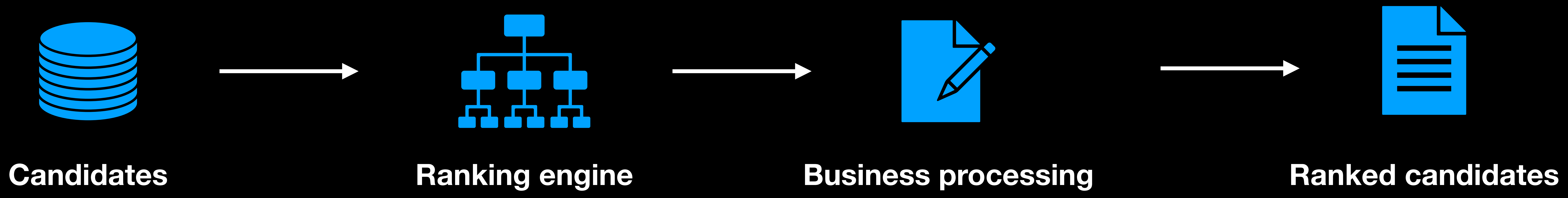
Recommendation Systems

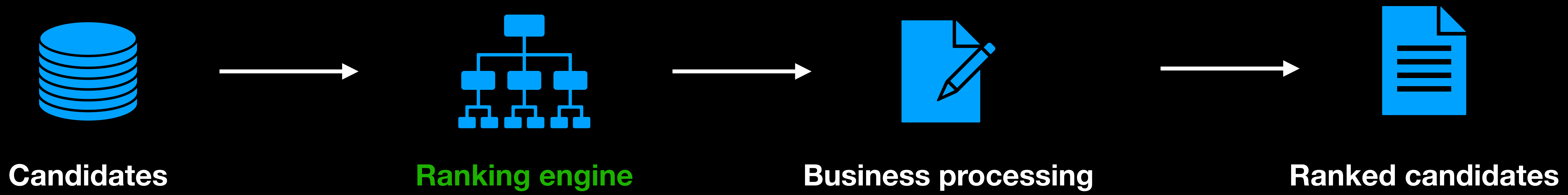
Learning to **rank**

Eugeny Malyutin / Sergey Dudorov

RecSystem structure



RecSystem structure



Ok, ML

How to define it?

- …
- $x \in X = \{x_1, \ldots x_n\}$ — items, $y \in Y = \{y_1, \ldots, y_n\}$ — «labels»
- There are X_n, Y_n — given dataset with «answers», we believe that there is implicit dependency $y^* : X \to Y$
- Need to define algorithm $\alpha : X \to Y$.
- If $Y \in \{0,1\}$ — this task is called binary classification
  If $Y \in \{y_0, y_1 \ldots y_n\}$ — this task is called multi class classification
  If $Y = R$ — this task is called regression

