Lec 04. Evaluation of Recommender Systems

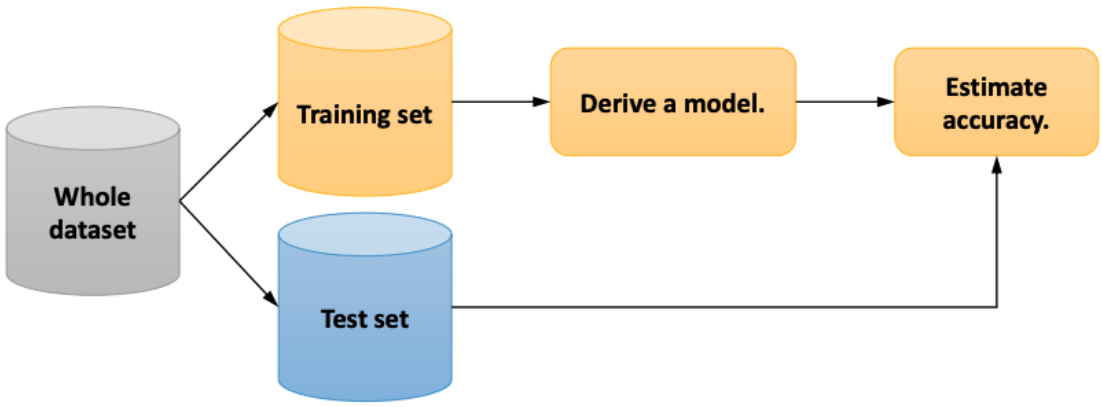
Contents

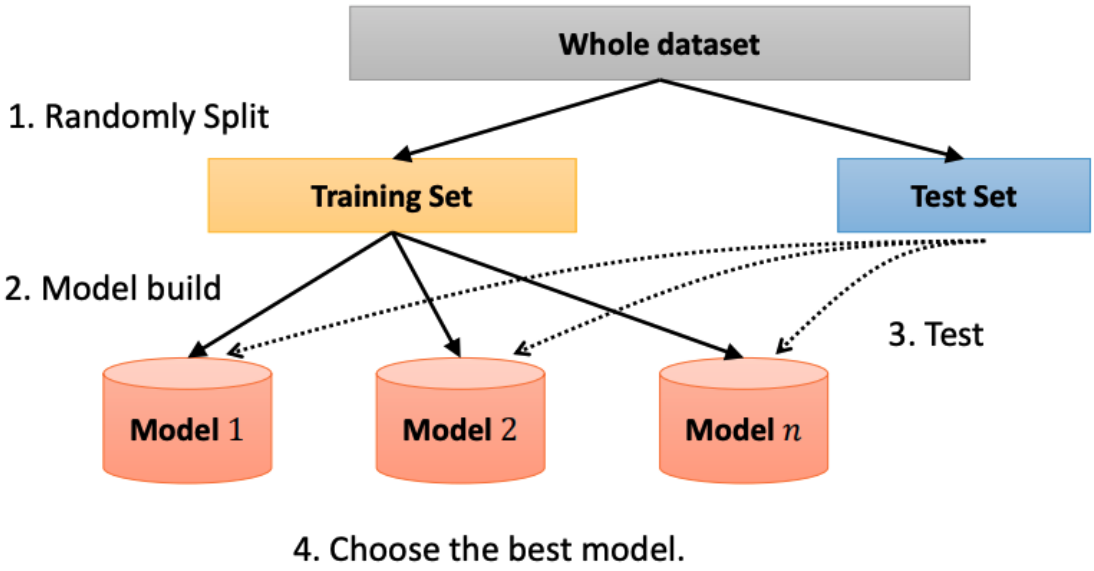
---

1. Evaluating Machine Learning Models

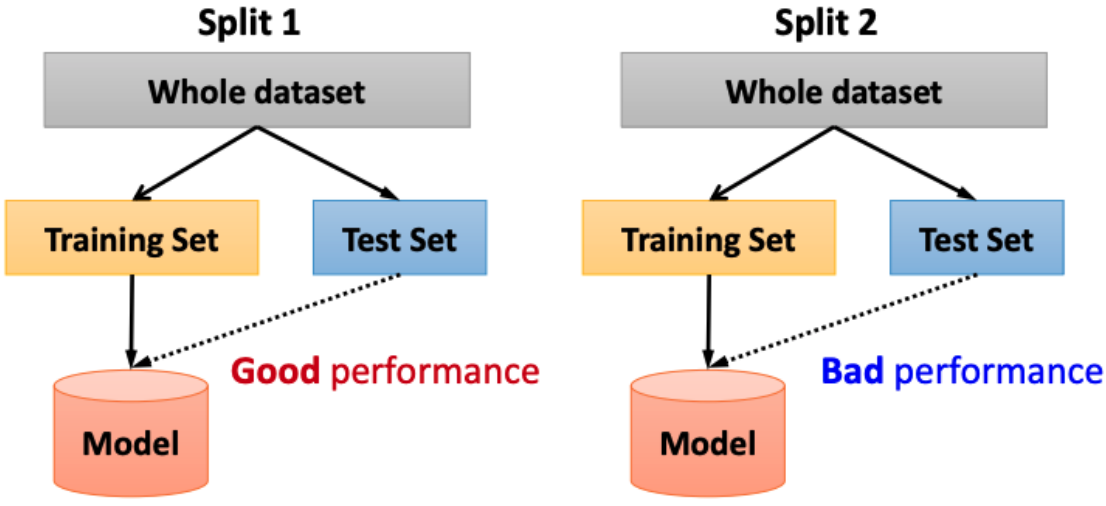
Hold-out Method

- Divide data into a **training set** and a **test set.**
  - The training set and the test set should Not overlap each other.
- How to choose a good model?
  - With the training set, build various models.
  - With the test set, evaluate each model.
  - Choose a model which shows the best performance with the test set.





