Letterboxd Recommendation System

Laya Gollamudi, Sierra Martinez-Kratz and Geraldine Nina Montano

APMA 4903: Senior Seminar

Table of contents

About us

About us

Laya Gollamudi

SEAS

Sierra Martinez-Kratz

CC

Geraldine Nina

SEAS

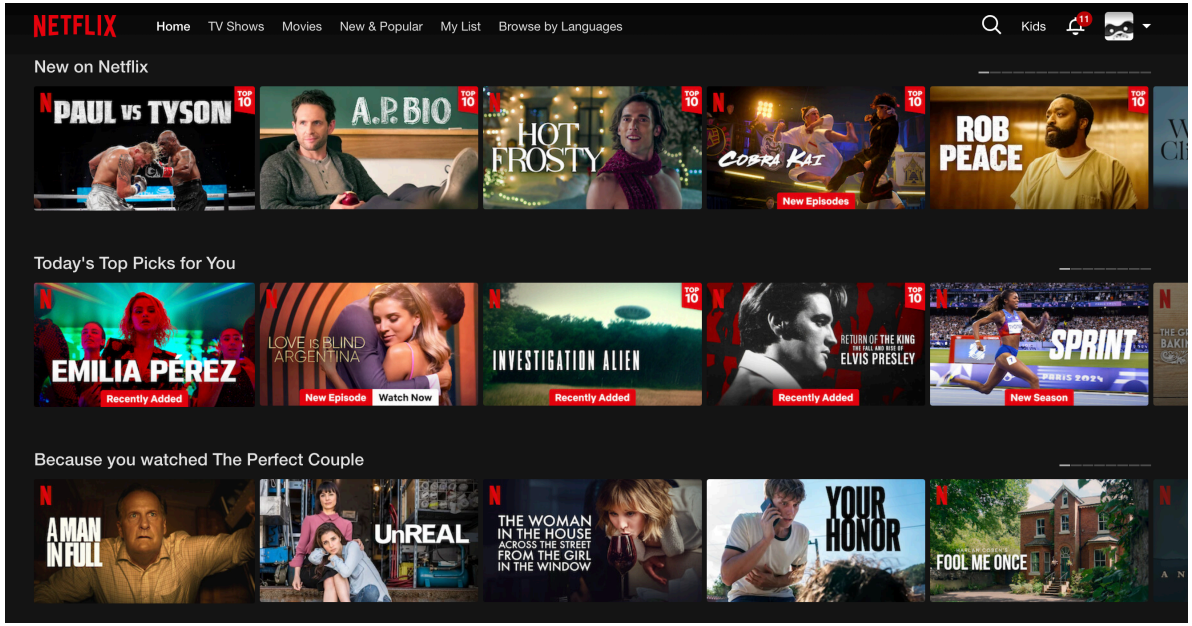
Why we care?

We are interested in—

- Data science
- Machine learning
- Letterboxd
- Movies!

Why *you* should care?

- Recommendation systems suggest products, services, or content based on user preferences.
- Examples:
  - Movie recommendations (Netflix).
  - Product suggestions (Amazon).
  - Music streaming (Spotify).
- Understanding recommendation systems helps you understand why certain things are recommended to you



References (1/3)

- Baad, D. (2020a). Sentiment Classification for Restaurant Reviews using TF-IDF. Retrieved from https://medium.com/swlh/sentiment-classification-for-restaurant-reviews-using-tf-idf-42f707bfe44d
- Burred, J. (2014). Detailed derivation of multiplicative update rules for NMF. https://www.jjburred.com/research/pdf/jjburred_nmf_updates.pdf
- cheng, jianqiao. (2023, January 9). Understanding the Two-Tower Model in Personalized Recommendation Systems. Hackernoon.com. https://hackernoon.com/understanding-the-two-tower-model-in-personalized-recommendation-systems
- Colace, F., Conte, D., De Santo, M., et al. (2022a). A content-based recommendation approach based on singular value decomposition. *Connection Science, 34*(1), 2158–2176. https://doi.org/10.1080/09540091.2022.2106943
- Colace, F., Conte, D., De Santo, M., et al. (2022b). Recommender systems: A novel approach based on singular value decomposition. *International Journal of Electrical and Computer Engineering (IJECE), 12*(6), 6513–6521. https://doi.org/10.11591/ijece.v12i6.pp6513-6521
- Hug, N. (2020). Surprise: A Python library for recommender systems. *Journal of Open Source Software, 5*(52), 2174. https://doi.org/10.21105/joss.02174
- Kim, H., Park, H. (2008). Nonnegative Matrix Factorization Based on Alternating Nonnegativity Constrained Least Squares and Active Set Method. SIAM Journal on Matrix Analysis and Applications, 30(2), 713–730. https://doi.org/10.1137/07069239x
- Dang, C. N., Moreno-García, M. N., Prieta, F. D. l. (2021). An Approach to Integrating Sentiment Analysis into Recommender Systems. Sensors, 21(16), 5666. https://doi.org/10.3390/s21165666

References (2/3)

- Learner, S. (n.d.). Letterboxd recommendations. GitHub. Retrieved November 17, 2024, from https://github.com/sdl60660/letterboxd_recommendations/blob/main/README.md
- Lee, D. D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems, 13*, 556–562.
- Murray, I. (2022). Netflix Prize. Retrieved November 21, 2024, from University of Edinburgh Machine Learning and Pattern Recognition course notes:https://mlpr.inf.ed.ac.uk/2022/notes/w8b_netflix_prize.pdf
- Netflix. (2022). How Netflix's Recommendations System Works. Netflix Help Center.https://help.netflix.com/en/node/100639
- Pinoli, P., et al. (2015). Computational algorithms to predict gene ontology annotations. *BMC Bioinformatics, 16*(S6), Article S4. https://doi.org/10.1186/1471-2105-16-s6-s4
- Pu, L., & Faltings, B. (2013). Understanding and improving relational matrix factorization in recommender systems. *Proceedings of the 2013 ACM International Conference on Recommender Systems* (pp. 41–48). https://doi.org/10.1145/2507157.2507178
- Rajput, S., Mehta, N., Singh, A., Hulikal Keshavan, R., Vu, T., Heldt, L., ... Sathiamoorthy, M. (2023). Recommender systems with generative retrieval. Advances in Neural Information Processing Systems, 36, 10299-10315.
- Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009)*, 452–461.