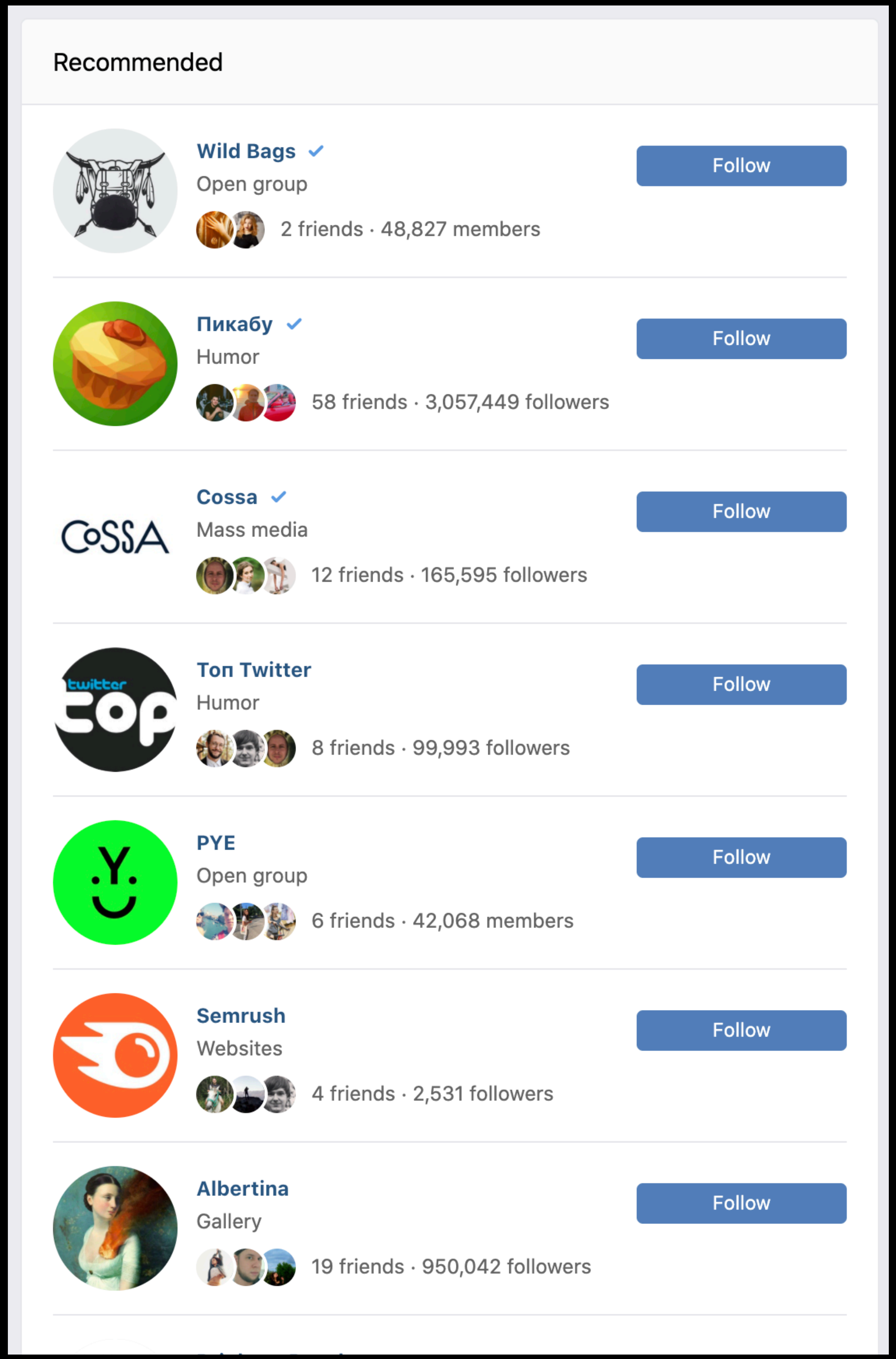
Ok, ML

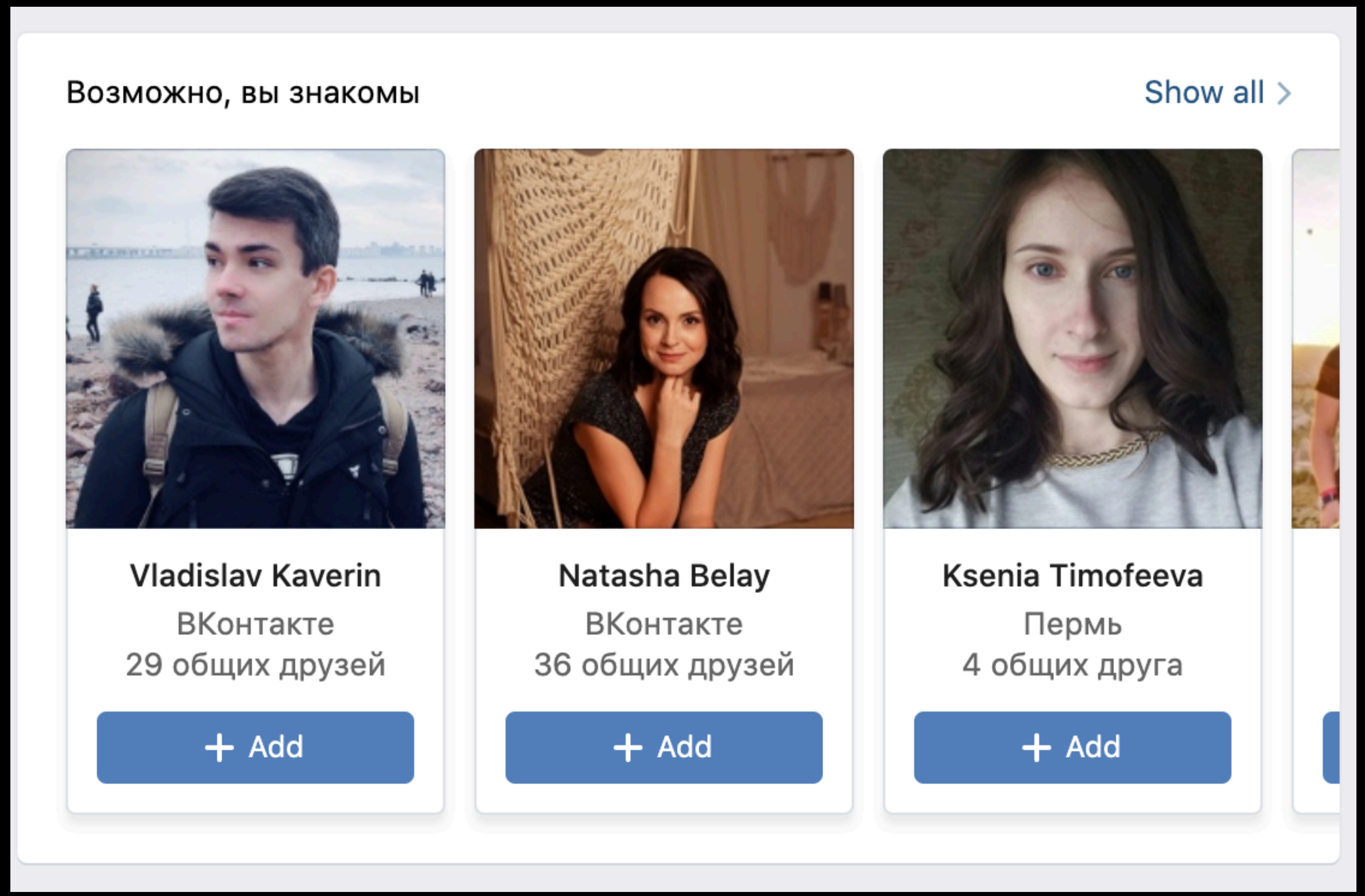
How to define it?