- Advantage : Simple and easy
- Disadvantage
  - Random Split : Evaluation can be different depending on data split.

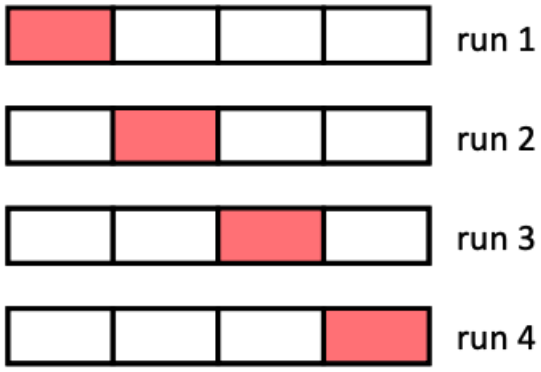


Cross Validation

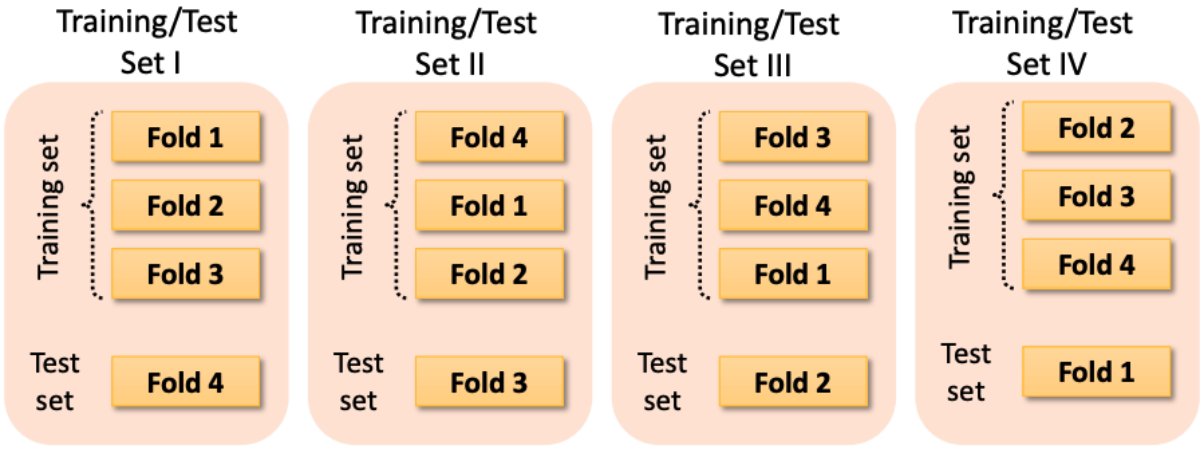
- Cross-validation (k-fold)

- Data D is randomly partitioned into **k mutually exclusive subsets** { $D_1, \ldots, D_k$}, each approximately equal size.

- Overall procedure
  - The data is partitioned into k groups.
    - $k-1$ of the groups are used for training the model.
    - One remaining group is used for evaluating the model.
  - Repeat procedure for all k choices
  - Performance from the k runs are averaged.



Example: 4-Fold Cross Validation



Choose a model by the average performance of four sets.

➢ Summary

- Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set.
- The average error across all k trials is computed.
- The variance is reduced as k is increased.

➢ Advantage

- Less dependent on how the data gets divided.
- Every data point gets to be in a test set exactly once and gets to be in a training set $k-1$ times.

➢ Disadvantage

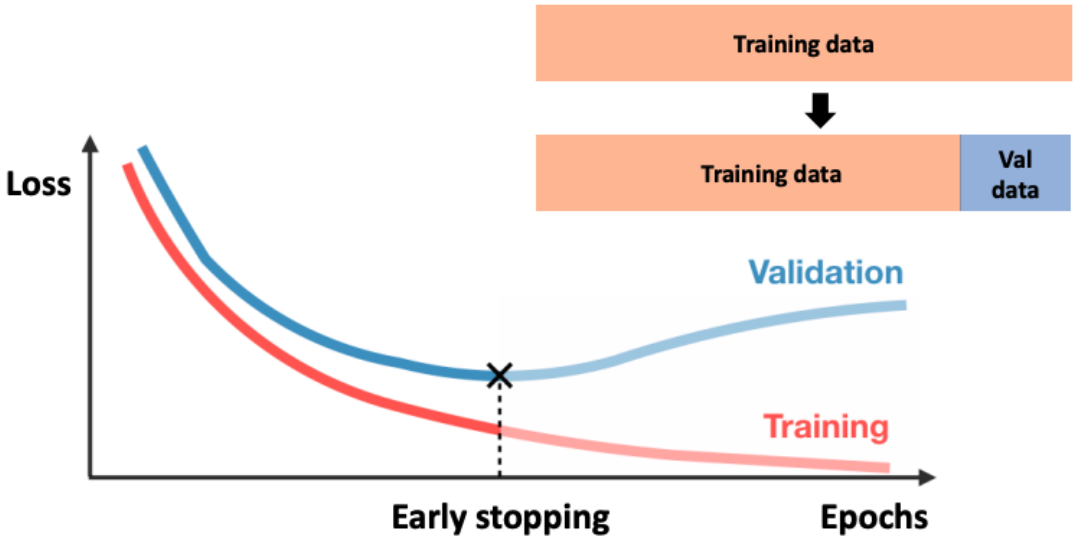
- Time!

Three Datasets for Evaluation

- Training dataset
  - It is used to fit the model.
- Validation dataset
  - It is used to provide an independent evaluation of a model fit on the training dataset while tuning model hyperparameters.
- Test dataset
  - It is used to provide an independent evaluation of a final model fit on the training dataset.

