UNIVERSITAT DE BARCELONA

Master on Foundations of Data Science

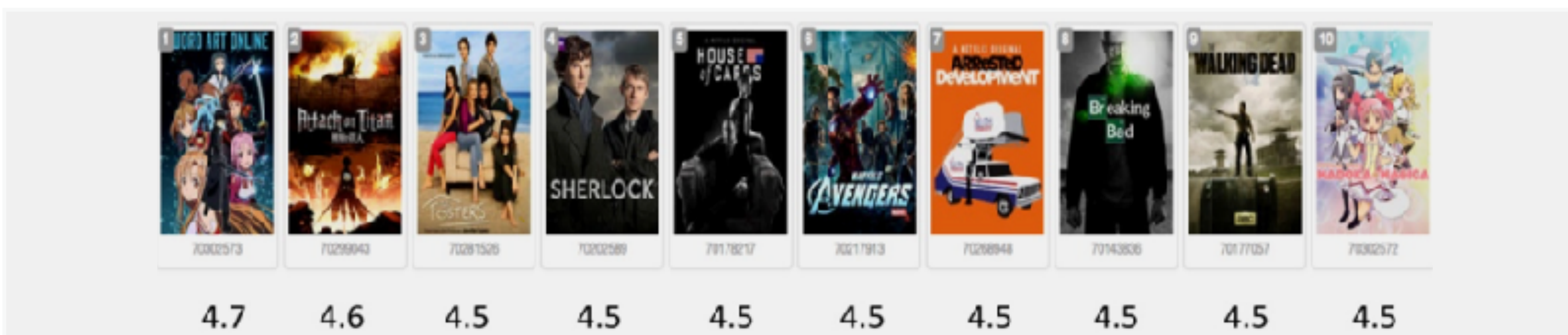
Recommender Systems

Learning to rank

Santi Seguí | 2019-20

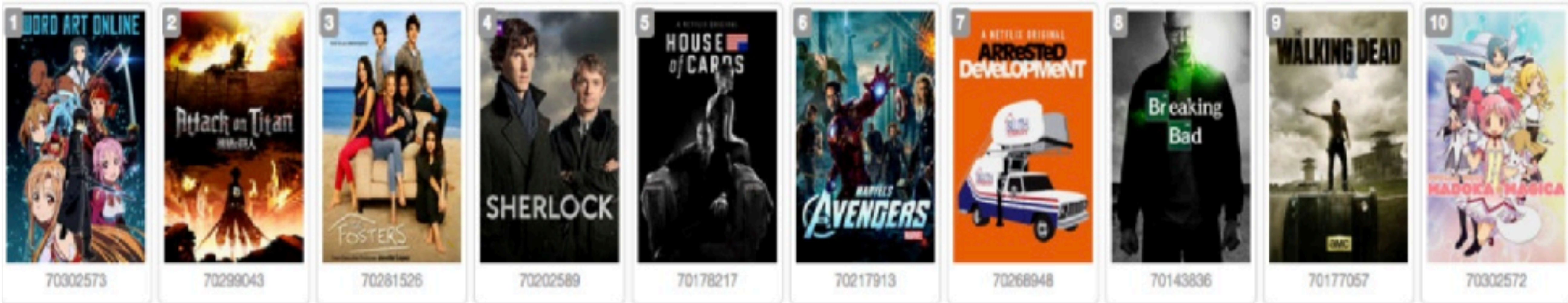
Ranking

- Most of the recommendations are **presented in a sorted list**.



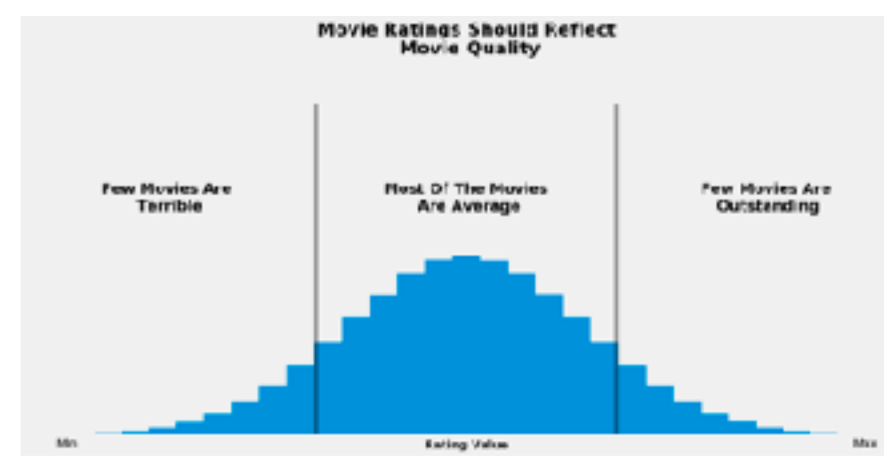
- Recommentation can be understood as a ranking problem.
  - Popularity could be always considered as your baseline

Top-ranking



4.7 4.6 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5

- If we serve the recomentation as a top-ranked list **RMSE**, **MAE** and other similar metrics are not appropiate to evaluate how good is the system.



How to rank?

Learning to rank - Metrics

- The quality of a ranking can be measured using metrics like:
  - Normalized Discount Cumulative Gain
  - Mean Reciprocal Rank (MBR)
  - Fraction of Concordant Pairs (FCP)
  - and many others…
- But, it is **hard to optimize** a machine learning model using these **measure** since the are **not differenciable**.

Learning to rank

LTR solves a ranking problem on a list of items.

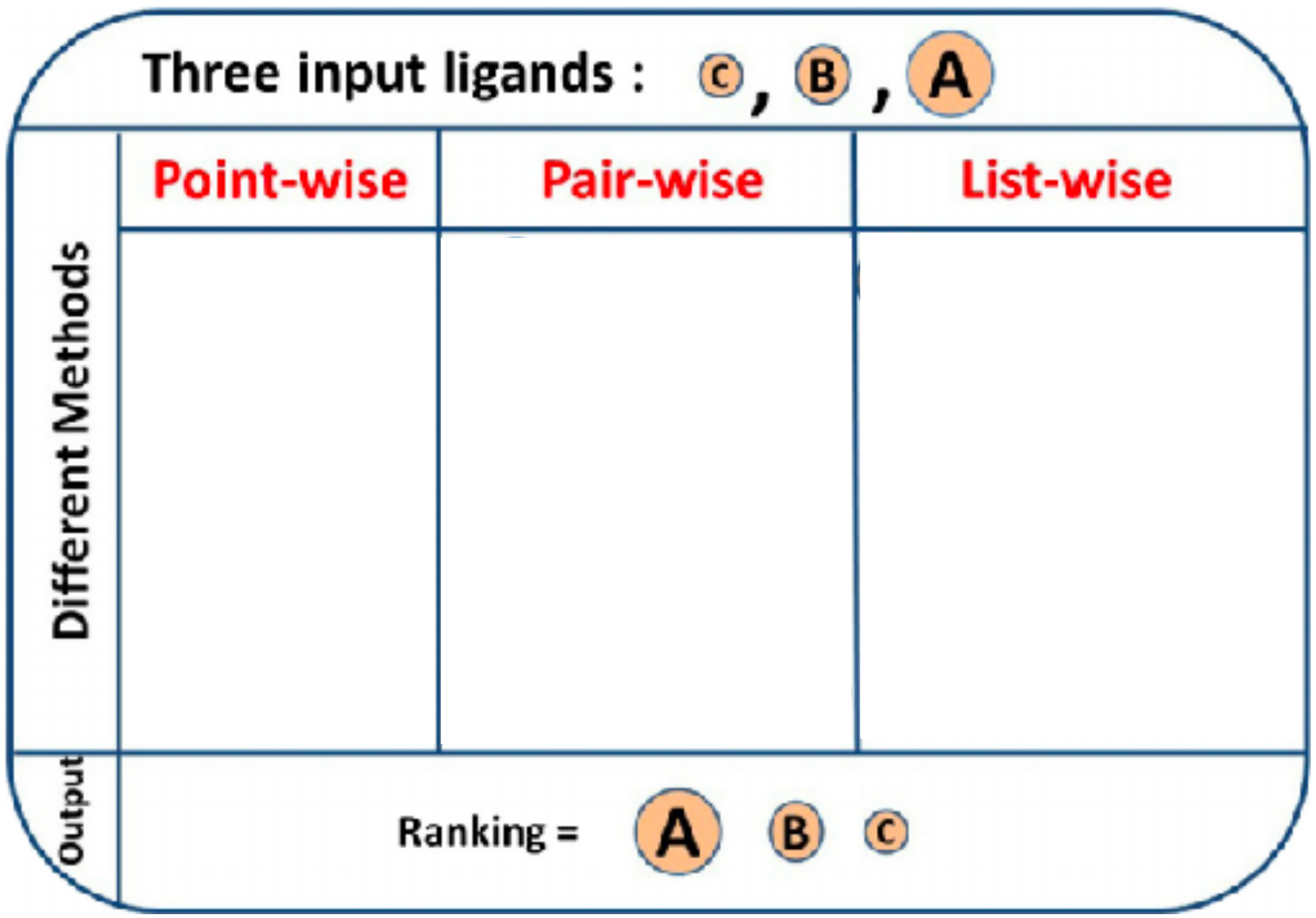
The aim of LTR is to come up with optimal ordering of those items.