References (3/3)

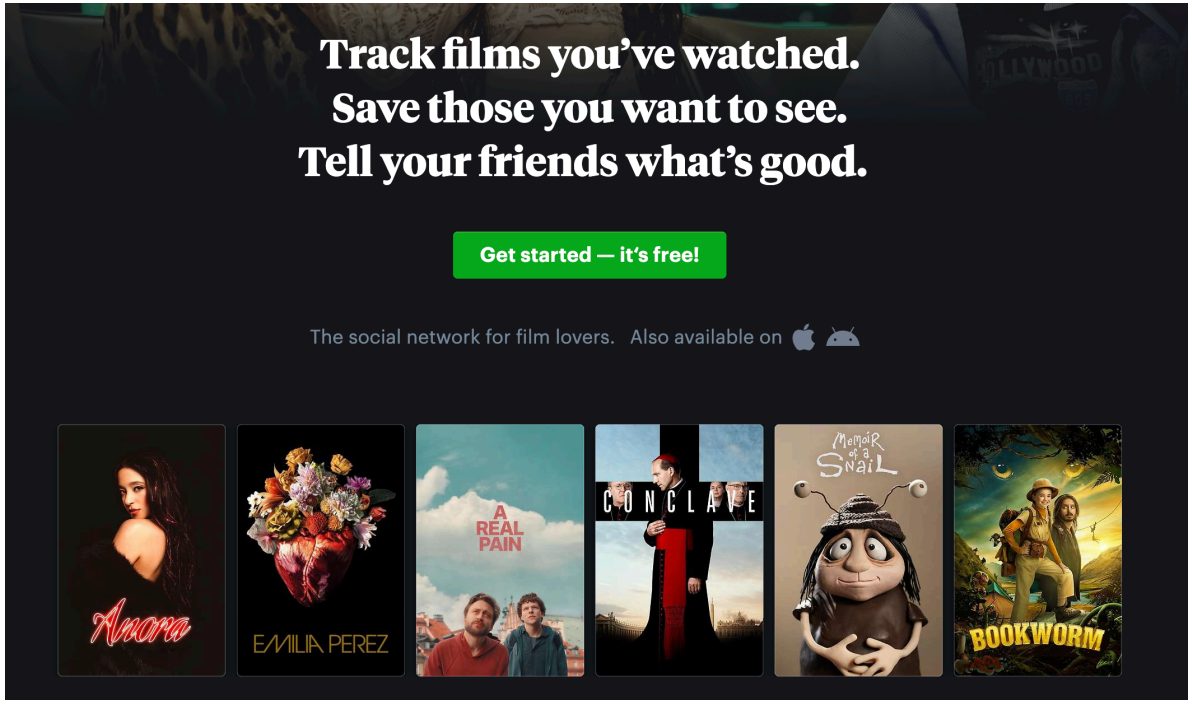
- Wankhade, M., Rao, A.C.S. & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review, 55*, 5731–5780. https://doi.org/10.1007/s10462-022-10144-1
- How Tiktok recommends content. (n.d.). Retrieved from https://support.tiktok.com/en/using-tiktok/exploring-videos/how-tiktok-recommends-content
- Barnard, J. (2024). What are word embeddings? Retrieved from https://www.ibm.com/topics/word-embeddings#%3A~\protect\protect\leavevmode@ifvmode\kern+.2222em\relaxtext=Word%20embeddings%20are%20a%20way,relationships%20among%20the%20corresponding%20words.
- Recommenders Team. (n.d.). Cornac BPR deep dive. GitHub. Retrieved [Insert Date], from https://github.com/recommenders-team/recommenders/blob/main/examples/02_model_collaborative_filtering/cornac_bpr_deep_dive.ipynb
- E.M. Voorhees (1999). "Proceedings of the 8th Text Retrieval Conference" (PDF). TREC-8 Question Answering Track Report. pp. 77.
- Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, Tie-Yan Liu. 2013. A Theoretical Analysis of Normalized Discounted Cumulative Gain (NDCG) Ranking Measures. In Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013). pp 5.
- Zhai, J., Liao, L., Liu, X., Wang, Y., Li, R., Cao, X., ... Shi, Y. (2024). Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. arXiv preprint arXiv:2402.17152.

Problem Motivation

What is Letterbox?

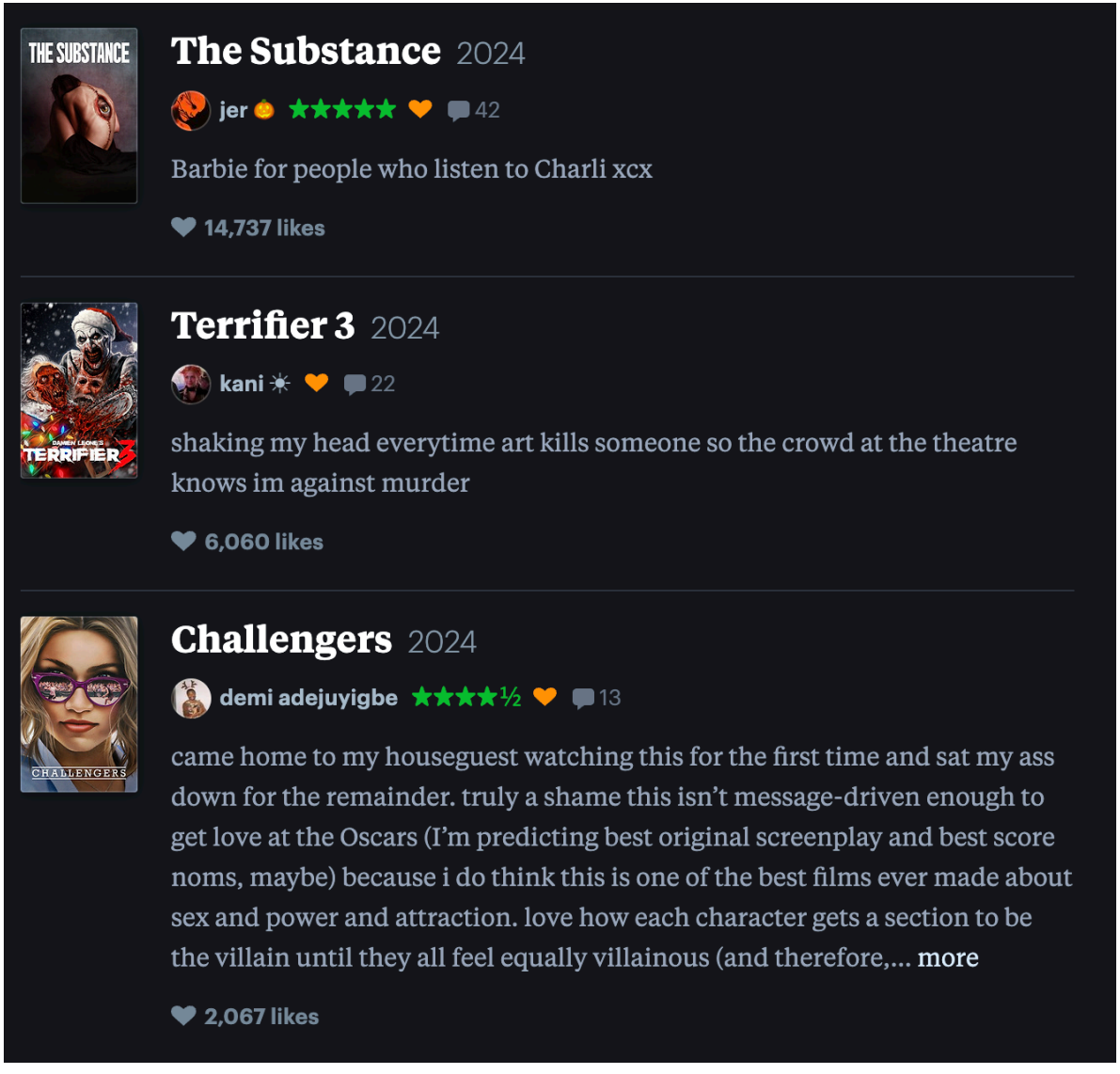
Like Beli, but for movies!