Early Stopping with the Validation set

- It is difficult to stop learning before converging too much.
- Usually, it is determined by a **validation set.**
  - The validation set is randomly chosen from the training set.
  - The validation set is NOT used for model training.



2. Evaluating Recommender Models

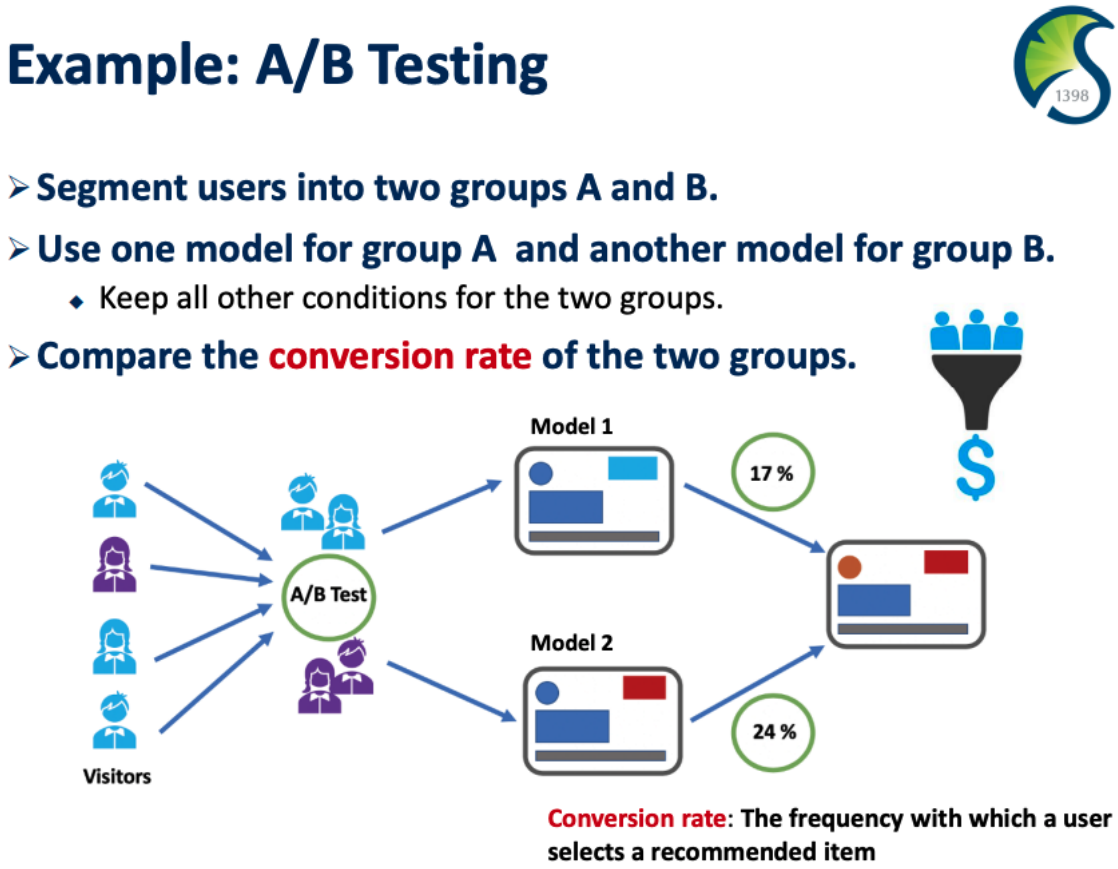
Key Questions about Evaluation

- Does the recommendation work well?
  - Do users like the recommended items?
  - Do they increase sales?
- Which algorithm is the most suitable for the service?
- A proper design for evaluation is crucial, but it is so difficult.
  - The evaluation is often multifaceted.
  - A single criterion cannot achieve many goals of the designer.
  - An incorrect design can lead to underestimation or overestimation of the true performance of a recommender model.

Online Evaluation

- User studies
  - Test subjects are actively recruited.
  - They are interacted with the recommender system to perform a task.

        - It is difficult and expensive to recruit large cohorts of users.
        - Recruited users can be biased, i.e., not the representative of real users.
- Online evaluation : **A/B testing**
    - Users are often **real users** in a commerical system.
    - It is less susceptible to bias from the recruitment process.
        - It is usually **not openly accessible.**
        - It is **limited** to use this method during the **start-up process.**
        - It is often **not generalizable** to system-independent benchmarks.
    - Recently, it is related to multi-armed bandit recommendation.



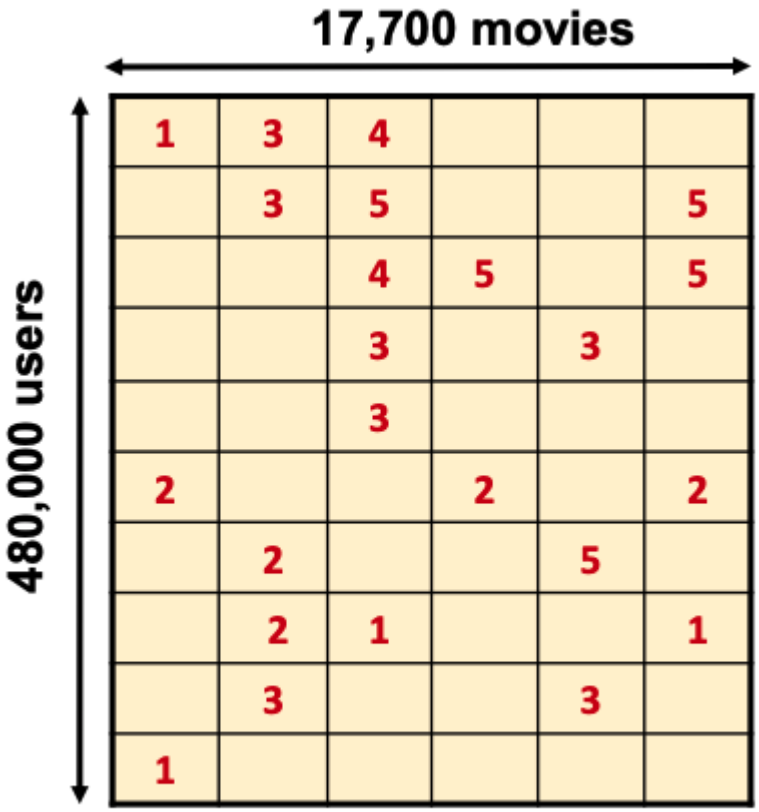
Offline Evaluation

- Offline evaluation
    - **Historical data**, e.g., **star ratings** or **click logs**
        - Temporal information, i.e., timestamp, may be associated with ratings.

