - Stories that are untrue may be the most popular
- Focusing less on personalization and more on allowing for alternative viewpoints to combat polarization

Image source:
www.cnn.com%2Fvideos%2Fpolitics%2F2016%2F11%2F17%2Ffake-news-social-media-tapper-dnt-lead.cnn&psig=AOvVaw31ncXaptd_6SI8QyckikAk&ust=1665074424676000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCKC3j9_DyfoCFQAAAAAdAAAAABAI

thank you!