LTR doesn't care much about the exact score that each item gets, but cares more about the relative ordering among all the items.

Learning to rank

- Using Machine Learning, the goal is to **construct a ranking model** from the training data.

- The problem can be treat as a standard classification problem



Point-wise

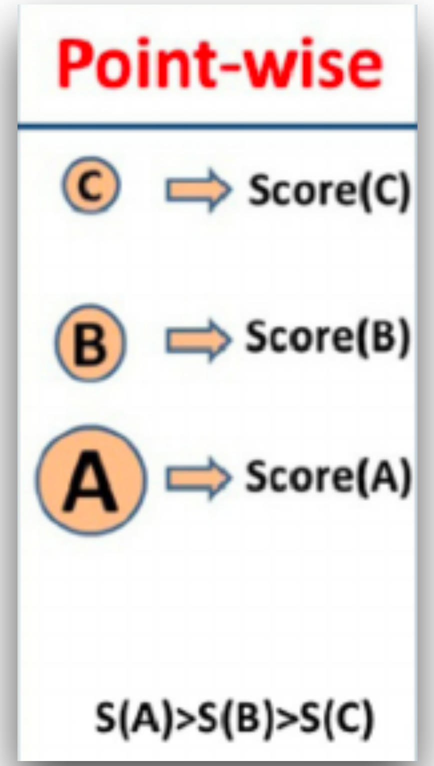
- Pointwise approaches look at a **single item** at a time in the loss function. They essentially take a single item and train a classifier / regressor on it to predict how relevant it is for the current query/user. The final ranking is achieved by simply sorting the result list by these document scores.

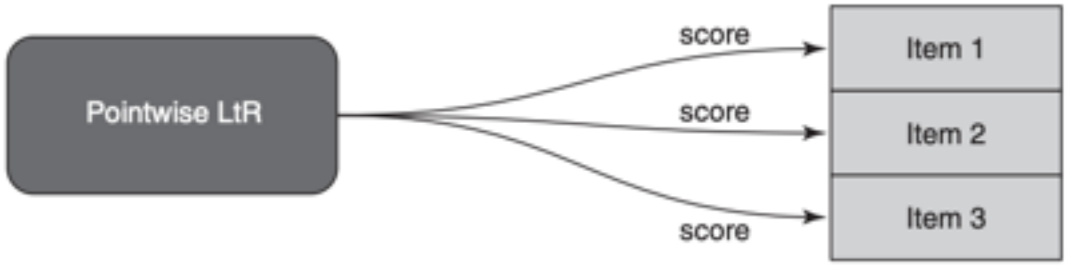
- For pointwise approaches, the score for each item is independent of the other items that are in the ranking list.

- All the standard regression and classification algorithms (SVM, XGBOOST, NN,…) can be directly used for pointwise learning to rank.

    - These methods tries are trained trying to minimize the error on the rating prediction… Usually RMSE, CE, or other similar loss function…

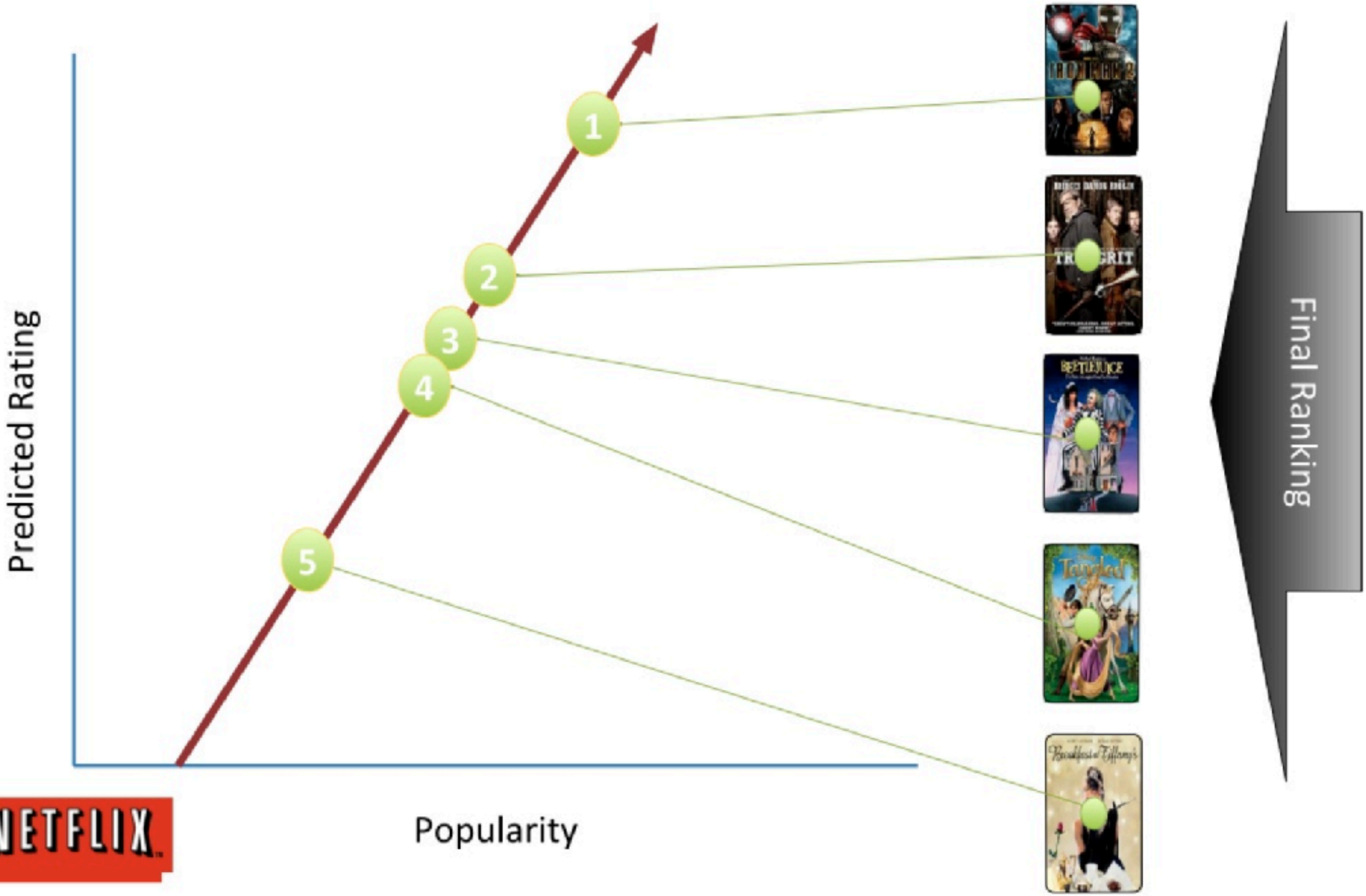
    - **Problem**: Obtained accuracy is highly biased to popular items.





