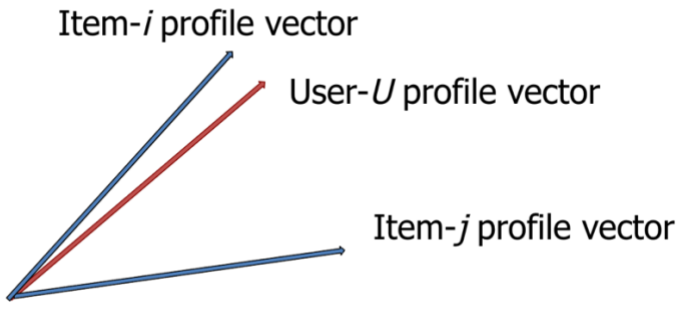
- Jaccard Similarity
    - The jaccard similarity of two sets is the size of their intersection divided by the size of their union
    - Jaccard similarity(0-1): Sim(c1,c2) = |c1 intersect c2|/|c1 union c2|
        - 0: complete different, 1: identical set
    - Jaccard distance(0-1): 1- jaccard similarity
        - 0: identical set, 1 complete different
    - Pro: simple algorithm for binary problem
    - Con: not good for data with rating.
- Cosine Similarity
    - Dot product of two vectors divide by Euclidean distance of two vectors x and y from origin

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$



- Minkowski distance:
    - Euclidean distance (when q = 2)

$$d(X, Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q},$$

- **Content-based approach** (TF*IDF) (feature based):
    - **Use characteristic/feature of item**
    - **Ways to approach:**
        - Item profile: 1 . for movie recommendation system, use Boolean vector to represent movie stars in movies. Then match movie with have similar movie stars using cosine distance. 2. Item profile based on textual content can use vector include set of words' TF*IDF
        - User profile: create description vectors for users. Then use cosine similarity to find matching item.
    - **Pro:**
        - No need for data on other users. No cold start for item or sparsity problems.
        - Able to recommend to users with unique taste
        - Able to recommend new and unpopular items. Even when they are new without any rated.
        - Able to provide explanations of recommended item by provide content features
    - **Con:**
        - Finding appropriate features is hard: Data type such as images, movies, music
        - Hard provide recommendation for new user.
        - Unable to recommends out of existing user profile. Explore other interest
- **Content-based Example**:
    - Web-page/books, movies, search input
    - **TF*IDF (Term frequency*inverse doc frequency): shows most significant word represent a document.**
    - **TF (0-1)= number of times term t in document / most occurrences of any term**
    - **IDF = LOG2(total number of docs/ number of docs that mentioned term t)**

- **Collaborative Filtering** (Pearson correlation)(AKA. Behavior correlation):
    - Based on user past behaviors/ use similar decision made by other users
    - **User/item Neighborhood based correlation filtering:**
        - Choose a subset of other users/items (neiborhood), use weighted sum predict target user behavior

- Con: User based Collborative filtering doesn't scale well to million users. Because user behavior change more than item.
- Neighborhood based example Steps:
  - Assign weights to all users with respect to similarity with the active user
  - Select k neighbor users that have the highest similarities with active user (neghborhood)
  - Pearson correlation: Computer a weighted combination of selected neighbors' ratings (**Normalization needed**)

- **Pearson correlation coefficient: (Cosine similarity with normalization)**

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$

- **To make a prediction for an active user *a* on an item i:**

Your own opinion

Your Neighbor *u* opinion

How much you like u

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|}$$

where $\bar{r}_a$ and $\bar{r}_u$ are the average ratings for the user a and user u on all other rated items, and $w_{a,u}$ is the weight between the user a and user u. The summations are over all the users $u \in U$ who have rated the item i. F

- **User-based CF: (User's Neighbors => recommend)**

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \qquad P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|}$$

- **Item-Based CF: (Item's Neighbors => recommend)**

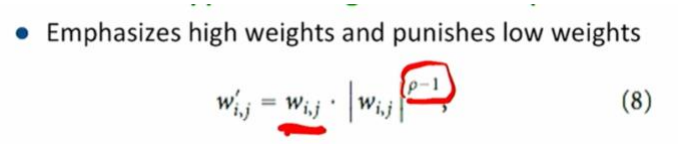
$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}} \qquad P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|}$$

41

- **Pearson correlation problem:**
  - When there are few neighbors, model become unreliable
  - Focuses on co-rated item and neglects global rating behavior. For example, A and B both rated on move x, y but A like movie Z a lot and B like movie H a lot.