- …
- $x \in X = \{x_1, \ldots x_n\}$ — items, $y \in Y = \{y_1, \ldots, y_n\}$ — «labels»
- There are X_n, Y_n — given dataset with «answers», we believe that there is implicit dependency $y^* : X \to Y$
- Need to define algorithm $\alpha : X \to Y$.
- If $Y \in \{0,1\}$ — this task is called binary classification
  If $Y \in \{y_0, y_1 \ldots y_n\}$ — this task is called multi class classification
  If $Y = R$ — this task is called regression

Ok, binary classification.

Does our cases fits binary classification?





Learning to rank

Definition

- X — set of objects. Exists: $x_i \prec x_j$ — **order** relation.

  Our task to implement ranking function: $x_i \prec x_j \rightarrow a(x_i) < a(x_j)$

- Most of times order exists alongside with groupId (query relevant documents search engine) if $y(q, d) = \{0,1\}$ — binary relevance, $d_q^{(i)}$ - i-th doc by a(x_i) desc

- Ok, what's do you need more?

- Metrics

- Algorithms

- Tricks

Learning to rank

Metrics

- **Precision:** $P_n(q) = 1/n \sum_{i=1}^{n} y(q, d_q^{(i)})$ (percentage of correct in first n).
- Problems?
- **Average Precision:** $AP(q) = \sum y(d, d_q^{(n)}) P_n(q))/ \sum y(q, d_n^q)$
  (average P_n by all relevant docs position)
- Problems?
- **Mean average precision:** $MAP = 1/|Q| \sum_q AP(x)$

Learning to rank

Metrics

- **MRR:** $MRR = 1/|Q|\sum_{i=1}^{|Q|} 1/rank_i$:
  - Sum over relevant!
  - Easy to interpret
- **DCG** (discounted cumulative gain): $DCG_n(q) = \sum_{i}^{n} G_q(d_q^i)D(i)$

  $G_q(d_q^i) = (y(d, q))$ — gain, 1/0 for relevant/unrelevant doc's

  $D(i) = 1/log2(i+1)$ —discounts, higher relevant docs are — better
- **NDCG:** $NDCG(q) = DCG_n(q)/maxDCG_n(q)$ — normalized version

Learning to rank

Metrics

- **DCG** (discounted cumulative gain): $DCG_n(q) = \sum_{i}^{n} G_q(d_q^i)D(i)$

  $G_q(d_q^i) = (y(d, q))$ — gain, 1/0 for relevant/unrelevant doc's

  $D(i) = 1/log2(i + 1)$ —discounts, higher relevant docs are — better

- **NDCG:** $NDCG(q) = DCG_n(q)/maxDCG_n(q)$ — normalized version

- Also NDCG exists w.r.t. groupId, possible to use averaged over groups.

- Strongly correlates with AUC-per-groupId, but works with ranking objectives (unbounded)

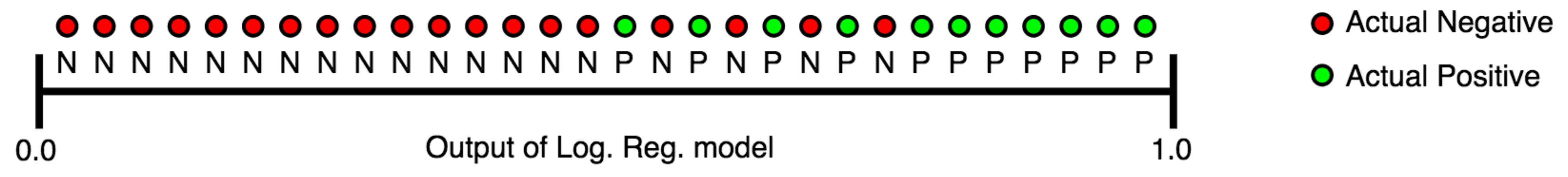


Figure 6. Predictions ranked in ascending order of logistic regression score.

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.

Learning to rank

NDCG by example

NDCG - Example

4 documents: d_1, d_2, d_3, d_4

i	Ground Truth		Ranking Function$_1$		Ranking Function$_2$	
	Document Order	r_i	Document Order	r_i	Document Order	r_i
1	d4	2	d3	2	d3	2
2	d3	2	d4	2	d2	1
3	d2	1	d2	1	d4	2
4	d1	0	d1	0	d1	0
	$NDCG_{GT}$=1.00		$NDCG_{RF1}$=1.00		$NDCG_{RF2}$=0.9203	

$$DCG_{GT} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.6309$$

$$DCG_{RF1} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.6309$$

$$DCG_{RF2} = 2 + \left(\frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.2619$$