UNIVERSITAT DE BARCELONA

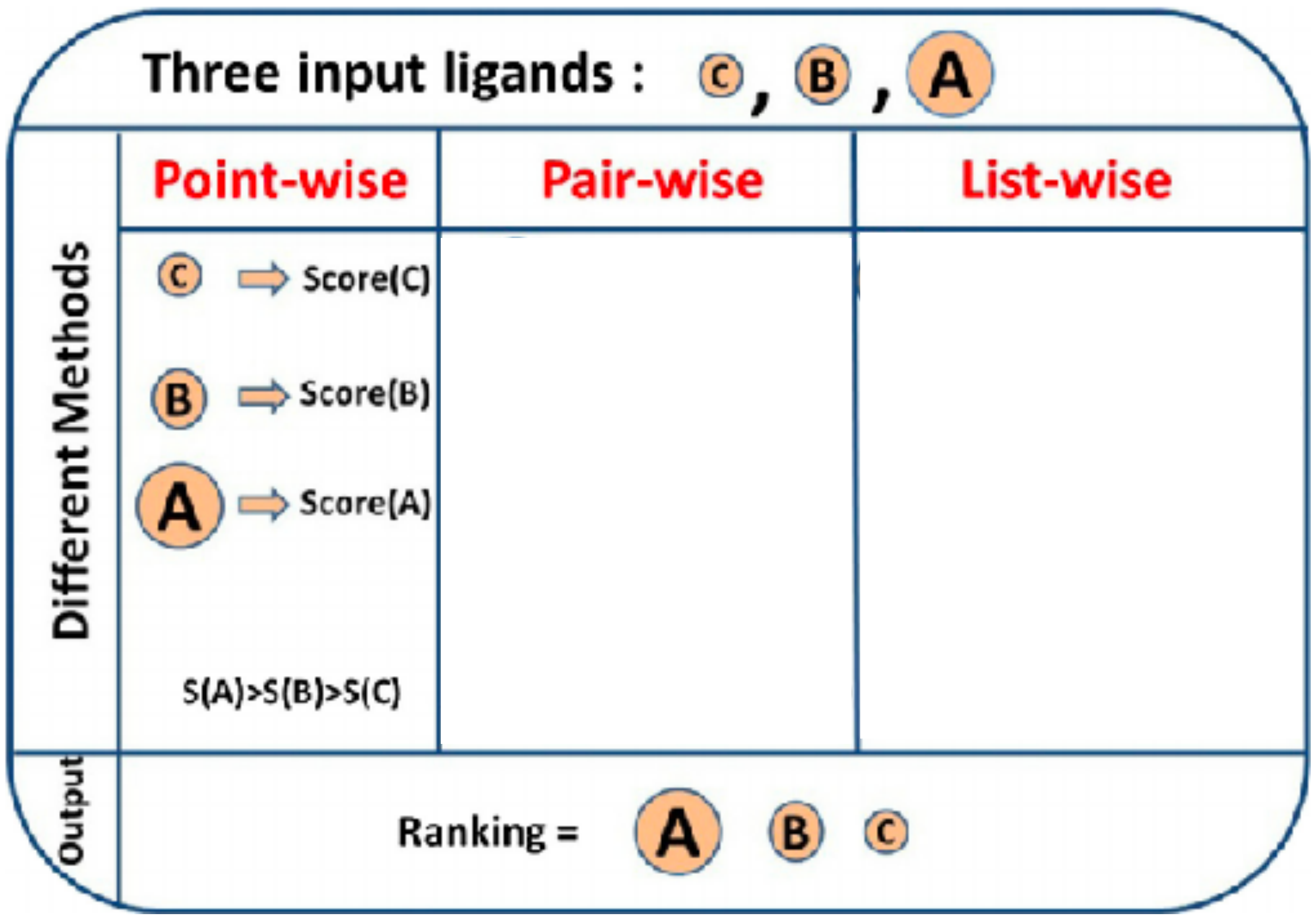
Example: Two features, linear model



NETFLIX

Learning to rank

- Using Machine Learning, the goal is to **construct a ranking model** from the training data.
- The problem can be treat as a standard classification problem



Instead of focusing on recommendations as a rating prediction problem, it sometimes makes more sense to look at **how the items should be stacked**

Pairwise approach

- **General Criteria**: The ranking function f learns tor rank pairs of items (i.e. for $\{x_i, x_j\}$, is y_i greather than y_j?).

Pairwise approach

- **General Criteria**: The ranking function f learns tor rank pairs of items (i.e. for $\{x_i, x_j\}$, is y_i greather than y_j?).

- predict for every **pair of items** based on feature vector x
- users' **relative preference** regarding the documents.
- training determines the **parameter** θ based on a **loss function** (e.g. the number of inverted pairs)

Advantage:
Models relative order

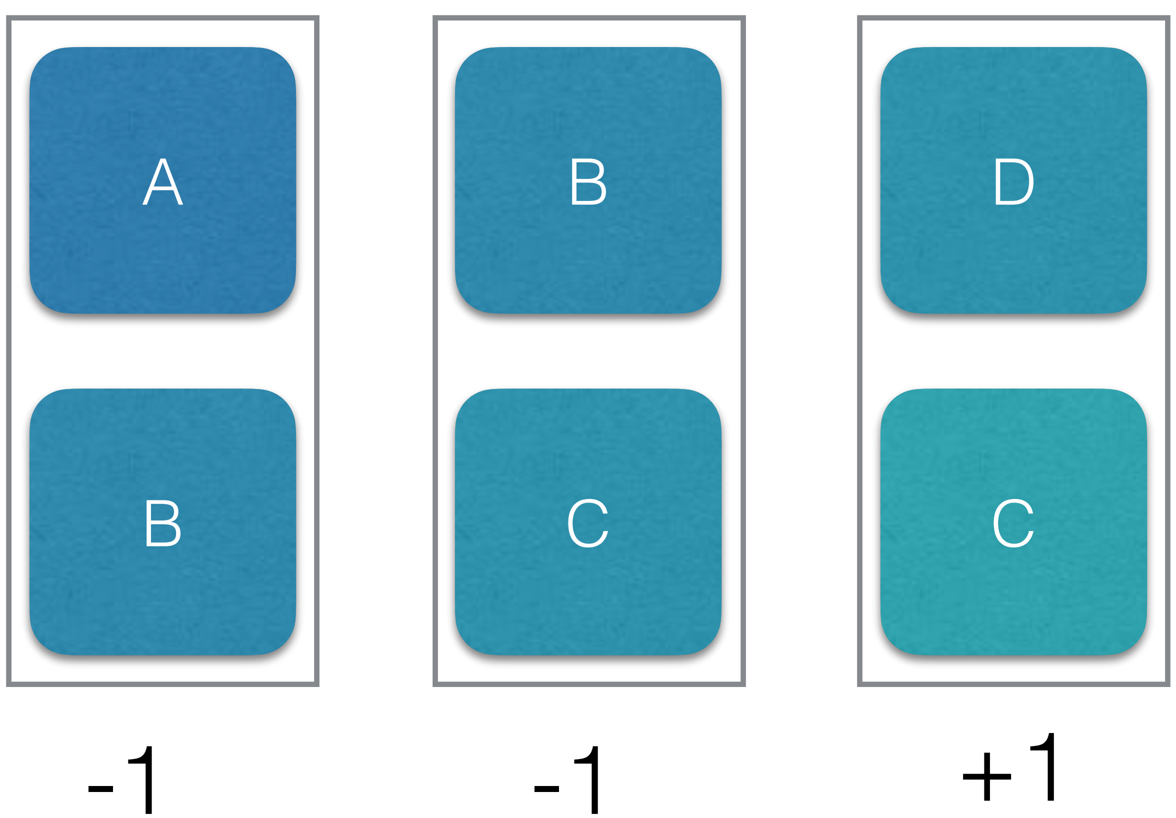
Main disadvantage:
- no disctinction between excellent-bad and fair-bad
- sensitive tot noise labels.