- Advangate
  - It does **not require access to real users.**
  - It is the **most popular method** for testing recommender models.
- Disadvantage
  - They **do not measure the actual propensity(경향)** of users.
    - As data evolve over time, it may not reflect predictions for the future.

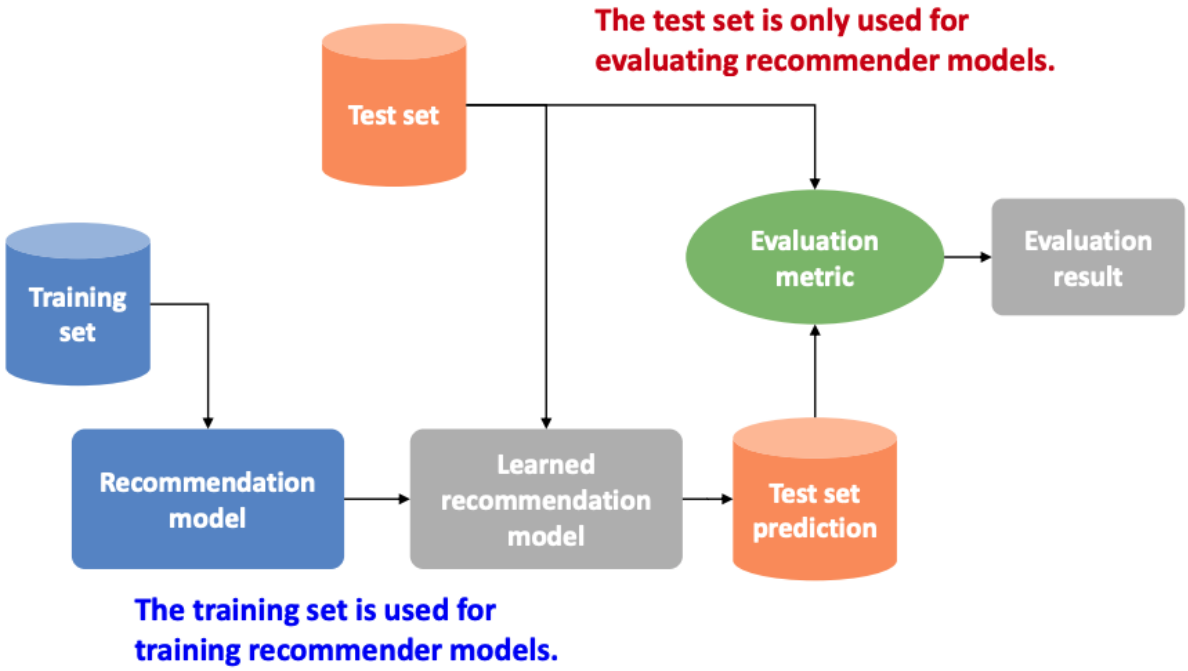
Example : Offline Evaluation

- 실제 데이터는 triplet : (user_id, item_id, ratings)
- we are given a **user-item rating matrix.**
- Commonly, it measures the difference between predicted and actual results.
  - Rating prediction
  - Top-N recommendation



The user-item matrix in Netflix

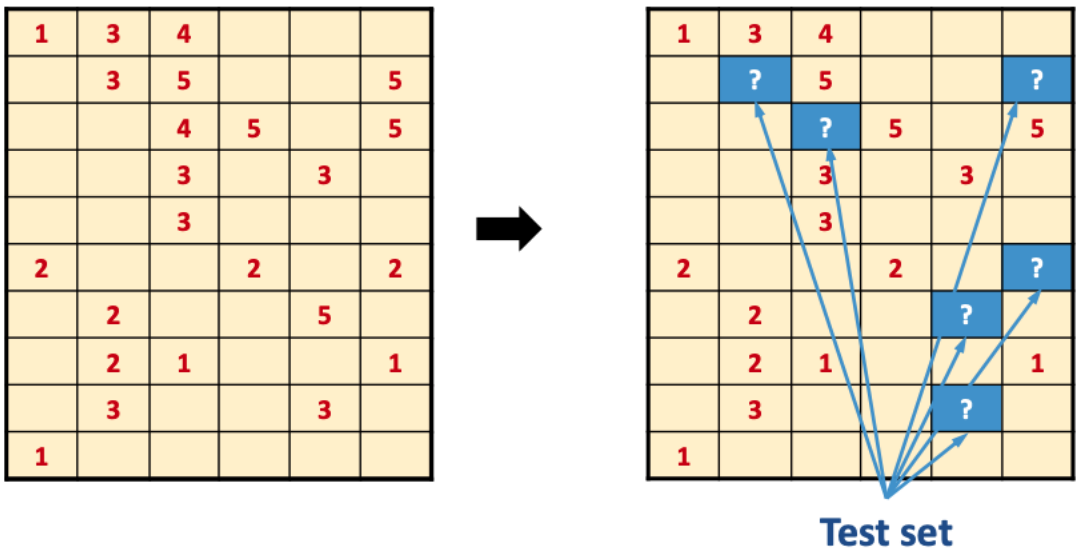
Workflow for Offline Evaluation



Data Split for Training/Test Sets

- Split data into training and test sets
  - **Training set** : used for setting model parameters
  - **Validation set** : A subset of the training set to simulate the test set
    - Model selection and parameter tuning
  - **Test set** : used for performance evaluation
- Train/test set division
  - Typical ratio : 80% train set and 20% test set
  - N-fold cross validation : N folds, in each turn one-fold is the test set.
- Common mistake : use the same data for parameter tuning and for testing.
  - Overestimates the accuracy because of the overfitting problem.