Founded in 2011



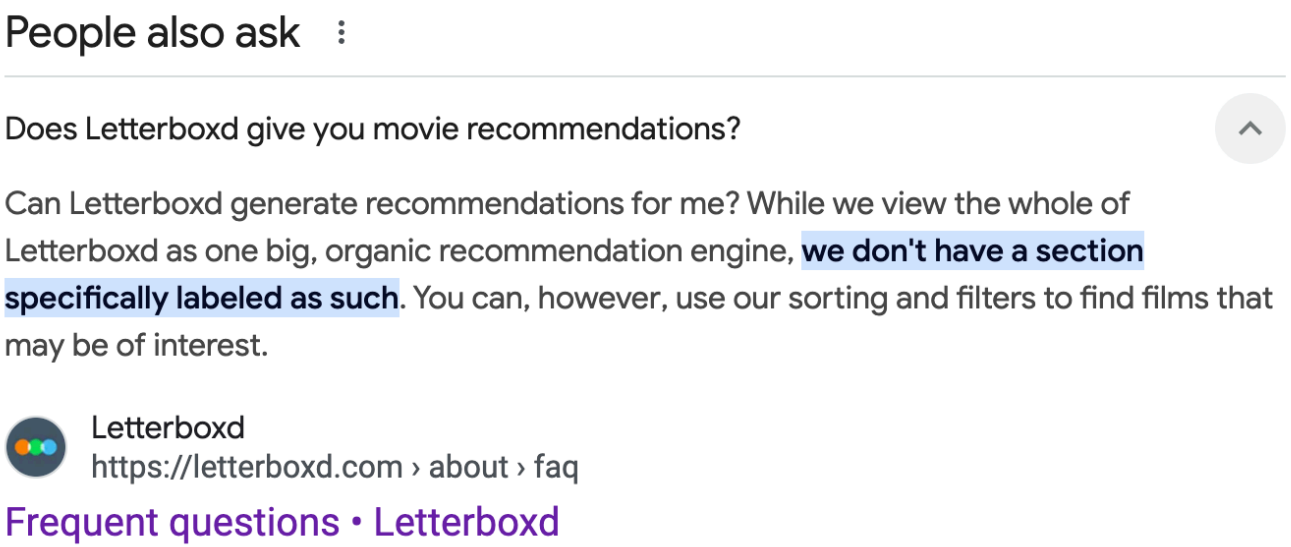
letterboxd.com

How does it work?



What’s the problem?

No recommendation system :(

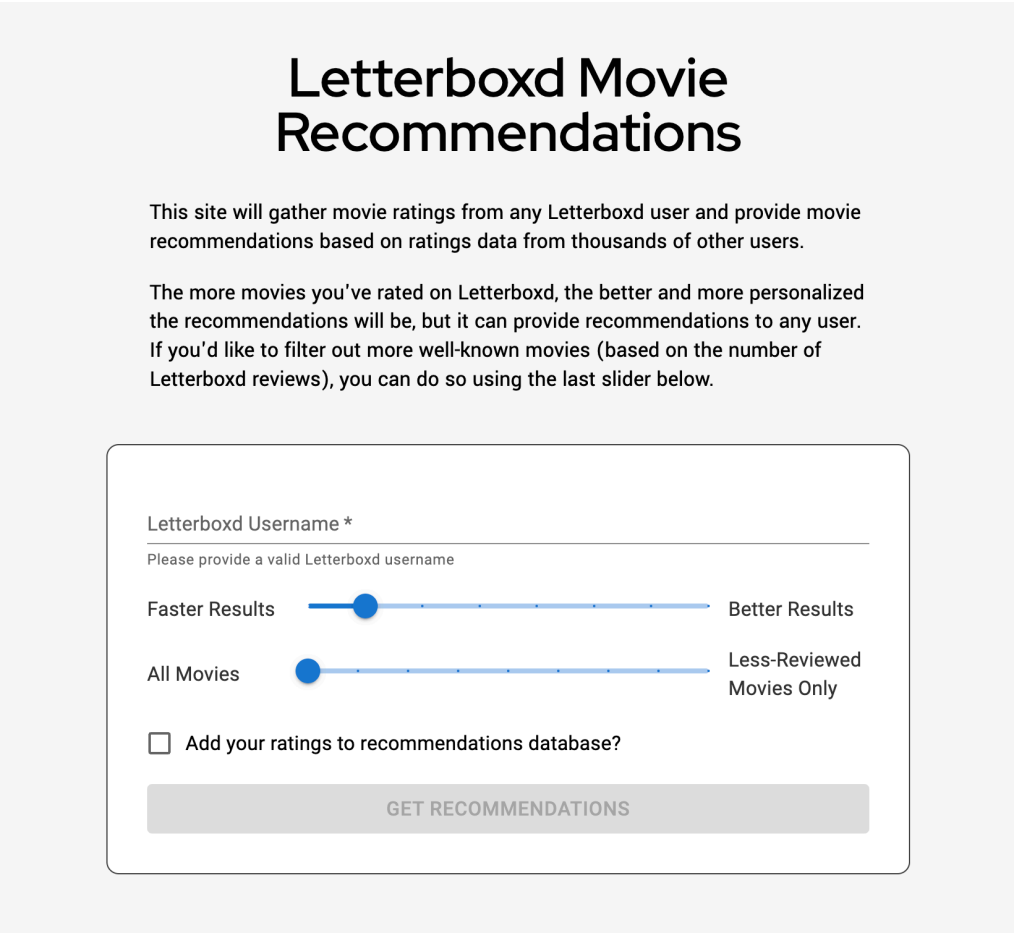


Sam Lerner's Solution

Created in December, 2020

Data was being automatically updated from Jan 2023-Jun 2024

Hasn't pushed anything since a Google tag in September 2024



Source: Sam Lerner's Letterboxd Recommendation Website

Sam Learner's solution

1. 

Carol (2015)
Romance Drama
10.00

2. 

Hamilton (2020)
History Drama
10.00

3. 

Metropolis (1927)
Drama Science Fiction
10.00

4. 

2001: A Space Odyssey (1968)
Science Fiction Mystery Adventure
10.00

How does it work?

Recommendations generated for Geraldine

Not great recommendations, mainly popular movies

Source: Sam Lerner's GitHub

- Problem: Predict user preferences
- Collaborative filtering is a popular approach for building recommendation systems.
- It relies on the behavior and preferences of other users to make predictions.
- Two main types:
  - **User-based collaborative filtering**: Recommends items based on similar users.
  - **Item-based collaborative filtering**: Recommends items based on the similarity between items.