Pairwise approach

Data

Item	A	B	C	D	E
Score	1	2	3	4	5

- Training data instances are item pairs in learning



With labels:
+1 (i>j) or **-1** (j<i)

Pairwise approach

- **Several Methods**:
  - RANK SVM
  - RankBoost
  - RANK NET
  - Lambda Rank
  - Lambda Mart

RankNet

- RankNet was originally developed using neural nets.

- **The cost function for RankNet aims to minimize the number of *inversions* in ranking**. Here an inversion means an incorrect order among a pair of results, i.e. when we rank a lower rated result above a higher rated result in a ranked list. RankNet optimizes the cost function using Stochastic Gradient Descent.

Learning to Rank using Gradient Descent

Keywords: ranking, gradient descent, neural networks, probabilistic cost functions, internet search

Chris Burges CBURGES@MICROSOFT.COM
Tal Shaked* TAL.SHAKED@GMAIL.COM
Erin Renshaw ERINREN@MICROSOFT.COM
Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399

Ari Lazier ARIEL@MICROSOFT.COM
Matt Deeds MADEEDS@MICROSOFT.COM
Nicole Hamilton NICHAM@MICROSOFT.COM
Greg Hullender GREGHULL@MICROSOFT.COM
Microsoft, One Microsoft Way, Redmond, WA 98052-6399

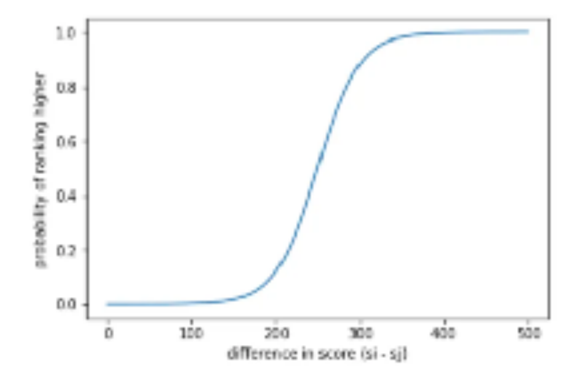
RankNet

- RankNet [Burges et al., 2005] is a pairwise loss function—popular choice for training neural L2R models and also an industry favourite [Burges, 2015]

- Predicted probabilities:

$$p_{i,j} = p(s_i > s_j) = \frac{1}{1 + e^{-\gamma(s_i - s_j)}}$$

$$p_{j,i} = \frac{1}{1 + e^{-\gamma(s_j - s_i)}}$$



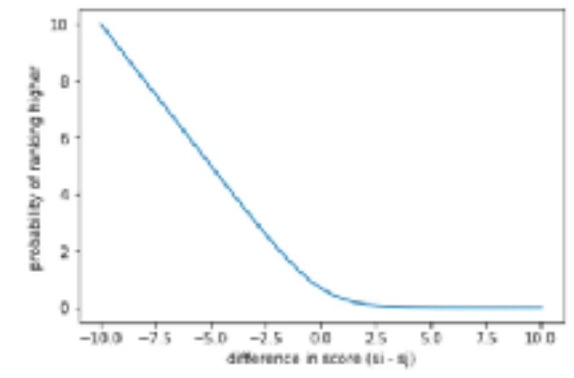
- Desired probabilites:

$$\bar{p}_{ij} = 1 \text{ and } \bar{p}_{ji} = 0$$

- Computing cross-entropy between $\bar{p}$ and p

$$L_{RankNet} = -\,\bar{p}_{ij} log(p_{ij}) - \bar{p}_{ji} log(p_{ji}) = -\,log(p_{ij})$$