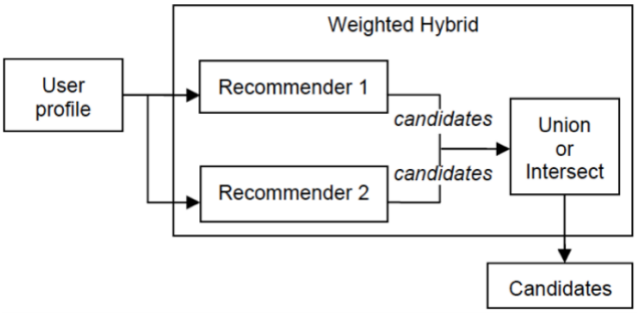
People like Z and H are actually two types of people. And Pearson correlation ignore this behavior.

- **Normalization:**
  - Deal better with non-rated items
  - Ensure correlation is not affected by rating habit
    - For example, one tends to rate on extreme scale. Another like to be conservative.
- **Collaborative Filtering feature engineering**:
  - **Pearson Correlation Case amplification**: p typical use 2.5

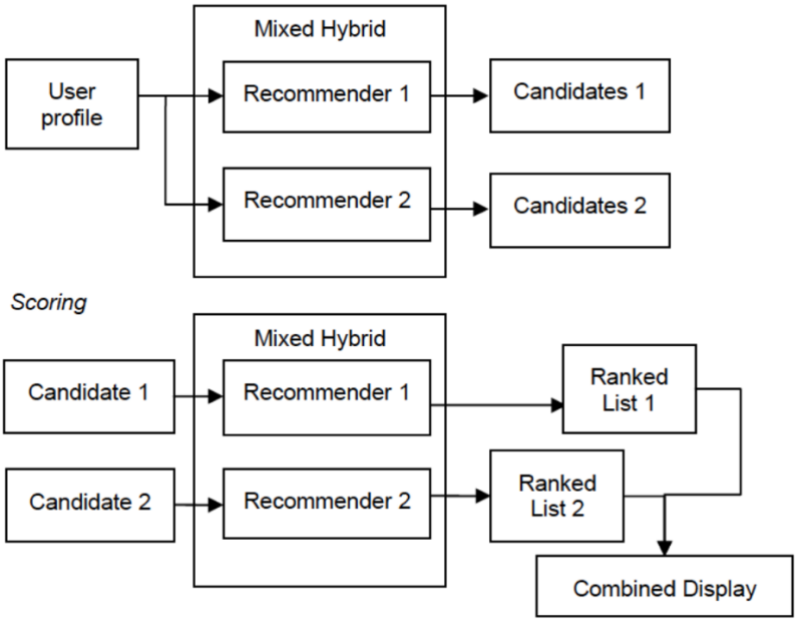


  - Missing value:
    - Mean imputation: fill missing value with mean
  - Default voting:
    - Herlocker et al. accounts for small intersection sets by reducing the weight of users that have fewer than 50 items in common
    - Chee et al. use average of the small group as default
    - Brrese et al. use neutral or somewhat negative preference for the unobserved ratings. Then computes similarity between users on the resulting rating data.
  - Inverse frequency: if all people buys an item, then reduce it's weight.
    - Ex: if all people buy milk, then it's useless information
  -
- **Model Based Collaborative Filtering**:
  - Bayesian models
  - Clustering model
  - Dependency networks
  - Classification algorithms
  - Regression and single value decomposition methods for numerical rating
- **Challenge of Collaborative Filtering:**
  - Data sparsity: content base hybrid solution.
  - Cold start problem: no rating established. Use content base.
  - Synonyms: different name for same product
  - Scalability: even O(n) is high, use dimension reduction.
  - Gray sheep and black sheep: Cannot conclude based on past behavior
  - Shilling attacks: seller rate on their own stuff to promote other buyer

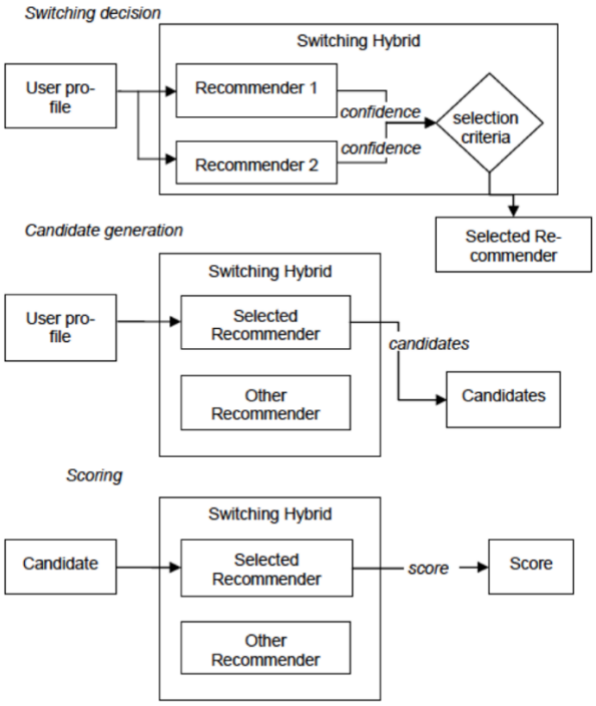
- **Hybrid recommendation system**:
  - **Weighted hybrid**:
    - Definition: Merges multiple recommendation results.
    - each candidate is rated by two or more recommenders. Weighted combination become item's predicted rating.