- Performance should not be **overestimated** or **underestimated.**
- The rating matrix is typically sampled in an **entry-wise way.**
  - Randomly hide some user-item ratings and predict them.



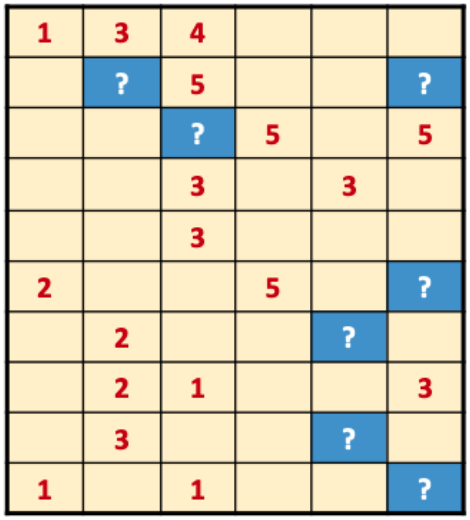
Case Study : Netflix Prize Data

- Proportional division of ratings
  - Validation data is considered as a part of training data.
  - When the sizes of the rating matrices are **large**, it is possible to use much smaller proportions for validation and testing.

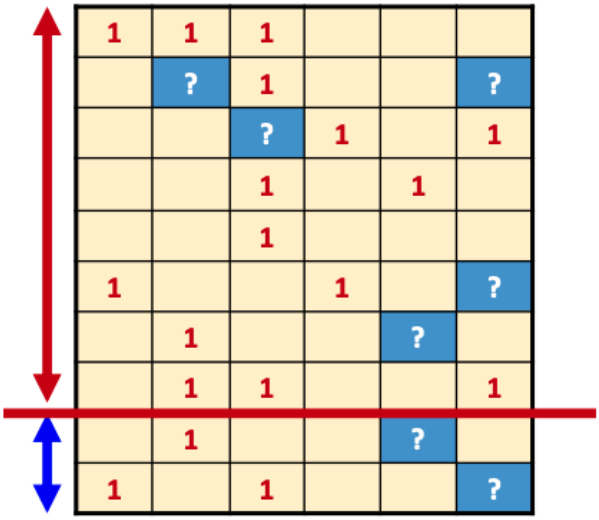
- Division in Netflix Prize data

Weak/String Generalization

- For each user, recommend top-N items from unseen items.
- Then, calculate the metrics for top-N items



Weak generalization: The test set is used as the evaluation for the same user.



Strong generalization: The test set is used as the evaluation for new users.

---

3. Various Evaluation Metrics

Various Evaluation Metrics

- Recommender systems have various goals and factors.
  - **Accuracy** : how well does it **correctly predict preferred items?**
  - **Coverage** : the proportion of recommended items out of an item set
  - **Confidence** : uncertainty about the accuracy of the prediction
  - **Novelty** : the proportion of items that the user is not aware of
  - **Serendipity** : the level of surprise in recommendations
    - All serendipitous items are novel, but the converse is not always true.
  - **Diversity** : how different items are recommended?
  - **Robustness** : how much is it affected in the presence of attacks?
  - **Scalability** : efficiency in large datasets
    - E.g., training time, prediction time, and memory consumption

Effectiveness

- The most fundamental measure to evaluate models
  - Rating-oriented(rating prediction) vs. ranking-oriented(top-n recommendation)
- Rating prediction
  - **Estimate rating values of items by regression.**
  - Usually, it is used for explicit feedback
  - E.g., NEtflix Prize competition
- Top-N recommendation
  - **Estimate the order of items by ranking.**
  - It is used for both explicit and implicit feedback.

Metrics for Rating Prediction

- Error is measured by the difference between predicted ratings and actual ratings.

Predicted $\hat{r}_{ui}$	Actual r_{ui}	Error $\hat{r}_{ui} - r_{ui}$
2.3	2	0.3
4.5	4	0.5
4.7	5	−0.3
2.1	3	−0.9
3.6	4	−0.4

$$MSE = \frac{1}{5}(0.3^2 + 0.5^2 + (-0.3)^2 + (-0.9)^2 + (-0.4)^2)$$

Mean squared error (MSE)

$$MSE = \frac{1}{|\mathcal{E}|} \sum_{(u,i)\in\mathcal{E}} (\hat{r}_{ui} - r_{ui})^2$$

Root mean squared error (RMSE)

$$RMSE = \sqrt{\frac{1}{|\mathcal{E}|} \sum_{(u,i)\in\mathcal{E}} (\hat{r}_{ui} - r_{ui})^2}$$

$\mathcal{E}$: a set of entries in the test set
r_{ui}: actual rating of entry (u, i)
$\hat{r}_{ui}$: predicted rating of entry (u, i)

Binary Prediction

- Examples : buy, click, view, and watch
- prediction : probability p
- Characteristics
  - Difficult to evaluate properly
  - Related to model evaluation for binary classification
- Metrics : log-likelihood

$$LL = \sum_{i=1}^{n} c_i \log(p_i) + (1 - c_i) \log(1 - p_i)$$

Metrics for Classification

- Build a confusion matrix for the relationship between actual and predicted results.