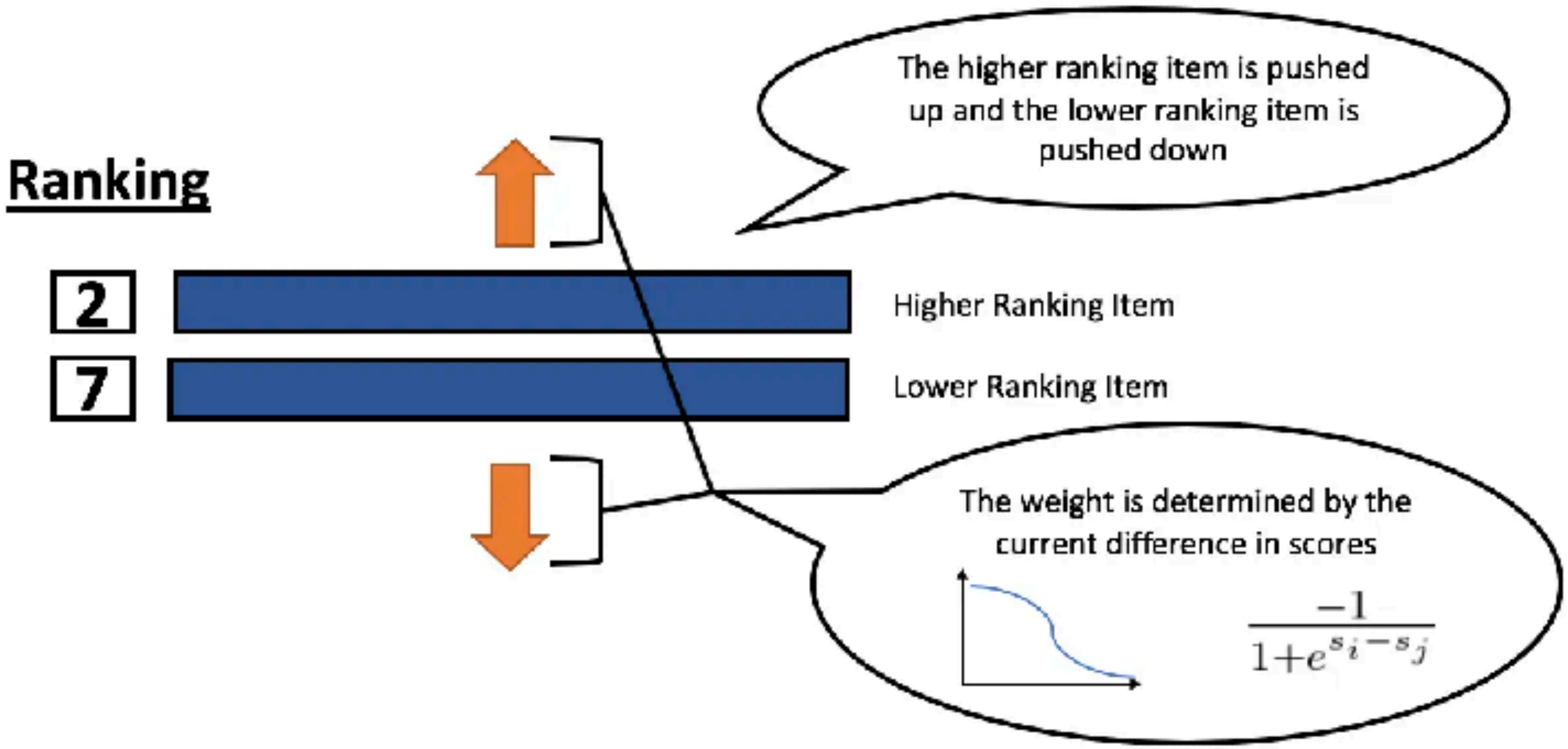
$$= log(1 + e^{-\gamma(s_i - s_j)})$$



RankNet

RankNet and is pretty good in many situations. There is one major pitfall to the RankNet approach though: the loss is agnostic to the **actual ranking** of the item.

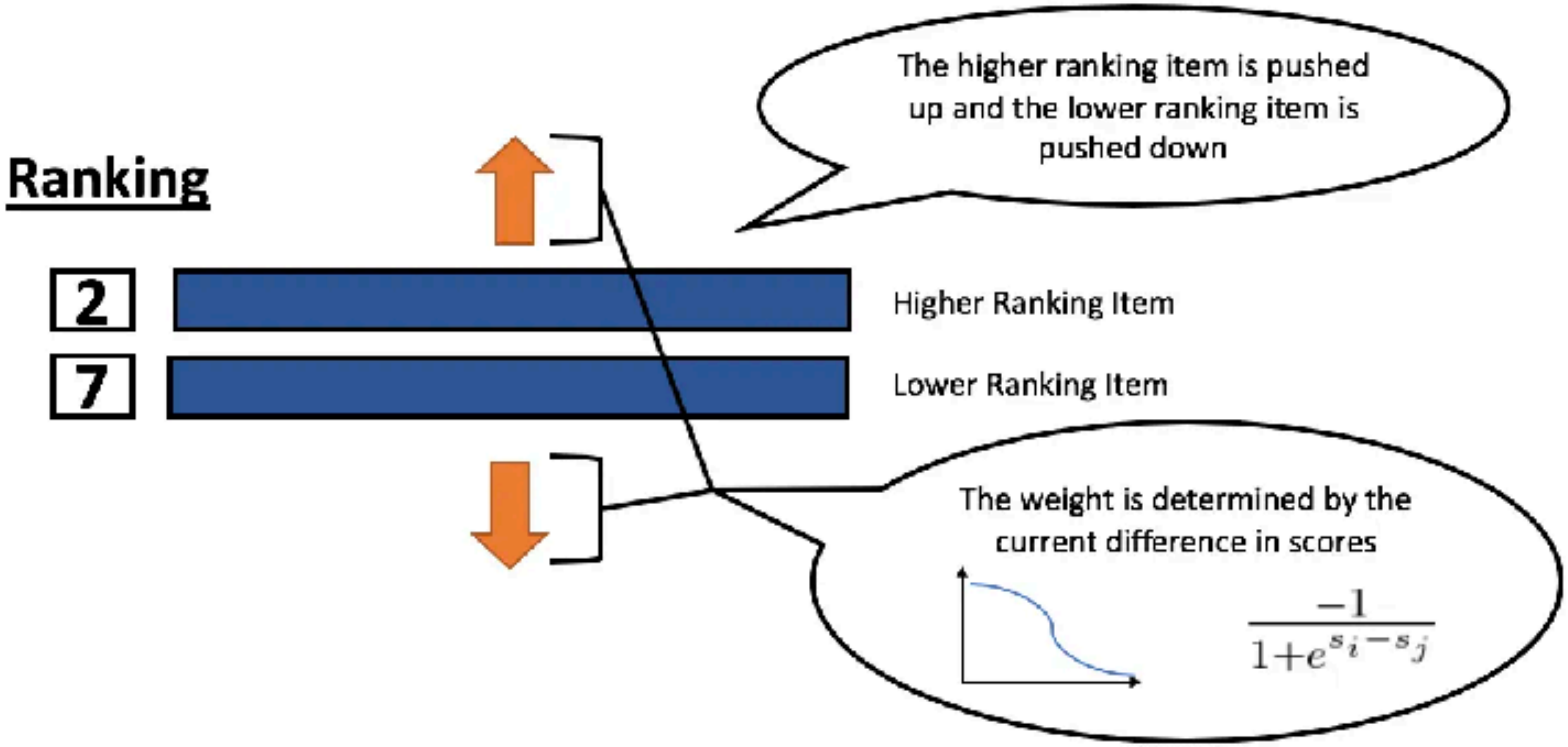
The strength of the push is determined only by the current difference in scores. So it doesn't matter if the items that rank below are 1, 2, or 500 ranks below


Ranking
2
7
Higher Ranking Item
Lower Ranking Item
The higher ranking item is pushed up and the lower ranking item is pushed down
The weight is determined by the current difference in scores
$\frac{-1}{1+e^{s_i - s_j}}$

Lambda Rank

- Burgess et. al. found that during RankNet training procedure, you don't need the costs, only need the gradients (λ) of the cost with respect to the model score. You can think of these gradients as little arrows attached to each document in the ranked list, indicating the direction we'd like those documents to move.

- Further they found that scaling the gradients by the change in NDCG found by swapping each pair of documents gave good results. The core idea of LambdaRank is to use this new cost function for training a RankNet. On experimental datasets, this shows both speed and accuracy improvements over the original RankNet.