$$MaxDCG = DCG_{GT} = 4.6309$$

Learning to rank

Metrics

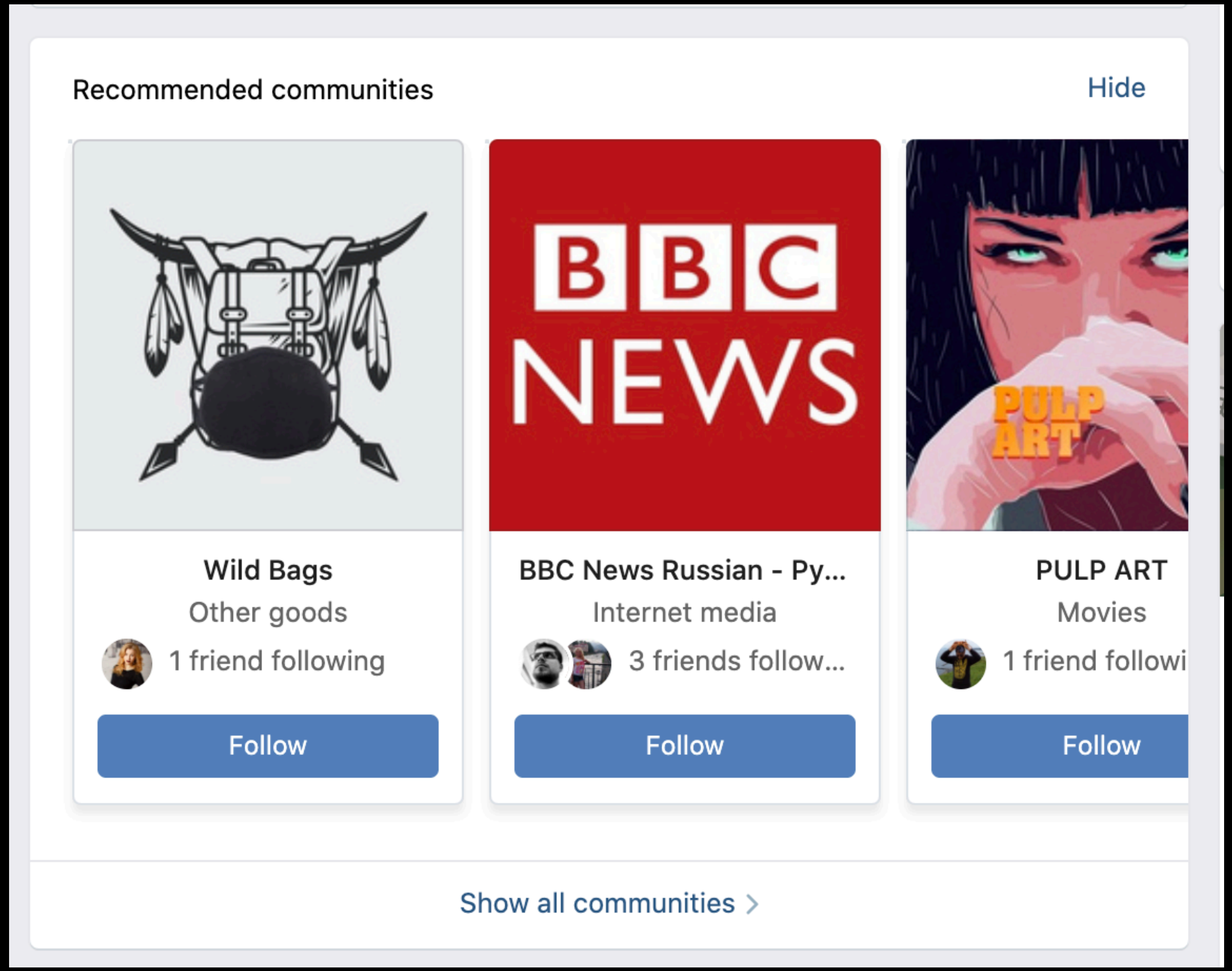
- **DCG** (discounted cumulative gain): $DCG_n(q) = \sum_i^n G_q(d_q^i)D(i)$

  $G_q(d_q^i) = (y(d,q))$ — gain, 1/0 for relevant/unrelevant doc's

  $D(i) = 1/log2(i+1)$ —discounts, higher relevant docs are — better

- **NDCG:** $NDCG(q) = DCG_n(q)/maxDCG_n(q)$ — normalized version

- Also popular **NDCG-at-k** (k=3, for example)