Source: (Colace et al., 2022a)

The User-Item Matrix

- Represents the relationship between users and items
- Users $U = \{u_1, u_2, \ldots, u_m\}$ and Items $I = \{i_1, i_2, \ldots, i_n\}$.
- The **user-item matrix** R contains user preferences:

$$R_{ij} = \begin{cases} \text{rating given by user } u_i \text{ to item } i_j, & \text{if rated,} \\ \text{null,} & \text{otherwise.} \end{cases}$$

- Matrix R is usually sparse and quite large (most entries are missing)

Source: (Colace et al., 2022a)

Simon Funk and SVD in the Netflix Prize

- **Context**:
  - Netflix launched a competition in 2006 to improve its recommendation engine by 10%.
  - The challenge involved predicting ratings in a large, sparse user-movie matrix.
- **Funk's SVD Approach**:
  - Simon Funk introduced an SVD method using stochastic gradient descent.
- **Impact and Popularization**:
  - Funk's approach quickly outperformed Netflix's existing system.
  - His open sharing of the method popularized SVD in collaborative filtering. [Funk's blog from 2006]
- **Legacy**:
  - Inspired further research and development in using SVD for recommendation systems.

Source: (Murray, 2022)

Using SVD for Matrix Factorization

- SVD: Singular Value Decomposition
- SVD decomposes the user-item matrix R:

$$R \approx U\Sigma V^T$$

- U: User-feature matrix (latent factors for users).
- V: Item-feature matrix (latent factors for items).
- Σ: Diagonal matrix with singular values (importance of each latent feature).

Source: (Colace et al., 2022b)

Latent Factor Model

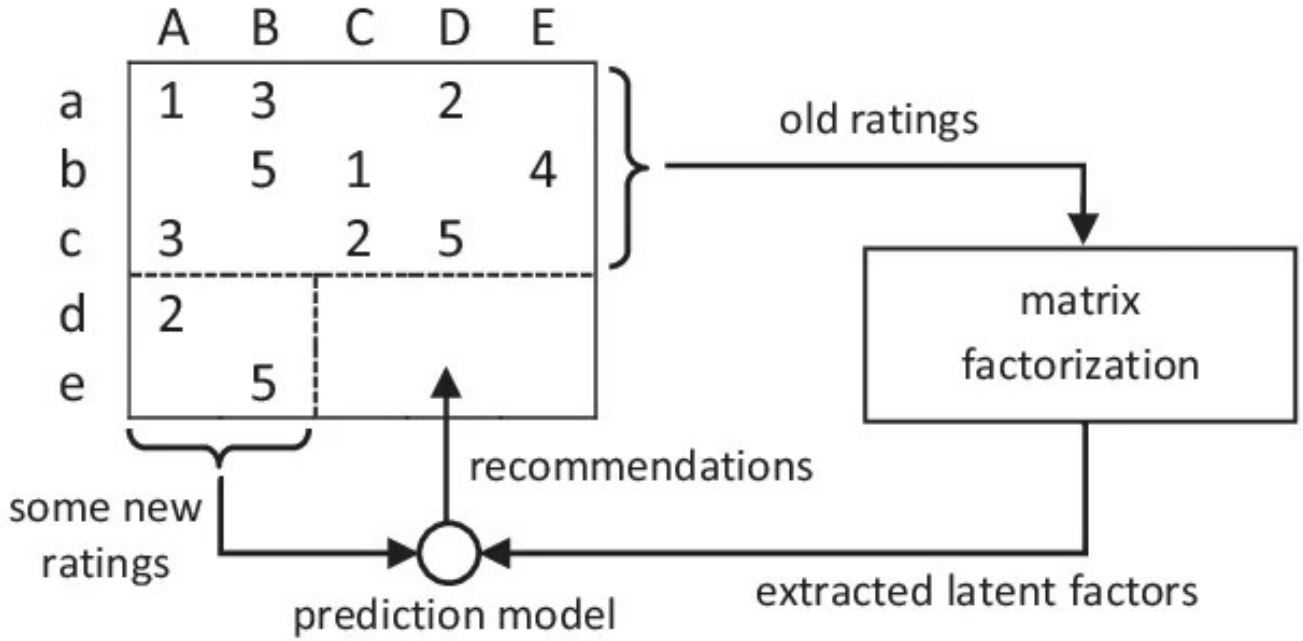
- Latent factors capture the underlying characteristics:
  - For movies: genre, director, or popularity.
  - For users: preferences for specific genres.
- Each user and item are represented by a vector in the latent space.
- Prediction: The user u_i rating for item j is:

$$\hat{R}_{ij} = \sum_{k=1}^{K} U_{ik} \Sigma_k V_{jk}^T$$

Sources: (Colace et al., 2022b), (Pu & Faltings, 2013)

Matrix Factorization Visualized

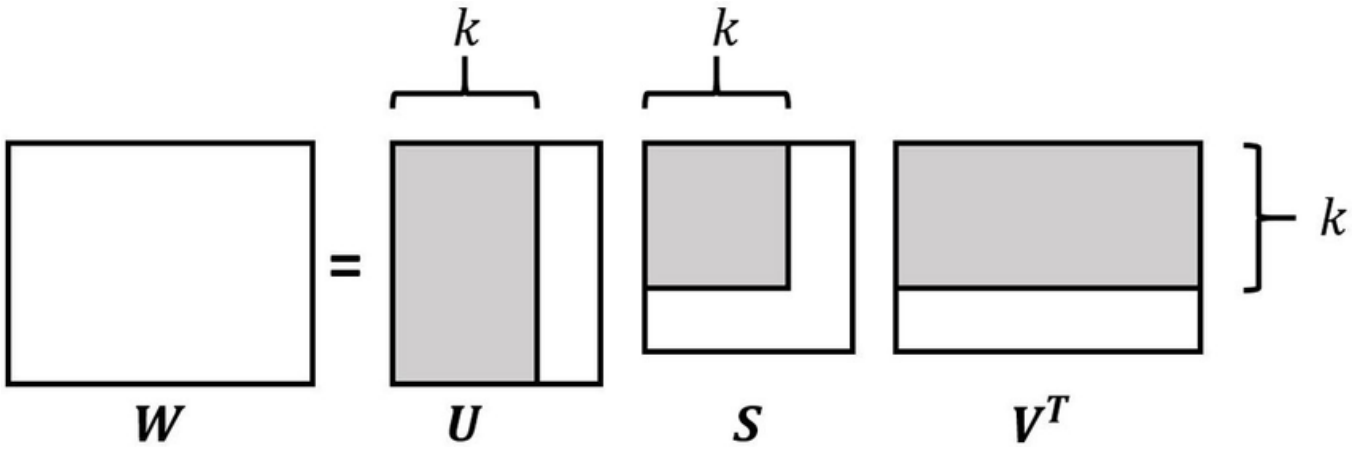
- ”Old ratings” are fed into matrix factorization.
- Matrix factorization extracts latent factors, which represent hidden user and item characteristics.
- A prediction model combines these latent factors with any new ratings to make personalized recommendations.



Source: (Pu & Faltings, 2013)

Truncated SVD for Recommendation Systems

- In practice, we use **truncated SVD** to retain only the top k singular values.
- Reduces complexity, dimensionality, and computational cost.
- Keeps only the most important latent factors without losing too much information.
- Simpler models tend to generalize better and are less likely to overfit, making more accurate predictions on unseen data.