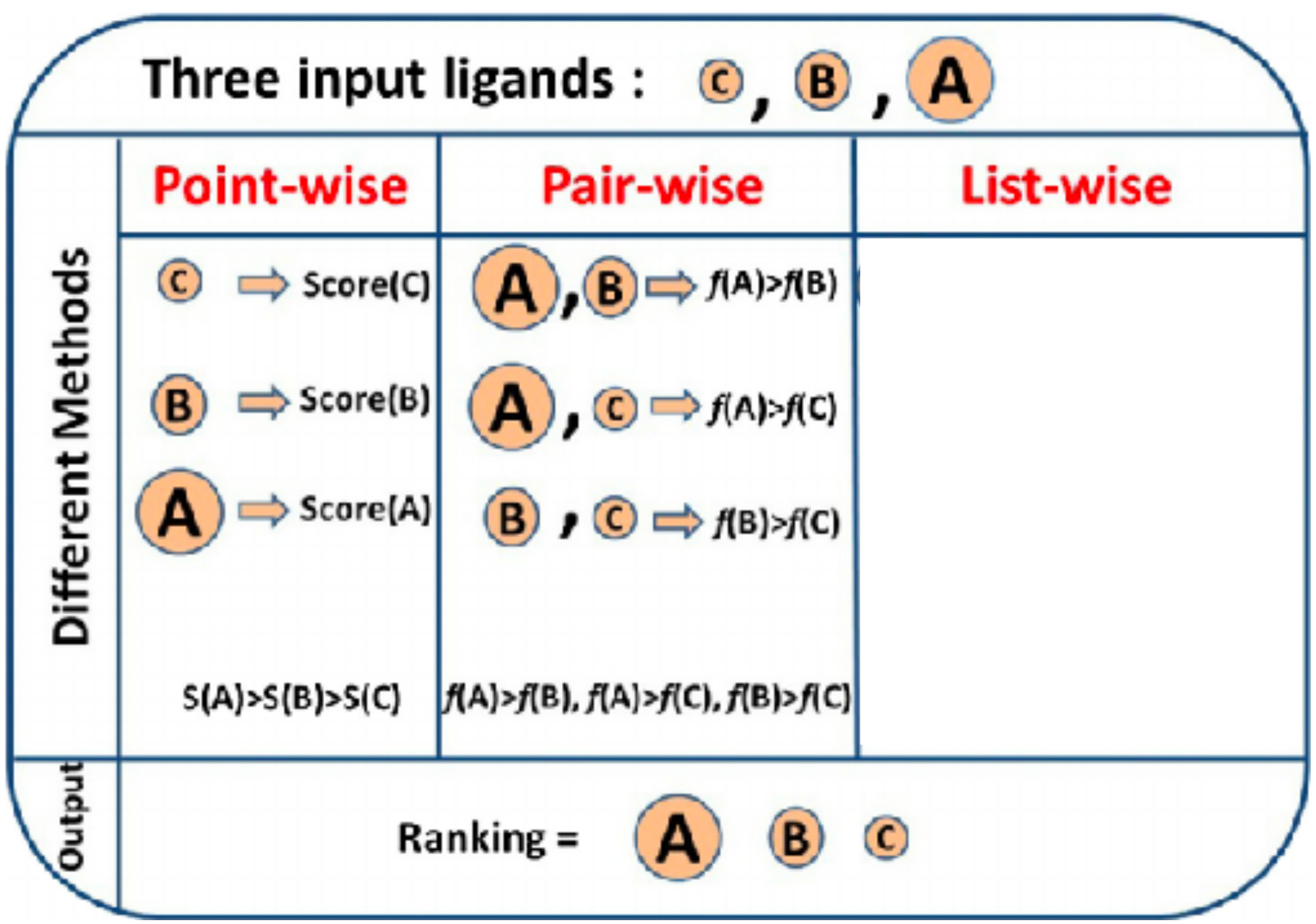
Now, $\lambda_{ij} = \dfrac{-\Delta(i,j)}{1+e^{(s_i-s_j)}}$ where $\Delta(i,j)$ is a penalty corresponding to how "bad" it is to rank items i and j in the wrong order.

Multiple metrics can be used to compute Δ, but the most commin is the NDCG (normalized discounted cumulative gain)

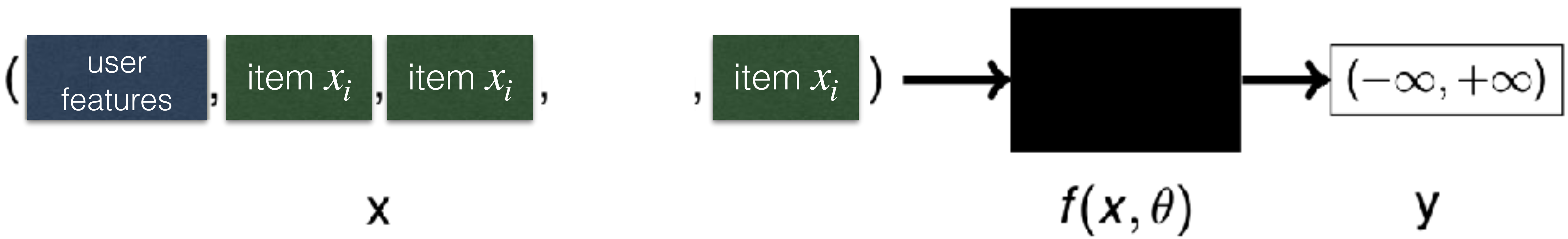
Learning to rank

- Using Machine Learning, the goal is to **construct a ranking model** from the training data.

- The problem can be treat as a standard classification problem



Listwise approach



- predict for ranked list of documents based on feature vector x
- **effectiveness** of ranked list **y** (e.g., MAP or nDCG)
- training determines the **parameter** θ based on a **loss function**

Advantage:
positional infirmation visible to loss function

Main disadvantage:
- high training complexity

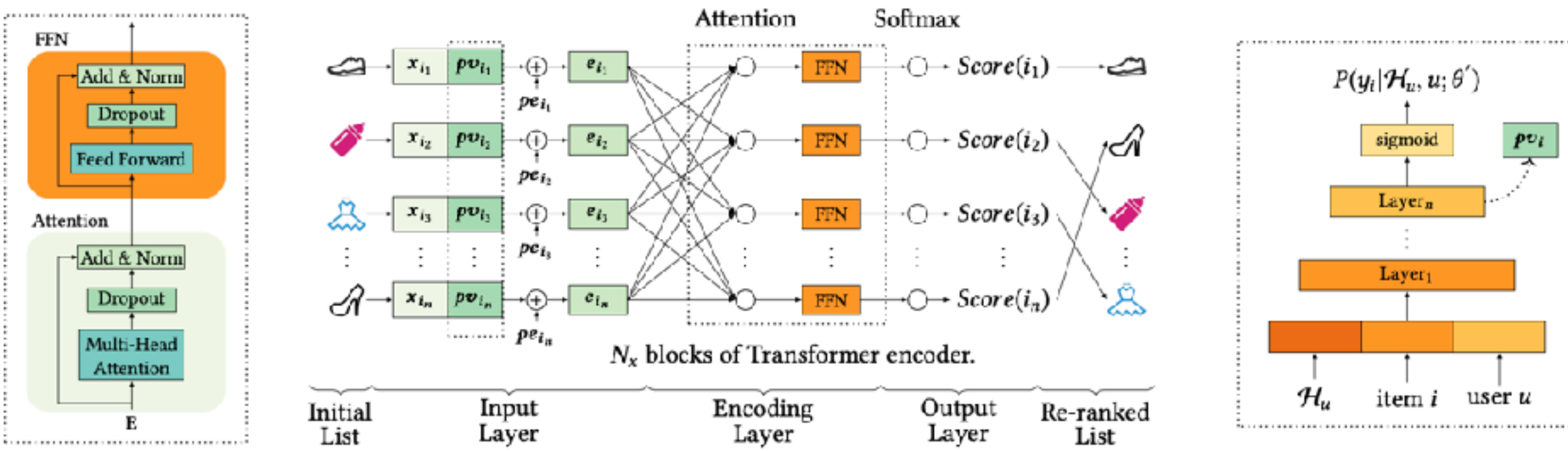
Lambda Mart

- LambdaMART combines LambdaRank and MART (Multiple Additive Regression Trees)
- MART uses gradient boosted decision trees for prediction tasks, LambdaMART uses gradient boosted decision trees using a cost function derived from LambdaRank for solving a ranking task

https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

Code and extra documention:

https://mlexplained.com/2019/05/27/learning-to-rank-explained-with-code/



(a) One block of Transformer encoder.