Learning to rank

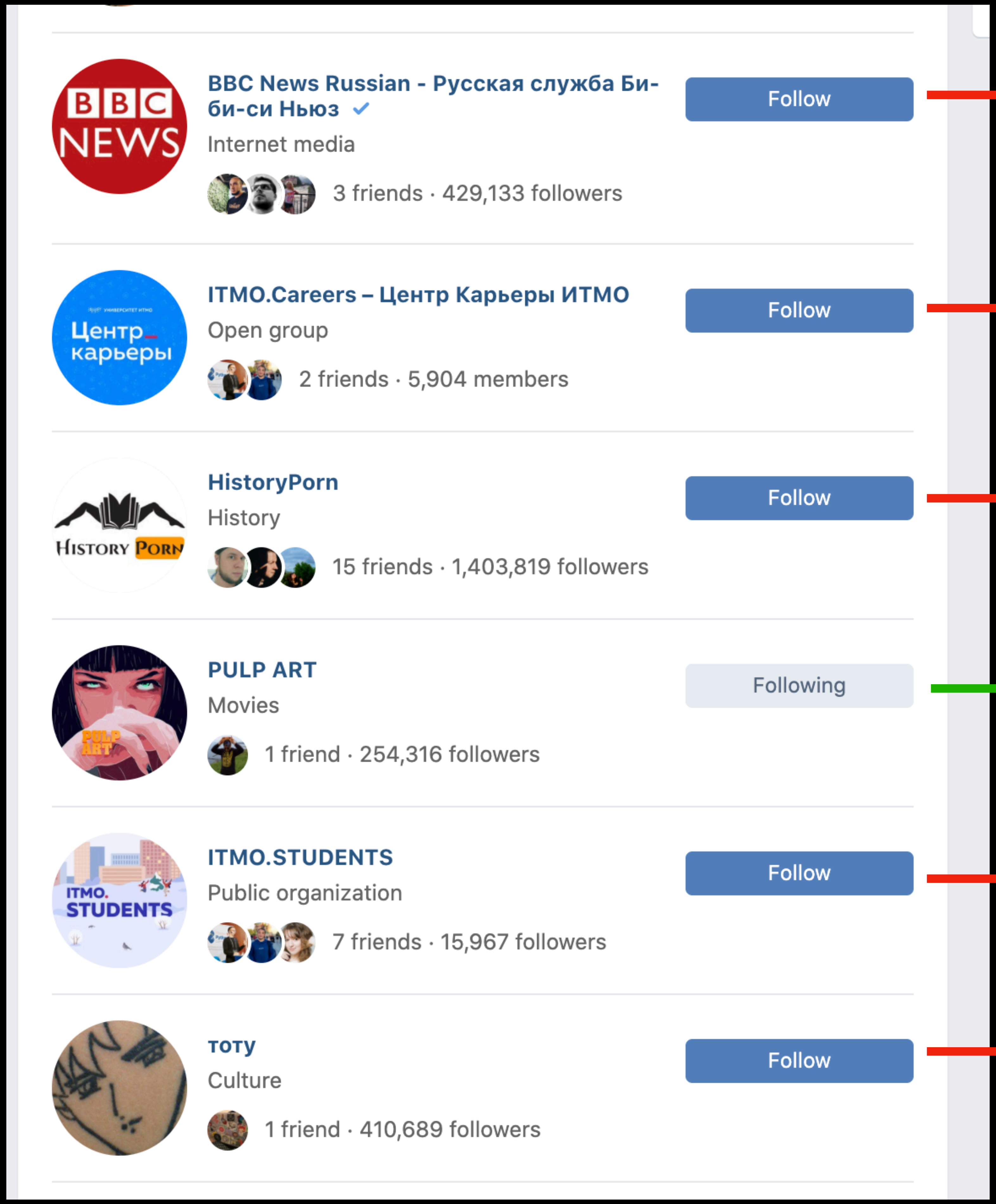
Classification ontology

- **Pointwise:** sorting by score $\alpha(x_i)$ estimating «relevance»
- **Pairwise:** compares objects $a(x_i, x_j) :> 0 \ if \ x_i \prec x_j$ — decide which one is more relevant
- **Listwise:** Takes **the entire list** of candidates and optimise it's order

Learning to rank

Classification ontology

- **Pointwise:** sorting by score $\alpha(x_i)$ estimating «relevance»



Negative

Negative

Negative

Positive

Negative

Negative

- **Positives** — $y_3 = 1$
- **Negatives** — $y_{0,1,2,4,5} = 0$
- $\alpha(x)$ gives relevance score $\in [0,1]$

Learning to rank

Classification ontology

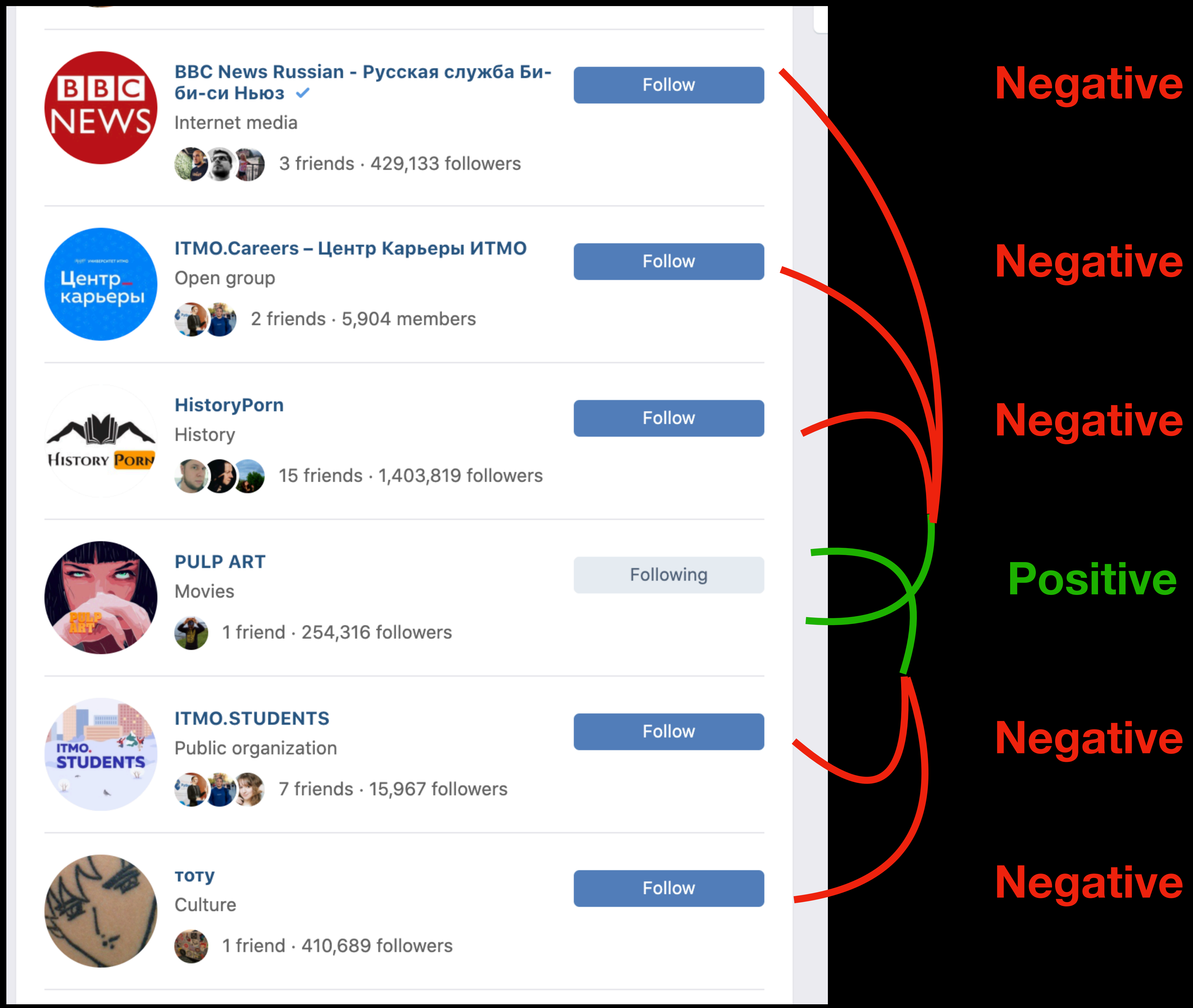
- **Pairwise:** decide which one is more relevant