Source: (Pinoli et al., 2015)

Benefits of SVD for Recommendations

- Captures complex relationships between users and items.
- Works well for collaborative filtering—recommending items based on patterns across users.
  - Finds patterns across users to recommend items they are likely to enjoy.
  - Makes accurate predictions by leveraging these latent patterns.
- Truncated SVD reduces dimensionality, making it easier to handle large datasets with many users and items.
  - Focuses on the most meaningful factors, improving generalization.
  - Helps lower generalization error, leading to better performance on unseen data.

Sources: (Colace et al., 2022a), (Pinoli et al., 2015), (Pu & Faltings, 2013)

Limitations of SVD for Recommendation Systems

- Often requires retraining when new data is added.
- Struggles with very sparse datasets and cold start problem.
- Assumes linear interactions between latent factors.
- Typically uses only explicit feedback.

Sources: (Colace et al., 2022a), (Pu & Faltings, 2013)

Summary of SVD for Recommendations

- SVD factorizes the user-item matrix into latent user and item vectors, capturing hidden patterns.
- Latent factor model is used to predict missing ratings by identifying relationships between users and items, helping the model generalize
- SVD captures complex user-item relationships but has limitations, including struggles with sparse data and the cold start problem.

Non-Negative Matrix Factorization (NMF)

Non-Negative Matrix Factorization (NMF)

- **Definition:** NMF is a matrix factorization technique that decomposes a non-negative matrix R into two non-negative matrices W and H such that:

$$R \approx W \times H$$

- **Purpose:** Extracts latent features or patterns while preserving non-negativity, making results more interpretable in areas like topic modeling, image processing, and recommendation systems.

Source: (Lee & Seung, 2001)

NMF Problem Formulation

- **Objective:** Minimize the reconstruction error:

$$\min_{W,H} \|R - WH\|_F^2$$

- **Constraints:** Elements of W and H must be non-negative.
- **Interpretation:**
  - W: Latent feature representation for rows of R.
  - H: Latent feature representation for columns of R.

Source: (Lee & Seung, 2001)

Frobenius Norm in Matrix Factorization

Definition: The Frobenius norm of the matrix $R - WH$ is given by:

$$\|R - WH\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |r_{ij} - (w_i^T h_j)|^2}$$

Where:

- $R \in \mathbb{R}^{m \times n}$ is the original matrix (e.g., user-item interactions).
- $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ are the factorized matrices.
- $w_i^T h_j$ is the dot product between the i-th row of W and the j-th column of H.

Key Properties:

- Measures the error or approximation between the original matrix R and the approximation WH.
- It is equivalent to summing the squared differences between the entries of R and WH.

Source: (Burred, 2014)

Comparison to Singular Value Decomposition (SVD)

- **SVD:** Factorizes a matrix R into $U\Sigma V^T$, where:
  - U and V contain both positive and negative entries.
  - Σ contains singular values representing the strength of each latent dimension.
- NMF also uses matrix factorization on explicit feedback
- **Key Differences:**
  - **Non-negativity:** NMF ensures that W and H are non-negative, unlike SVD.
  - **Interpretability:** NMF can yield more interpretable features since it preserves natural constraints.

Source: (Lee & Seung, 2001)

Solving NMF using Alternating Least Squares (ALS)

- **Basic Idea:** Fix one matrix (e.g., W) and solve for the other (e.g., H) alternately.
- **Optimization:** Minimizes the objective function by iteratively updating W and H.
- **Benefits of ALS:** Simplifies optimization by breaking it into convex subproblems.

Source: (Kim & Park, 2008)

Pseudocode for NMF with Alternating Least Squares (ALS)

Algorithm 1 NMF using ALS

Initialize W and H with non-negative values
for iteration in 1 to max_iter **do**
    Fix H, update W by solving non-negative least squares
    for each row i of R **do**
        $W[i,:] \leftarrow \arg\min_{W[i,:]} ||H^T W[i,:] - R[i,:]||_2^2$
    end for
    Fix W, update H by solving non-negative least squares
    for each column j of R **do**
        $H[:,j] \leftarrow \arg\min_{H[:,j]} ||WH[:,j] - R[:,j]||_2^2$
    end for
    Compute reconstruction error; if below tolerance, break
end for
return W, H

Summary

- **NMF:** Powerful technique for non-negative data factorization, often yielding interpretable results.
- **Comparison to SVD:** NMF's non-negativity constraint aligns well with data like text, images and **reviews!**
- **ALS for NMF:** Alternates between W and H updates, providing a simple and effective solution.

NMF Demo

(Code Demo)

Bayesian Personalized Ranking

Introduction to Bayesian Personalized Ranking (BPR)

- **BPR** is a ranking approach designed for recommendation systems with **implicit feedback**.
- Focuses on **pairwise ranking** of items based on observed interactions.
- Goal: Rank items a user has interacted with higher than items they haven't.

Source: (Rendle et. al, 2009)

Problem Setup

- Given a user u and items i and j:
  - i: an item the user has interacted with (positive).
  - j: an item the user has not interacted with (negative).
- Objective: Maximize the probability that the model ranks item i higher than j.

Source: (Rendle et. al, 2009)

Score Calculation

- We are looking to decompose the user-item matrix into two matrices.

$$R \approx P \times Q^T$$

- Each user u and item i have associated **latent vectors**:

$$P_u \quad \text{and} \quad Q_i$$

- Predicted score for user u on item i:

$$\hat{x}_{ui} = P_u \cdot Q_i$$

- Goal: For positive item i and negative item j,

$$\hat{x}_{ui} > \hat{x}_{uj}$$

Source: (Rendle et. al, 2009)

Pairwise Preference Probability

- Define a **probabilistic model** for preference:

$$P(u, i, j) = \sigma(\hat{x}_{ui} - \hat{x}_{uj}) = \sigma(\hat{x}_{uij})$$

- Here, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the **sigmoid function**.
- This models the likelihood that u prefers i over j.

Source: (Rendle et. al, 2009)

Log-Likelihood Objective

- Maximize the **log-likelihood** of observed preferences:

$$\log P(u, i, j) = \log \sigma(\hat{x}_{uij})$$

- Add a **regularization term** to prevent overfitting:

$$\log \sigma(\hat{x}_{uij}) - \lambda(\|P_u\|^2 + \|Q_i\|^2 + \|Q_j\|^2)$$

- Here, λ controls the regularization strength.

Source: (Rendle et. al, 2009)

Optimization with Gradient Descent

- Use **gradient descent** to update parameters:
  - User vector P_u
  - Item vectors Q_i and Q_j
- Adjust vectors to maximize the likelihood of positive item scores over negative item scores.

