Basics first….

<https://buomsoo-kim.github.io/recommender%20systems/2020/07/14/Recommender-systems-collab-filtering-2.md/>

1. **Collaborative filtering (CF)**

CF systems suggest items that similar users liked, i.e., “people-to-people” correlation. In general, there are two steps in CF - (1) identifying users with similar likings in the past and (2) recommending items that such users prefer.

1. **Memory-based**

The memory-based (aka heuristic-based or neighborhood) approach utilizes pre-computed user-item rating records, i.e., “memory,” to infer ratings for other items that the user has not encountered yet

1. user-based and

First finds similar users to a user of interest, i.e., *neighbours.* Then, the rating for a new item is inferred based on rating patterns of the neighbours

1. item-based

The rating for an item is predicted with ratings of the user for items that are similar to the item of interest.

1. **Model-based CF systems**

Model-based CF systems use various predictive models to estimate the ratings for a certain user-item pair.

1. Latent factor models

that attempts to characterize both items and users with a finite number of latent factors

1. Matrix factorization

is one of the most established methods for such approach.

<https://buomsoo-kim.github.io/recommender%20systems/2020/09/25/Recommender-systems-collab-filtering-12.md/>

1. **Content based (CB)**

Content-based systems recommend items that are close to the items that the user liked before.

Defining such similarity function might be tricky and burdensome since many items do not have explicit features that can be easily quantified.

1. **Hybrid**

**Surprisen tarjoamat suosittelumetodit ja niiden sopivuus eri tarkoituksiin:**

(chat gpt) The Surprise library in Python provides a robust framework for implementing recommendation systems. The choice of a method depends on the nature of your data and the recommendation problem you're addressing. Here's an overview of the main methods available and recommendations for their use.

1. **Collaborative Filtering (CF)**
2. **User-based Collaborative Filtering**
3. **Item-based Collaborative Filtering**
4. **Matrix Factorization**
5. **SVD (Singular Value Decomposition)**
6. **SVD++**
7. **KNN-based Approaches**
8. **Baseline Algorithms**
9. **KNNBaseline**

**Why and When to Use a Method:**

* Small Datasets: KNN-based methods or user-based CF can be effective because they are interpretable and require fewer computational resources.
* Sparse Datasets (paljon puuttuvia tietoja) : SVD or SVD++ are generally preferred due to their ability to handle sparsity and capture latent factors.
* Implicit Feedback: SVD++ is better equipped to handle additional information from implicit interactions.
* Cold Start Problems:

For new users: Use user-based CF or demographic-based methods.

For new items: Use content-based filtering (not directly supported by Surprise).

**Recommendation**

If you have no clear preference and a moderately large dataset:

Start with SVD, as it provides a balance between simplicity, accuracy, and scalability.

Use cross-validation and hyperparameter tuning (via GridSearchCV in Surprise) to optimize your model for your dataset.

Description of each method:

**1. Collaborative Filtering (CF)**

**(a) User-based Collaborative Filtering**

* Method: Similarity between users is computed based on their ratings of items.
* When to use:

When you have many users compared to items.

When you expect users to have overlapping preferences.

**(b) Item-based Collaborative Filtering**

* Method: Similarity between items is computed based on ratings they received from users.
* When to use:

When you have fewer items compared to users.

When items are likely to have strong interrelations (e.g., books by the same author or movies in the same genre).

esimerkki : <https://www.kaggle.com/code/laowingkin/netflix-movie-recommendation> (Netflix data, mutta ei Netflixin käyttämä)

**2. Matrix Factorization**

<https://surprise.readthedocs.io/en/stable/matrix_factorization.html>

**(a) SVD (Singular Value Decomposition)**

<https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD>

* Method: Decomposes the user-item matrix into lower-dimensional matrices.
* Why it's popular:

Effective for capturing latent features in the data.

Used in Netflix Prize-winning solutions.

* When to use:

When you have sparse data with many missing values.

When you want to predict missing ratings (not just recommend the top-N items).

esimerkki (Netflix): <https://github.com/recommenders-team/recommenders/blob/main/examples/02_model_collaborative_filtering/surprise_svd_deep_dive.ipynb>

**(b) SVD++**

* Method: Extends SVD by incorporating implicit feedback (e.g., interaction data like clicks or views).
* When to use:

When implicit feedback plays an important role.

When you want a more robust model at the cost of higher computation time.

**3. KNN-based Approaches**

<https://surprise.readthedocs.io/en/stable/knn_inspired.html>

* Method: Uses nearest neighbours for recommendation based on similarity metrics like cosine similarity, Pearson correlation, etc.
* When to use:

When you need a simple, interpretable model.

When your dataset is relatively small.

* Limitations: May not scale well for large datasets.

esimerkki: <https://www.kaggle.com/code/mammadabbasli/recommendation-engine-w-knn> (tässä tehdään kuitenkin matrix factorization ennen KNN:n käyttöä)

esimerkki: <https://www.kaggle.com/code/sirpunch/build-knn-and-apriori-based-recommendation-engines>

**4. Baseline Algorithms**

<https://surprise.readthedocs.io/en/stable/basic_algorithms.html>

* Method: Focus on simple models like global averages, user biases, and item biases.
* When to use:

As a baseline for comparison with more complex methods.

When simplicity and interpretability are crucial.

**5. KNNBaseline**

<https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBaseline>

* Method: Combines collaborative filtering with a baseline model for bias correction.
* When to use:

When you need better accuracy than basic KNN methods.

For datasets where biases (e.g., some users rating items consistently higher or lower) are significant.

Esimerkit Kerttulin ja SVOD google haun tarjoama