- **Train set:** $\{x_3 \prec x_0, x_3 \prec x_1, \ldots, x_3 \prec x_5\}$
- $\alpha(x_i, x_j)$ compares pair of objects
- OK, what's a problem?

Learning to rank

Classification ontology

- **Pairwise:** decide which one is more relevant

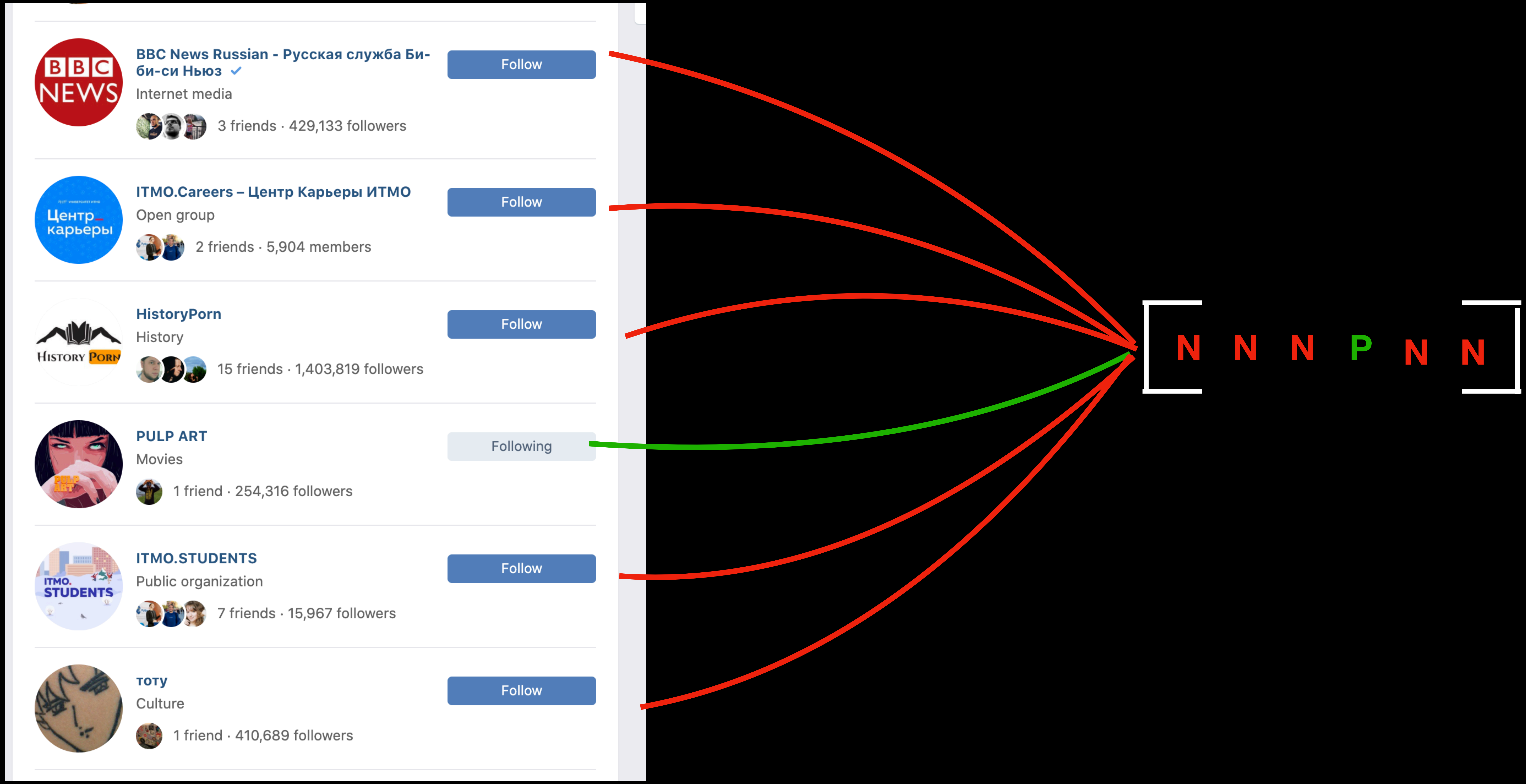


- **Train set:** $\{x_3 \prec x_0, x_3 \prec x_1, \ldots, x_3 \prec x_5\}$
- $\alpha(x_i, x_j)$ compares pair of objects
- OK, what's a problem?
- How much α applications do you need to sort 10000 - elemnts list?