Relevant (or good) items

Ground truth

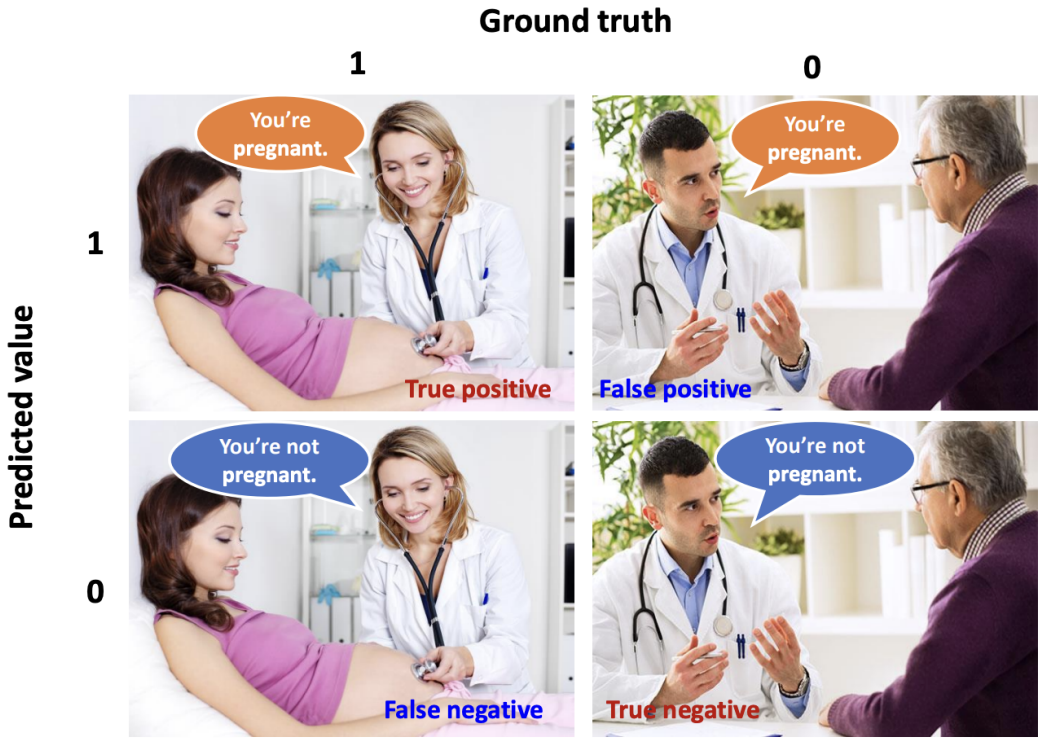
Recommended items

Predicted value

	Positive (1)	Negative (0)
Positive (1)	True Positive (TP)	False Positive (FP)
Negative (0)	False Negative (FN)	True Negative (TN)

- This can be used to measure accuracy, precision, recall, and so on.

Example: Confusion Matrix



Accuracy and Error Rate

- The fraction of these classifications that are correct

- Given an image, classify into “cat” or “No cat”.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$error\ rate = 1 - accuracy$$

		Ground truth: Positive (1)	Ground truth: Negative (0)
Predicted	Positive (1)	True Positive (TP)	False Positive (FP)
Predicted	Negative (0)	False Negative (FN)	True Negative (TN)

- They are ineffective in some domains.

Example: Accuracy

➢ **Which system is better in terms of accuracy?**

Dataset	*Actual*	Model A: *Predicted*	Model B: *Predicted*
x_1	+	+	−
x_2	−	+	−
x_3	−	+	−
x_4	−	−	−
x_5	−	−	−
x_6	−	−	−

A의 경우에 1/6, B의 경우에 5/6이다. 하지만 B모델은 좀 멍청?해보인다. - 만 선택했기 때문에..

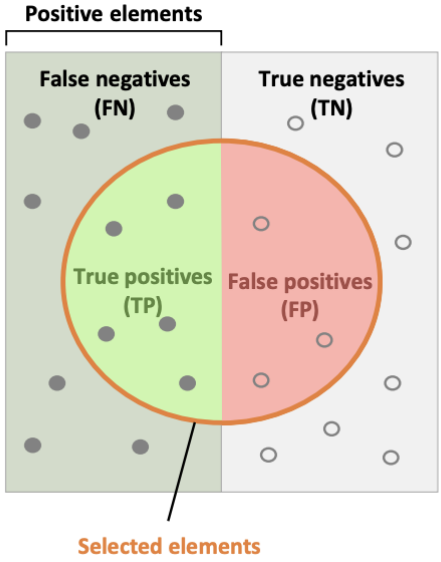
Why not just Use Accuracy?

- **99.9% of documents are irrelevant in most cases.**
  - Labeling every document as **irrelevant** has high accuracy but it is useless in the Web search engine.

Precision and Recall

- Precision : spam detection, document ranking
  - Exactness : How many selected items are positive?

$$Precision = \frac{TP}{TP + FP}$$



- Recall : medicat test
  - Completeness : How many positive items are selected?

$$Recall = \frac{TP}{TP + FN}$$

- The perfect score or both measures is 1.0.
  - In general, the **inverse** relationship between precision and recall

➢ **Recommendation is viewed as an information retrieval task**
- Retrieve (recommend) all items which are predicted to be **good**.

➢ **Precision: a measure of exactness**
- Determines the fraction of relevant items out of all items retrieved
- E.g., the proportion of recommended movies that are also good.

$$Precision = \frac{TP}{TP + FP} = \frac{|\#\ good\ movies\ in\ rec|}{|\#\ of\ rec\ movies|}$$

➢ **Recall: a measure of completeness**
- Determines the fraction of relevant items out of all relevant items
- E.g., the proportion of all good movies recommended.

$$Recall = \frac{TP}{TP + FN} = \frac{|\#\ good\ movies\ in\ rec|}{|\#\ of\ good\ movies|}$$

F-Measure (F-Score)

➢ F-measure (F_1 or F-score)

- The weighted **harmonic mean** of precision and recall

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where } \beta^2 = \frac{1-\alpha}{\alpha}$$

- When $\alpha = 0.5$ $(i.e., \beta = 1.0)$

$$F = \frac{2PR}{P+R}$$

➢ Why harmonic mean?