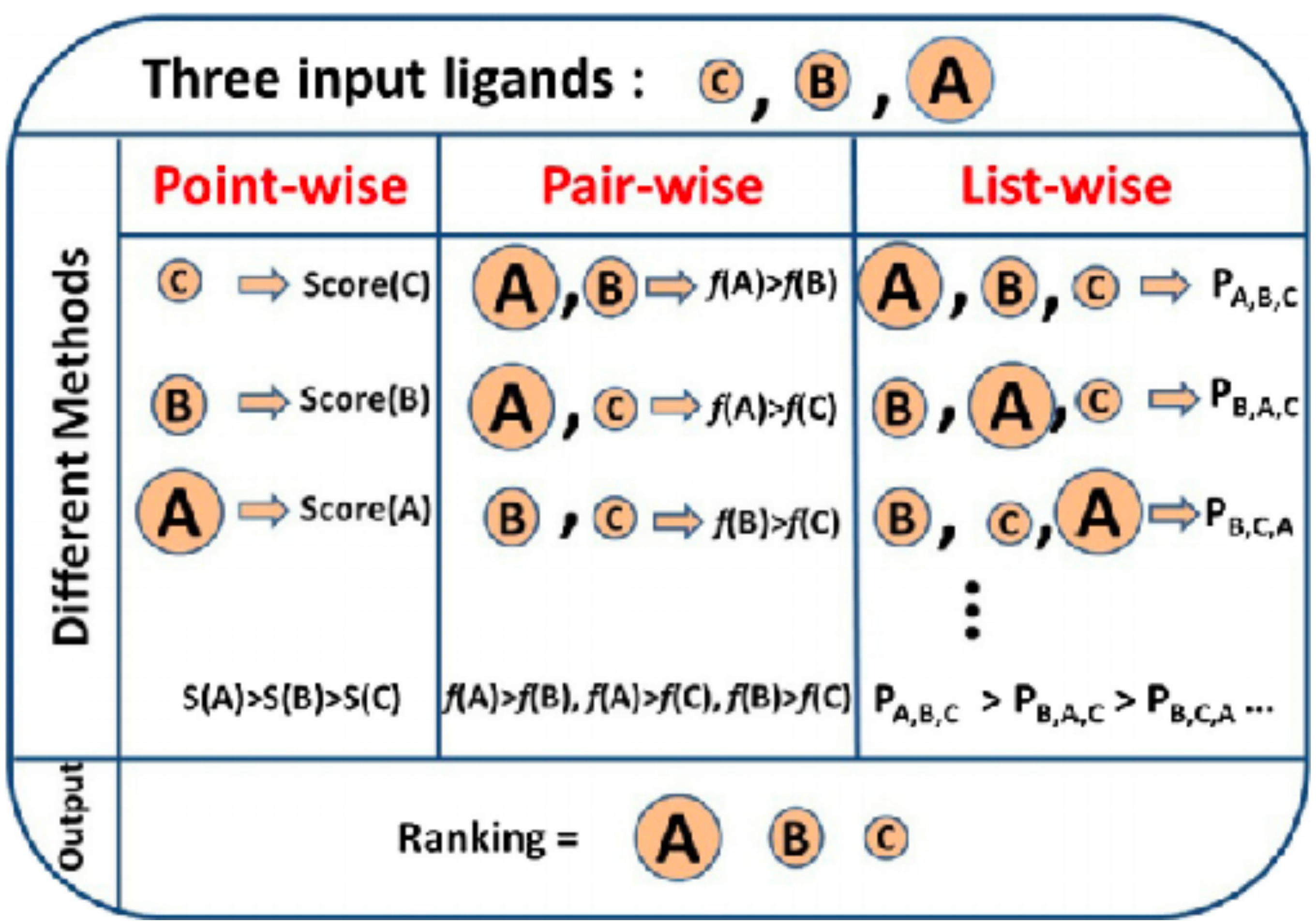
(b) Architecture of PRM.

(c) The pre-trained model to generate $\boldsymbol{pv_i}$, $i = i_1, ..., i_n$.

Figure 1: The detailed network structure of our PRM (Personalized Re-ranking Model) and its sub-modules.

Learning to rank

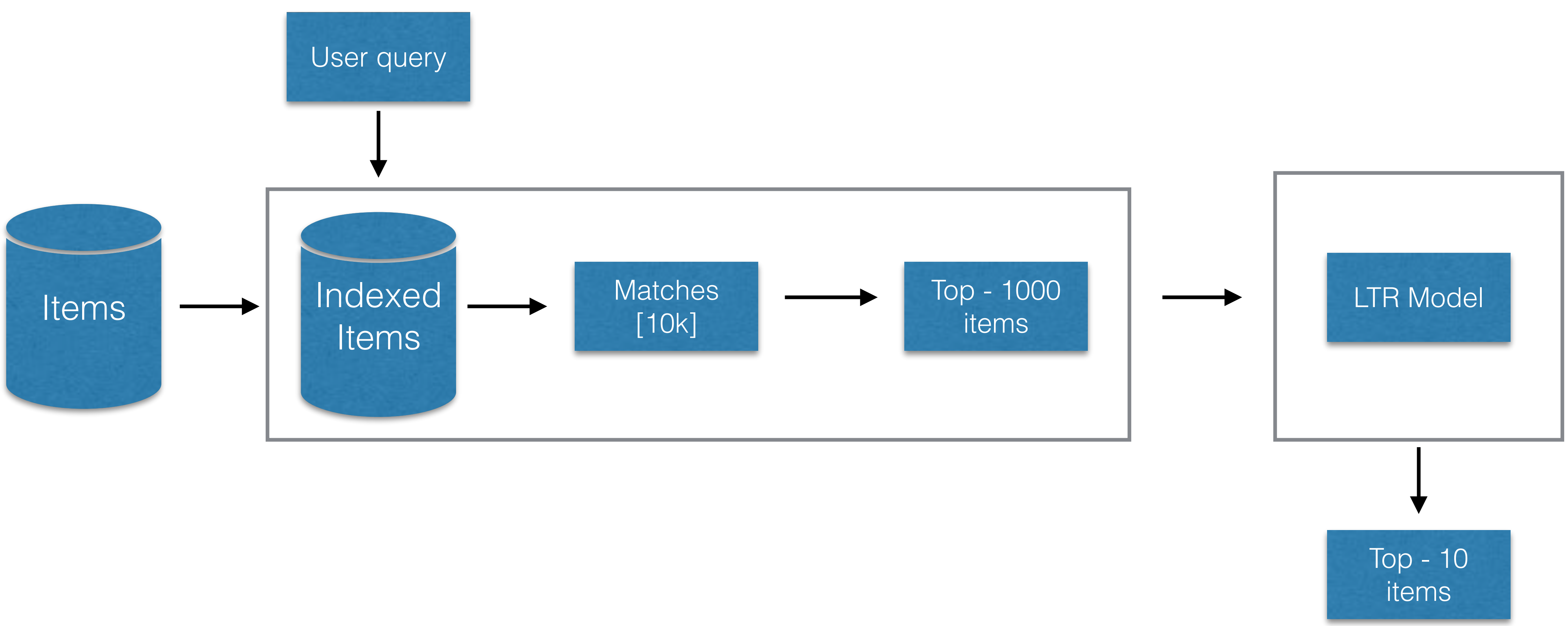
- Using Machine Learning, the goal is to **construct a ranking model** from the training data.
- The problem can be treat as a standard classification problem



LTR are great but task comsuming….