Prediction Formula:

$$\hat{x}_{ui} = P_u \cdot Q_i = \sum_{f=1}^{k} P_{uf} \cdot Q_{if}$$

Derivatives of Matrix Factorization:

$$\frac{\partial}{\partial \theta} \hat{x}_{uij} = \begin{cases} (Q_{if} - Q_{jf}) & \text{if } \theta = P_{uf} \\ P_{uf} & \text{if } \theta = Q_{if} \\ -P_{uf} & \text{if } \theta = Q_{jf} \end{cases}$$

Source: (Rendle et. al, 2009)

Pseudocode for BPR (Part 1: Initialization)

Algorithm 2 CREATEBPRTriplets(R)

Input: User-item interaction matrix R
Output: Set of training triplets D_R
$D_R \leftarrow \emptyset$ {Initialize the set of training triplets}
for each user $u \in U$ **do**
  for each item $i \in I$ such that $(u, i) \in R$ **do**
    for each item $j \in I$ such that $(u, j) \notin R$ **do**
      $D_R \leftarrow D_R \cup \{(u, i, j)\}$ {Add triplet to D_R}
    end for
  end for
end for
return D_R

Source: (Rendle et. al, 2009)

Pseudocode for BPR (Part 2: Training)

Algorithm 3 LEARNBPR(D_R, Θ)

Randomly initialize P and Q
repeat
  Draw (u, i, j) from D_R
  $P_u \leftarrow P_u + \eta \left((1 - \sigma(\hat{x}_{uij}))(Q_i - Q_j) - \lambda P_u \right)$
  $Q_i \leftarrow Q_i + \eta \left((1 - \sigma(\hat{x}_{uij}))P_u - \lambda Q_i \right)$
  $Q_j \leftarrow Q_j + \eta \left(-(1 - \sigma(\hat{x}_{uij}))P_u - \lambda Q_j \right)$
until convergence
return P, Q

Source: (Rendle et. al, 2009)

Model Performance Comparison

Precision@k

Definition: Precision@k measures the fraction of relevant items in the top-k recommendations.

Formula:

$$\text{Precision@k} = \frac{\text{Number of relevant items in top-}k}{k}.$$

Example: For recommendations {A, B, C} with relevant items {C, D}:

- Top-k: {A, B, C}, $k = 3$.
- Precision@3 = $\frac{1}{3}$.

Reciprocal Rank (RR)

Definition: RR is reciprocal rank of the first relevant item in a ranked list.

Formula:

$$\text{RR} = \frac{1}{\text{rank}_i},$$

where rank_i is the position of the first relevant item in user i's recommendation list.

Example: For recommendations {A, B, C} with relevant item {C}:

- Rank of C is 3.
- RR = $\frac{1}{3}$.

Source: (Voorhees, 2007)

Normalized Discounted Cumulative Gain (NDCG)

Definition: NDCG measures the quality of ranking by considering the positions of relevant items, assigning higher weights to items ranked higher.

Formula:

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}},$$

where

$$\text{DCG} = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)},$$

and IDCG is the DCG of the ideal ranking.

Example: Recommendations: {A, B, C}, Relevant items: {A: 3, C: 2, D: 1}.

- DCG = $\frac{3}{\log_2(1+1)} + \frac{0}{\log_2(2+1)} + \frac{2}{\log_2(3+1)} = 3.5.$
- IDCG = $\frac{3}{\log_2(1+1)} + \frac{2}{\log_2(2+1)} + \frac{1}{\log_2(3+1)} = 4.76.$
- NDCG = $\frac{3.5}{4.76} \approx 0.735.$

Source: (Wang et. al, 2013)

General and user-level performance comparison

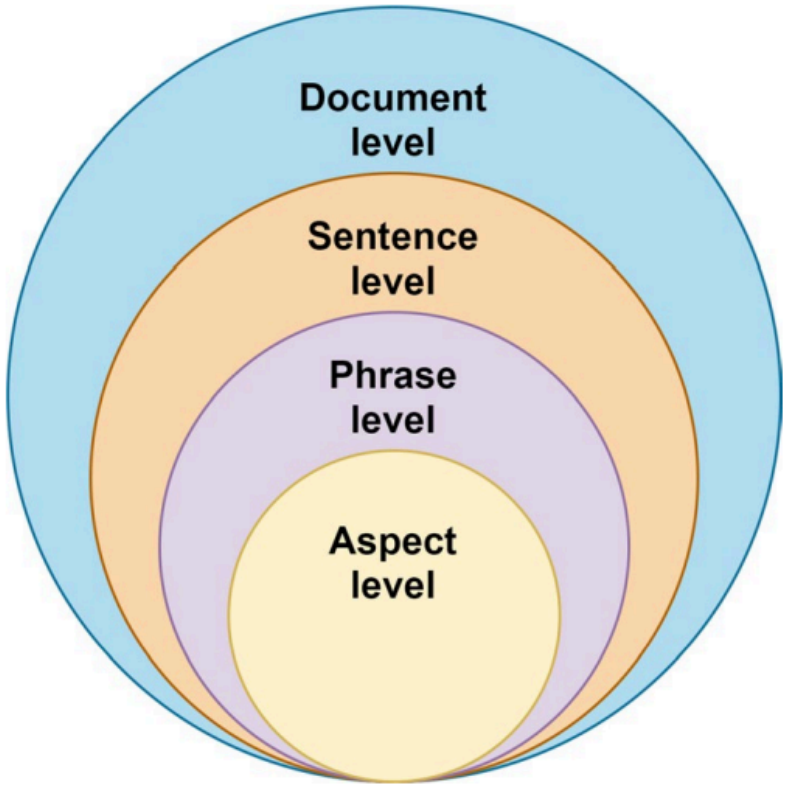
Code demo

Review Sentiment Analysis

Introduction to Sentiment Analysis

- Extracts subjective information from text using NLP and text mining
- Common applications
  - Analyze customer feedback
  - Identify/track market trends
  - Monitor brand reputation

Levels of Sentiment Analysis



Source: A survey on sentiment analysis methods, applications, and challenges

Levels of Sentiment Analysis

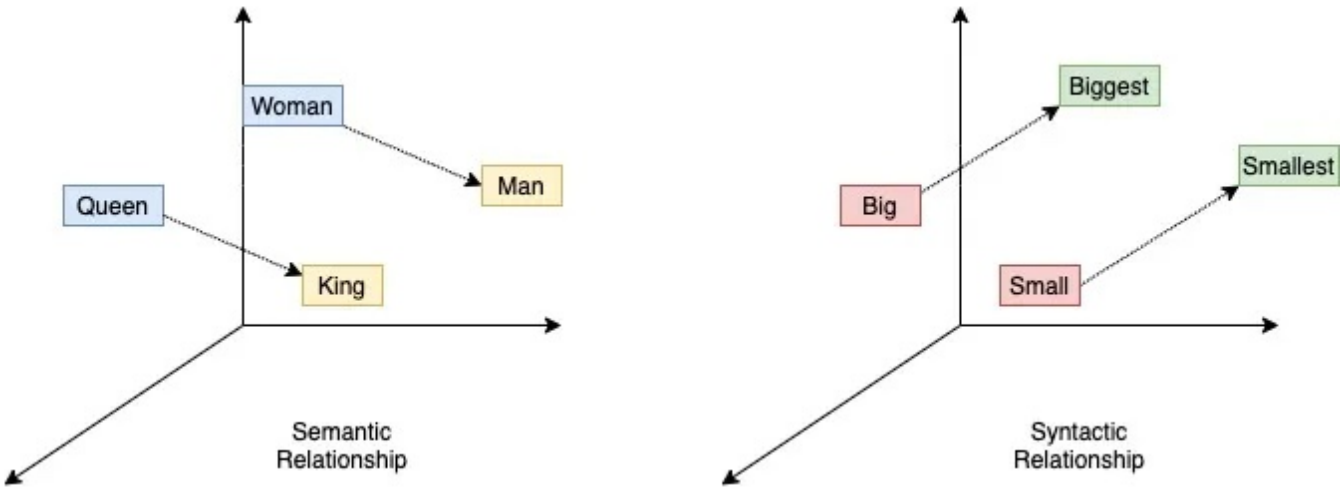
- Document level
  - For example, classify chapters of a book
  - Not as common
- Sentence level
  - Useful when a document has a wide range of sentiments
  - Subjective classification
  - Classified sentiments can be used individually or aggregated to classify the sentiment of the entire document
- Phrase level
  - Useful for product reviews that contain multiple lines
- Aspect level
  - Sentiment classification of individual words
  - Classify sentiment of each aspect and aggregate to classify sentiment of entire sentence