- The harmonic mean is **always less than or equal to** the arithmetic mean and the geometric mean.
- When P and R differ greatly, **the harmonic mean is closer to their minimum** than to their arithmetic mean.

Challenges of Precision/Recall

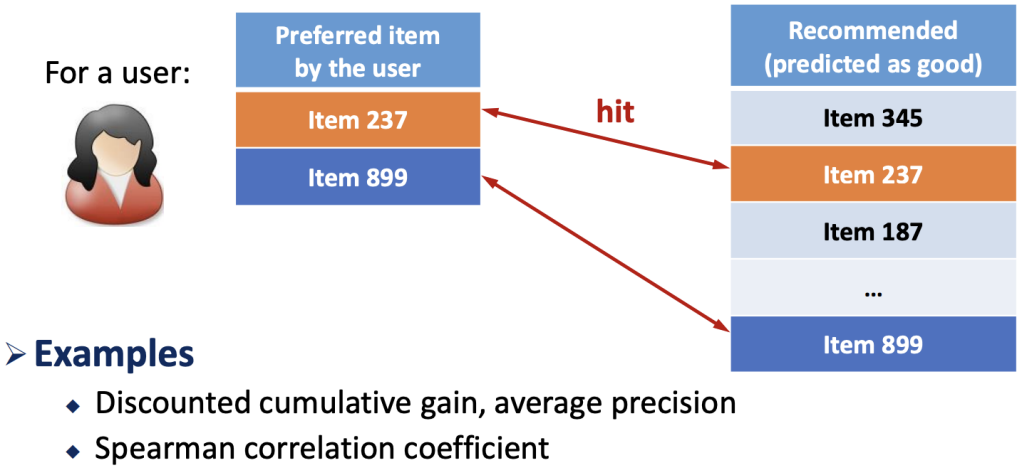
- Problem with precision/recall
  - **Need the ground truth for all items.**
  - If we have all ground truth, why do we bother with recommenders?
- **Typical output of RS : the ranked list of items**
  - Swap on the first-place matters more than swap on the 10th place.
- **Ranking metrics** : the extension for precision and recall

---

4. Ranking-aware Evaluation Metrics

Ranking Metrics : Positions Matter

- Extend **recall** and **precision** to take the positions of current items in the ranked list into account.
    - Relevant items are more useful when they appear earlier in the recommendation list.



Relevant vs. Recommended

relevant : 사용자가 실제로 관심 있는 item (preferred item)

recommended : 모델이 예측한 item

- A **relevant item** for a specific user-item pair means that this item is a **good recommendation** for the user.
- A **recommended item** means that this item is **provided by a recommender model** to the user.
- We are interested in **quantifying how well recommended items are related to relevant items.**

Ranking-aware Accuracy Metrics

- Precision and recall metrics can be extended.
    - **Neglect the ranking of relevant items** in the recommendation list.
- Representative ranking-aware metrics
    - **Mean reciprocal rank (MRR)**
    - **Mean average precision (MAP)**

- **Normalized discounted cumulative gain (NDCG)**

- They are commonly used for **search** and **recommender systems**.

Precision and Recall at N

 # good mives in rec ant N = **# of relevant item set** ∩ **# of recommended item set**

good movies = # preferred movies(relevant movies)

➢ **Precision at N is the proportion of items in the top-N set that are also relevant.**

$$Precision@N = \frac{|\#\ good\ movies\ in\ rec\ at\ N|}{N}$$

➢ **Recall at N is the proportion of relevant items found in the top-N recommendations.**

$$Recall@N = \frac{|\#\ good\ movies\ in\ rec\ at\ N|}{|\#\ of\ good\ movies|}$$

Example: Precision@3 and Recall@3

- Actual preferred movies are in red.
- Given top-3 items, how to calculate precision and recall at 3?

N이 커지면 보통 Recall 은 증가, precision은 감소

precision과 recall은 순서 고려 X

Mean Reciprocal Rank (MRR)

- Find the rank k_u of the first relevant recommendation.
- The reciprocal rank is computed by $1/k_u$.

$$RR_u = \sum_{k \le \min(k_u, N)} \frac{rel(k)}{rank(k)}$$

- $rank(k)$ is the **ranking** of the ***k*-th item**.
- $rel(k)$ is an **indicator** equal to **1** if the item at k is **relevant**, **0** otherwise.

- It is computed by the average for all the users.

$$MRR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} RR_u$$

Example: Reciprocal Rank (RR)

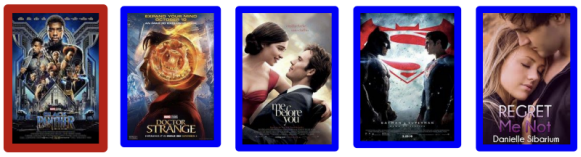
- Actual preferred movies are in red.
- Given top-3 items, how to calculate reciprocal rank at 3?

Example: Mean Reciprocal Rank (MRR)

➢ **How to compute mean reciprocal rank for two users?**

Reciprocal rank (RR)

$1/3$



$1/1$

$\text{MRR} = (1/3 + 1/1)/2 = 2/3$

➢ **Red: preferred, Blue: non-preferred**

Average Precision (AP)

- Computing the average value of precisions over the interval from recall = 0 to recall = 1

$$AP = \frac{\sum_{k=1}^{n} prec(k) rel(k)}{number\ of\ relevant\ items}$$

- $prec(k)$ is the **precision** at cut-off k in the list.
- $rel(k)$ is an **indicator** equal to **1** if the item at k is **relevant**, **0** otherwise.