  - **Mixed hybrid**:

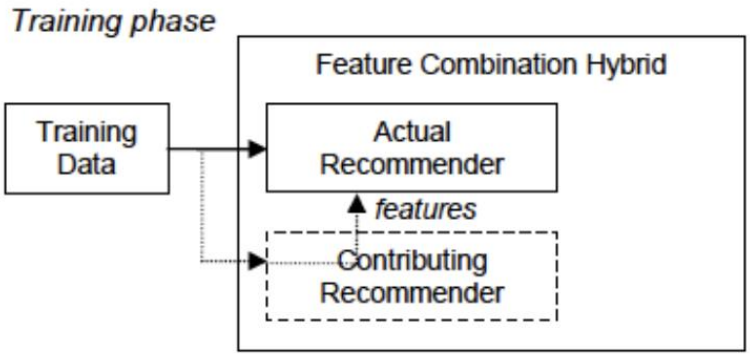


- Switching hybrid (choose one approach):
  - Definition: Choose one approach based on conditions
  - Assume WSJ has three recommendation components, content-based, collaborative. Candidate goes in order until getting high confidence. If new user, use content base, if active user, use collaborative filtering.

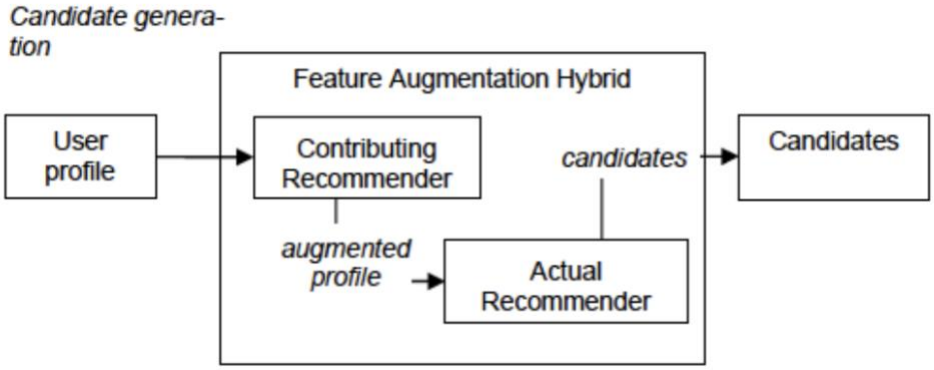


  -

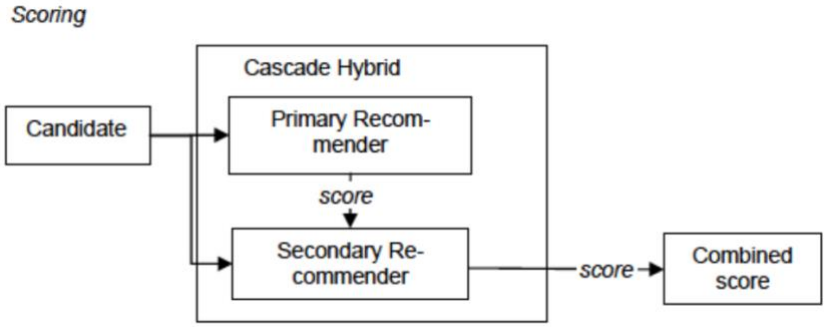
- Feature combination:
  - Definition: Merges features from multiple methods into **one model**.



- Feature augmentation:
  - Definition: Uses **output** from one model as **input features** for another model
  - Ex:
    - **Collaborative Filtering**: Use customer purchase history & ratings to train a CF model and generate **user & item embeddings**.
    - **Feature Augmentation**: Combine **CF embeddings** with **product metadata** (category, brand, price) as input for a content-based model.
    - **Final Content-Based Model**: Train a **machine learning model (e.g., XGBoost, Deep Learning)** using the augmented features to predict recommendations.



- Cascade hybrid:
  - Definition: uses hierarchical hybrid. Use strong and weak model, use strong model to decide ranking and then use weak model to break tie.
  - Example: Netflix use content based model on movie names. Then use user collaborative filtering to break tie of which movies to recommend.



- Meta-level hybrid:
    - Definition: similar to feature augmentation but replace original data. Uses a model learned by on recommender as input for another.