Learning to rank

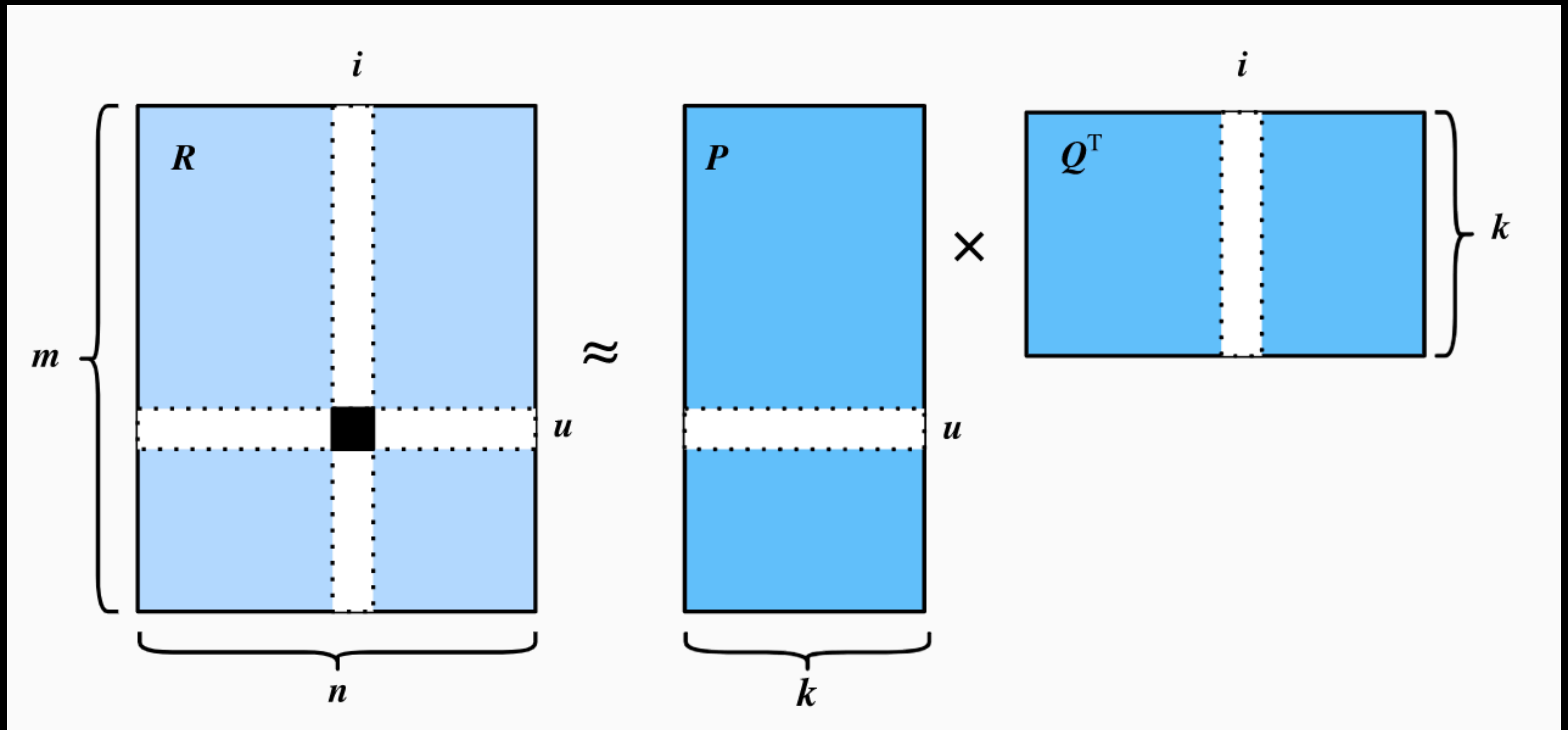
Classification ontology

- **Listwise:** optimize the whole list



Learning to rank

SVD-factorisation



$$argmin_{P,Q,b} \sum_{u,i \in K} ||R_{ui} - \hat{R}_{ui}||^2 + \lambda(||P||^2 + ||Q||^2 + b_u^2 + b_i^2)$$