LTR framework

- LTR methods are computationally expensive.
- Usually used as re-rankers



If you have implicit data

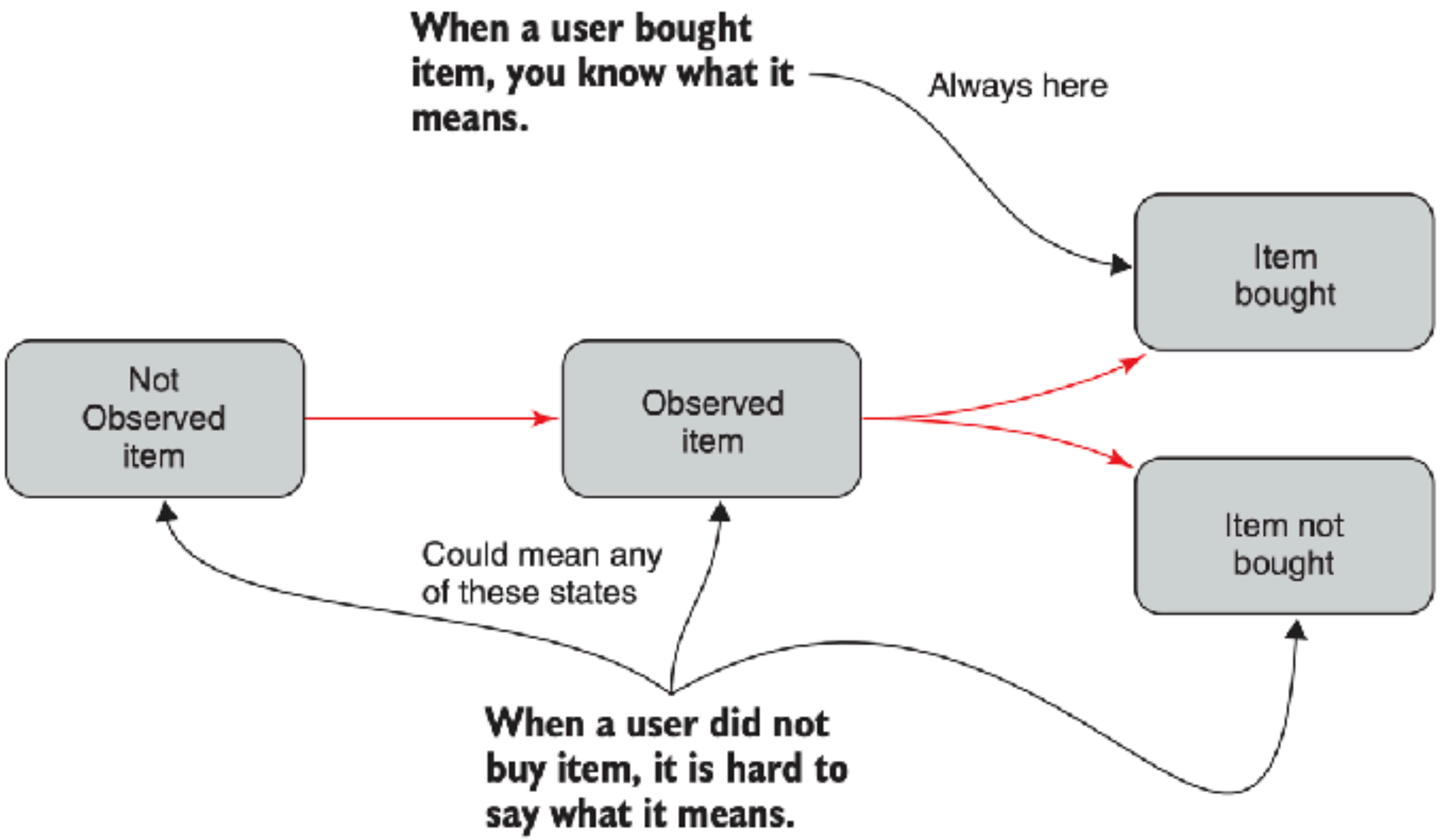
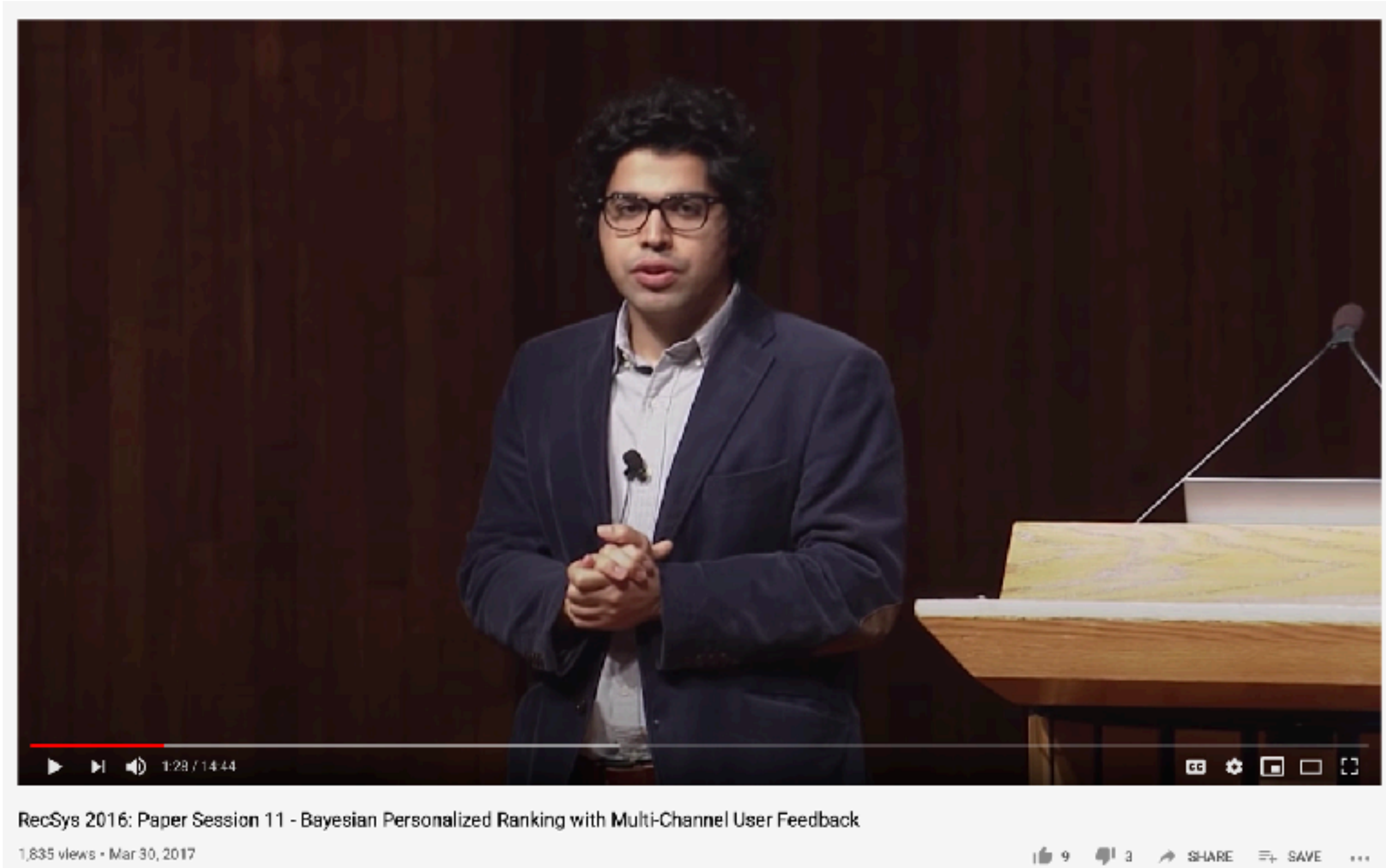


Figure 13.6 Different states of a user-item relationship. You know that when a user buys an item it's bought, but if a user doesn't buy an item, what does that mean?



https://www.youtube.com/watch?v=aKHLf4P3N08