AP : precision@ N(N=1,2,…N) 각각의 precision을 평균낸 것

- It is calculated by the average precision while the precision at k is computed for all the relevant items.

Example: Average Precision (AP)

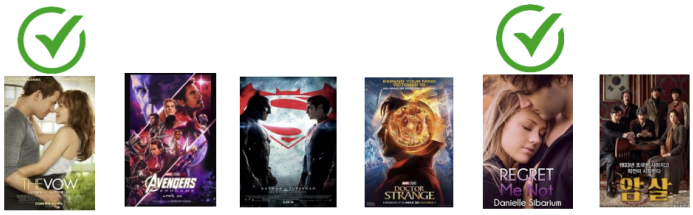
➢ **Recommender systems sort the items by predicted scores.**

(1/1 + 1/2 * 0 + 2/3 * 1 + 0/4 + 0/5 + 6/0)/2 = 5/6

@을 안주면 끝까지 다계산

Example: Average Precision (AP)

- Which is better for average precision?



1/1 0 0 0 2/5 0

$$AP = \frac{1}{2}\left(\frac{1}{1} + \frac{2}{5}\right) = \frac{7}{10}$$

0 1/2 2/3 0 0 0

$$AP = \frac{1}{2}\left(\frac{1}{2} + \frac{2}{3}\right) = \frac{7}{12}$$

Mean Average Precision (MAP)

- For query, we can calculate the corresponding AP.
- For recommender systems, each user represents a query.
- It is computed by the average for all the users.

$$MAP = \frac{1}{|\mathcal{U}|}\sum_{u \in \mathcal{U}} AP_u$$

$\mathcal{U}$: a set of users

Normalized Discounted Cumulative Gain

➢ Ideal discounted cumulative gain (IDCG)

- Assumption: items are sorted by relevance in decreasing order.

$$IDCG_{pos} = \sum_{i=1}^{pos} \frac{1}{\log_2(i+1)}$$

➢ Actual discounted cumulative gain (DCG)

- Logarithmic reduction factor

$$DCG_{pos} = \sum_{i=1}^{pos} \frac{rel_i}{\log_2(i+1)}$$

- pos is the position up to which relevance is accumulated.
- rel_i returns the relevance of recommendation at position i.

➢ DCG is normalized to the interval in [0, 1].

Example: NDCG

➢ **Recommender systems predict top-3 items.**

$$NDCG_3 = \frac{DCG_3}{IDCG_3}$$

Ours

Rank	Relevance
1	
2	Yes
3	Yes
4	
5	

Ideal

Rank	Relevance
1	Yes
2	Yes
3	
4	
5	

$$DCG_3 = \frac{1}{\log_2 3} + \frac{1}{\log_2 4}$$

$$IDCG_3 = \frac{1}{\log_2 2} + \frac{1}{\log_2 3}$$

Yes는 모델이 추천한 것을 표기

Example: NDCG

➢ **Recommender systems predict top-3 items.**

$$NDCG_3 = \frac{DCG_3}{IDCG_3}$$

Ours

Rank	Relevance
1	
2	Yes
3	Yes
4	
5	Yes

Ideal

Rank	Relevance
1	Yes
2	Yes
3	Yes
4	
5	

$$DCG_3 = \frac{1}{\log_2 3} + \frac{1}{\log_2 4}$$

$$IDCG_3 = \frac{1}{\log_2 2} + \frac{1}{\log_2 3} + \frac{1}{\log_2 4}$$

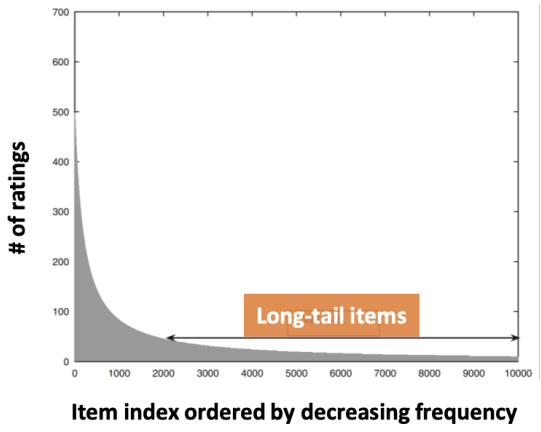
Limitation of Accuracy Metrics

➢ **Users tend to rate more popular items.**

⇒ **Missing entries in the rating matrix are NOT random.**

➢ **The rating distribution is missing-not-at-random.**

- When an item is popular, it is more likely to be considered being a preferred item.
- It incurs a **selection bias**.



57

Beyond Accuracy Metrics

- Coverage : **How many items** can recommender models **make predictions for all users?**
  - 항상 비슷한 아이템들이 추천될수도 있다.
  - How to address long-tail items?
  - Model A provides better accuracy than model B.
  - Model A recommends only a subset of easy-to-recommend items.
- Diversity : How wide the **spectrum** of recommended items is?
  - How many different categories/genres?
  - How many different artists/authors/sellers?
  - How different(i.e., distant) are the item embeddings?
    - Cosine similarity is used for the similarity between two items.

Beyond Accuracy Metrics

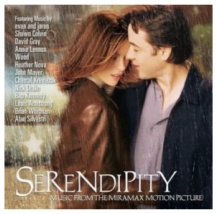
- **Novelty: how unusual are the recommendations for users?**

$$Novelty(i) = \frac{\#\ of\ users\ recommended\ item\ i}{\#\ of\ all\ users}$$

- **Serendipity: measuring unexpectedness multiplied by relevance**

$$Serendipity(i) = Unexpectedness(i) \times relevance(i)$$

  - Unexpectedness measures how **surprising** the recommended items are.



serendipity

(n.) finding something good without looking for it