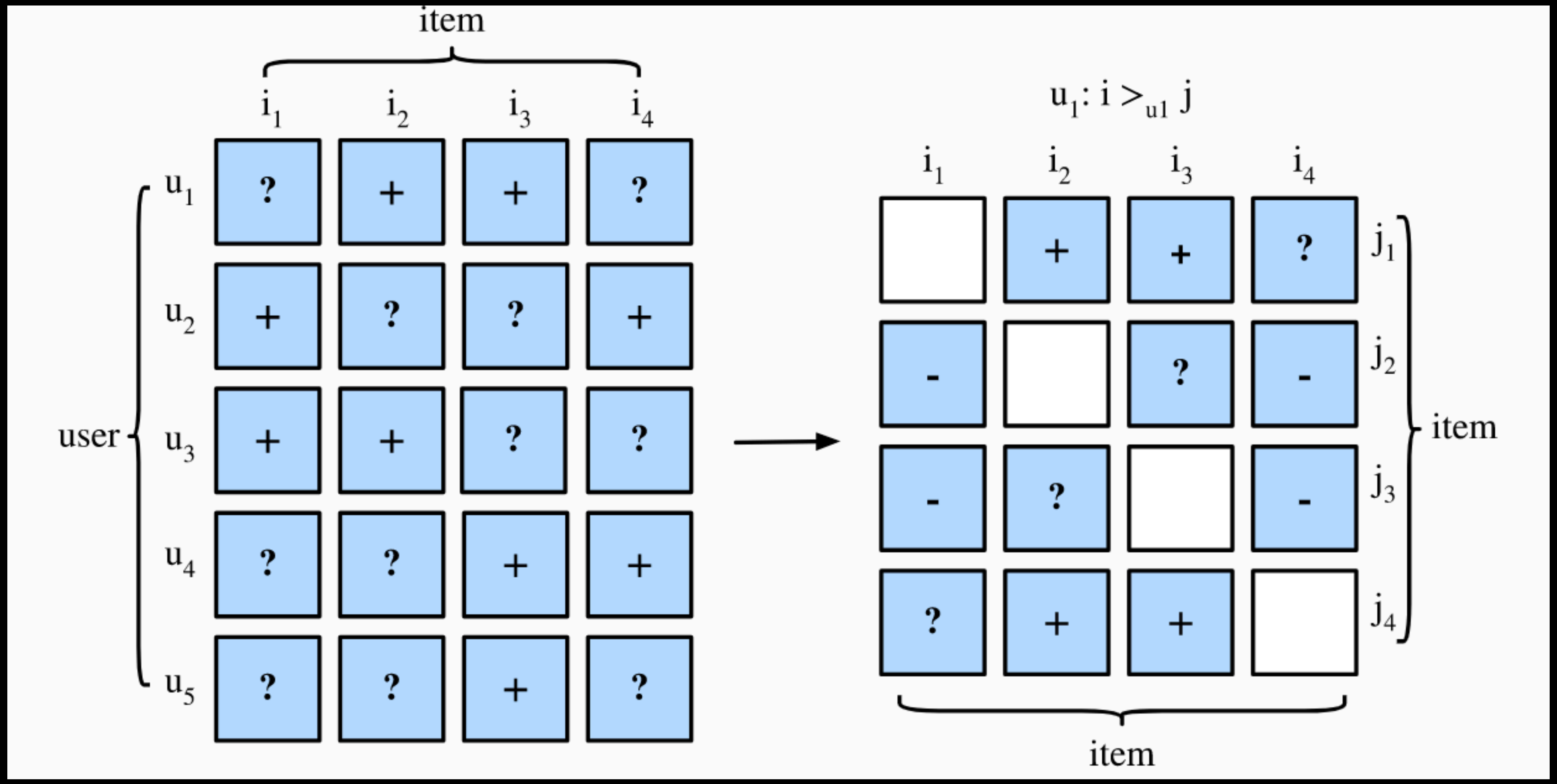
$$argmin_{p,q,b} \sum_{u,i \in K} (< p_u, q_j > + b_u + b_i + \mu - x_{ij})^2 + \alpha \sum_i ||u_i||^2 + \beta \sum_j ||p_j||^2 + \gamma ||b_u|| + \theta ||b_i||$$

$\alpha(u, i) = < p_u, q_i > + b_u + b_i + \mu$ ⟶ **Pointwise** algorithm(!)

https://d2l.ai/chapter_recommender-systems/mf.html

Learning to rank

BPR-factorisation



$$p(\Theta \mid >_u) \propto p(>_u \mid \Theta)p(\Theta)$$

$$\downarrow$$

$$\begin{aligned}\text{BPR-OPT} := & \ln p(\Theta \mid >_u) \\ & \propto \ln p(>_u \mid \Theta)p(\Theta) \\ & = \ln \prod_{(u,i,j \in D)} \sigma(\hat{y}_{ui} - \hat{y}_{uj})p(\Theta) \\ & = \sum_{(u,i,j \in D)} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \ln p(\Theta) \\ & = \sum_{(u,i,j \in D)} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) - \lambda_\Theta ||\Theta||^2\end{aligned}$$

$$r_{ui} = < e_u, e_i > + b_u + b_i + \mu$$

$$\alpha(x_i, x_j, u) = \ln(\sigma(< e_u, e_i > + b_u + b_i - < e_u, e_j > - b_u - b_j))$$

$$\downarrow$$

$$\alpha(u, i) = < e_u, e_i >$$

Sort with α

PAIRwise algorithm while learn
POINTwise algorithm on runtime!

https://d2l.ai/chapter_recommender-systems/mf.html

Learning to rank

Summary

- Ranking **differs** either classification or regression due to strict **order properties** on **relevance**
- **It** leads us to new metrics: MAP, MRR, AUC(~), NDCG
- Algorithms ontology: **point**-, **pair**-, **list**-wise
- There are **no** working «fair» **listwise** algorithms **IRL** (at least author saw no one)
- **Best pairwise** algorithms — camouflages themselves as **pointwise**.