Feature Extraction

- Extract most useful information that represents the text's most important features
- Terms frequency: number of times a word has appeared
- Parts of speech tagging: Classify token as its part of speech (i.e. noun, verb, adjective, etc.)
- Negation: words that can reverse polarity (e.g. not, never, none, etc.)
- Bag of words: combine vocabulary of words into a "bag"
    - Problem: doesn't consider syntactical meaning

Word Embedding

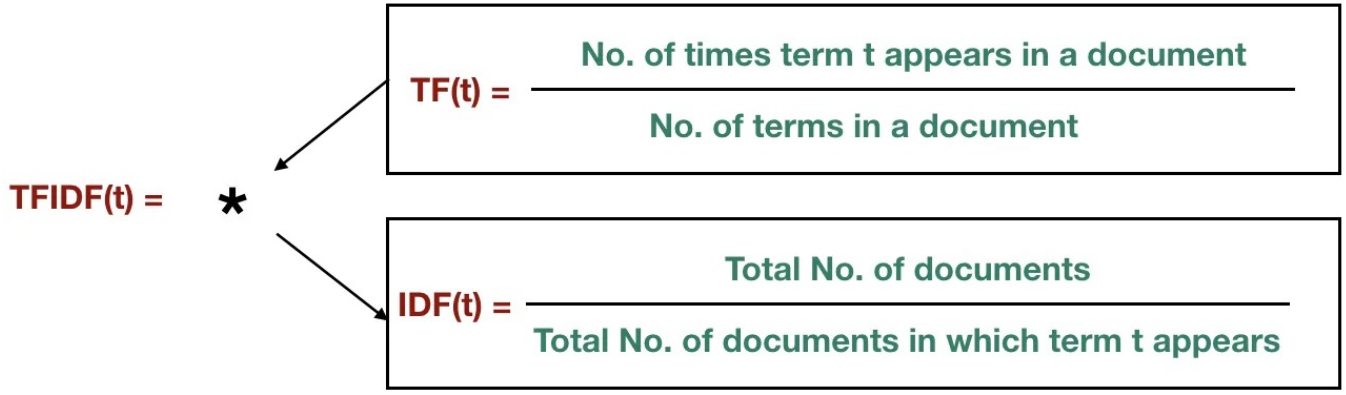
- Represent words in a vector space by grouping words with similar meanings
- Distance and direction between vectors are determined by the similarity and relationships between words



Source: Word2Vec Research Paper Explained

TF-IDF

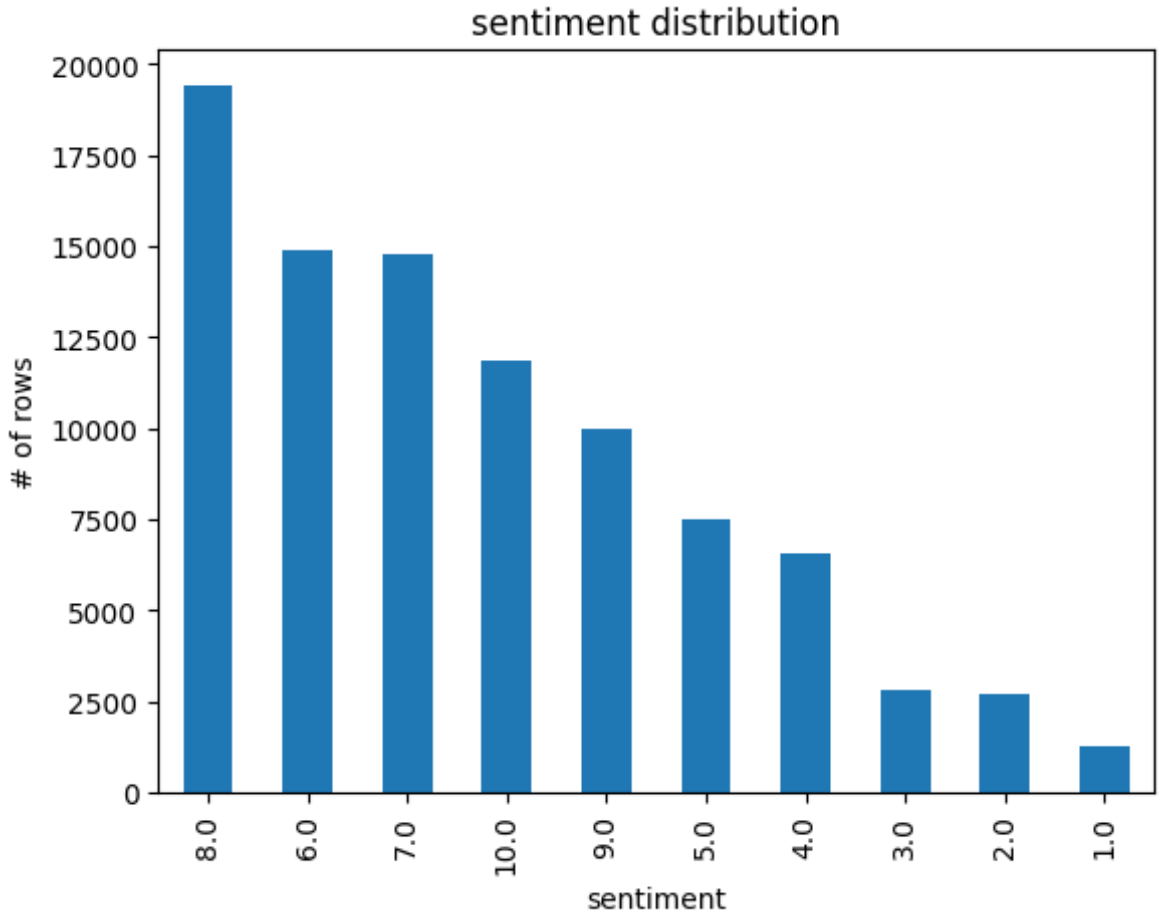
Frequency-based embedding that assigns a weight to each word that represents the significance of the word (first introduced in 1972)



Source: Sentiment Classification for Restaurant Reviews using TF-IDF

Outline of Sentiment Analysis - Balance Classes

Problem: imbalanced classes



Outline of Sentiment Analysis - Tokenization

- Split text into an array of words (tokens)
- Allows for each word to be transformed into a number

	tokens
4	[if, the, real, pompeii, was, this, boring, gl...
16	[eli, roth, needs, to, stick, to, horror, even...
50	[should, ve, stayed, at, hilton, hotel]
100	[what, pretentious, pile, of, horse, dung]
141	[would, like, to, formally, apologize, to, the...

Outline of Sentiment Analysis - Stemming

- Remove prefixes and suffixes to get the root word of a word
- Choose between PorterStemmer, LancasterStemmer, and SnowballStemmer based on accuracy of results
- We choose SnowballStemmer (developed 2013-2019)

	porter_stemmed
4	[if, the, real, pompeii, wa, thi, bore, glad, ...
16	[eli, roth, need, to, stick, to, horror, even,...
50	[should, ve, stai, at, hilton, hotel]
100	[what, pretenti, pile, of, hors, dung]
141	[would, like, to, formal, apolog, to, the, nat...

	lancaster_stemmed
4	[if, the, real, pompei, was, thi, bor, glad, t...
16	[el, roth, nee, to, stick, to, hor, ev, though...
50	[should, ve, stay, at, hilton, hotel]
100	[what, pretenty, pil, of, hors, dung]
141	[would, lik, to, form, apolog, to, the, nat, o...

	snowball_stemmed
4	[if, the, real, pompeii, was, this, bore, glad...
16	[eli, roth, need, to, stick, to, horror, even,...
50	[should, ve, stay, at, hilton, hotel]
100	[what, pretenti, pile, of, hors, dung]
141	[would, like, to, formal, apolog, to, the, nat...

Outline of Sentiment Analysis - TF-IDF

- Build dictionary from stemmed tokens
- Build corpus from converting stemmed tokens of each review into a bag of words
- Build TF-IDF model from corpus
- Generate TF-IDF vectors for each review
- Use DecisionTreeClassifier to perform sentiment classification

```
sample data from dictionary:
word: bore - id: 0
word: glad - id: 1
word: got - id: 2
word: if - id: 3
```

Combined Recommendation Method

Given m users and n items in the training rating matrix $R_{m \times n}$ ($r_{ij} \in R_{m \times n}$: rating of user u_i on item i_j), the predicted rating of user u_a on item i_j in the test set is given by:

$$pr_{aj} = \beta * pr\_mf_{aj} + (1 - \beta) * pr\_sent_{aj}$$

- pr_mf_{aj}: predicted rating by matrix factorization methods for user u_a on item i_j
- pr_sent_{aj}: predicted rating by sentiment model for user u_a on item i_j
- β adjusts the weight/importance of each term

Rating Prediction from Sentiment Model Pseudocode

Algorithm 4 Compute pr_sent_{aj}

```
1:  Initialize list of items I to an empty list
2:  for item i_k in training set do
3:      if user u_a has rated item i_k then
4:          Add i_k to I
5:      end if
6:  end for
7:  Initialize list of users U to an empty list
8:  for user u_b in training set do
9:      for item i_k in I do
10:         if user u_b has rated item i_j and user u_b has rated item i_k then
11:             Add user u_b to U
12:         end if
13:     end for
14: end for
15: Initialize list of similarities S to an empty list
16: if U is not empty then
17:     for user u_i in U do
18:         s_ai ←cosine_similarity(user u_a, user u_i) {using sentiment model}
19:         Add s_ai to S
20:     end for
21:     Calculate pr_sent_aj
22: end if
```

Relevant Equations in Sentiment Model Pseudocode

$$s_{ai} = \frac{\sum_{j=1}^{n} r_{aj} r_{ij}}{\sqrt{\sum_{j=1}^{n} r_{aj}^2} \sqrt{\sum_{j=1}^{n} r_{ij}^2}} \text{ (cosine similarity)}$$

$$pr\_sent_{aj} = \bar{r_a} + \frac{\sum_{i=1}^{k} s_{ai}(r_{ij} - \bar{r_i})}{\sum_{i=1}^{k} |s_{ai}|}$$

- r_{aj}: rating of user u_a on item i_j
- r_{ij}: rating of user u_i on item i_j
- $\bar{r_a}$: average rating of user u_a
- $\bar{r_i}$: average rating of user u_i
- s_{ai}: similarity between user u_a and user u_i

Sentiment Analysis Results

Code demo

Future work

Future of the field: Netflix

- Considers similar factors like viewing history, ratings, others with similar preferences, and information about movies such as genre, cast, year released
- Takes into account time of day, user's preferred language, time constraints, and the device being used
- Personalizes the layout of movies, such as the row they appear in and the order of the titles, and user searches

Source: Netflix

Future of the field: Amazon

- Considers item-to-item similarity instead of user-to-user similarity.
- Uses multi-layered neural networks for Prime Video recommendations. Considers watch history, time of day, day of week and even weather.

Source: Amazon

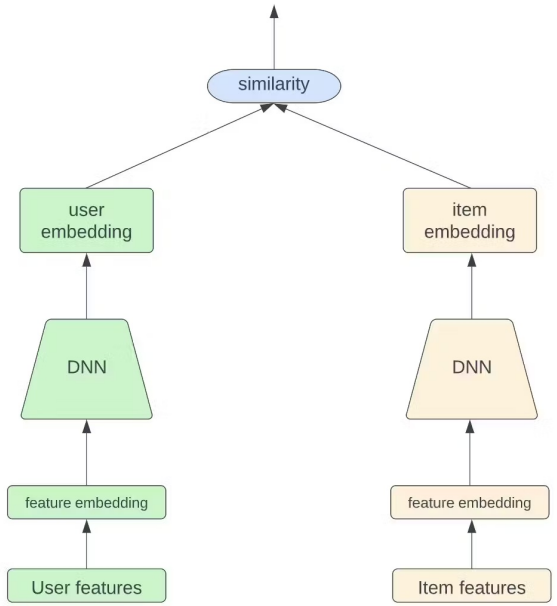
Future of the field: Tiktok

- Considers three main factors: user interactions, content information, user information
- Predictions also influenced by users with similar interests
- Example: For You
  - User interactions: likes, shares, comments, length of time spent watching a video
  - Content information: sounds, hashtags, number of views, country in which video was published
  - User information: device settings, language preference, location, time zone, day, device type

Source: TikTok

Future of the Field, More Generally

- Generative recommenders (2024): predict sequences of user behavior, well-suited for large amounts of frequently-changing data
- Generative retrieval (2023): predicts items a user is likely to interact with in the future, well-suited for handling new items
- Two-tower model:



Source: Understanding the Two-Tower Model in Personalized Recommendation Systems

Our futures

Laya Gollamudi
SEAS
Management
Consulting

Sierra Martinez-Kratz
CC
Data Science
Grad School

Geraldine Nina
SEAS
Risk Analysis

Questions?