Session EAP: Engineering Applications

Learning Non-linear Ranking Functions for Web Search using Probabilistic Model Building GP

Hiroyuki Sato,

Danushka Bollegala,

Yoshihiko Hasegawa,

and Hitoshi Iba

THE UNIVERSITY OF TOKYO

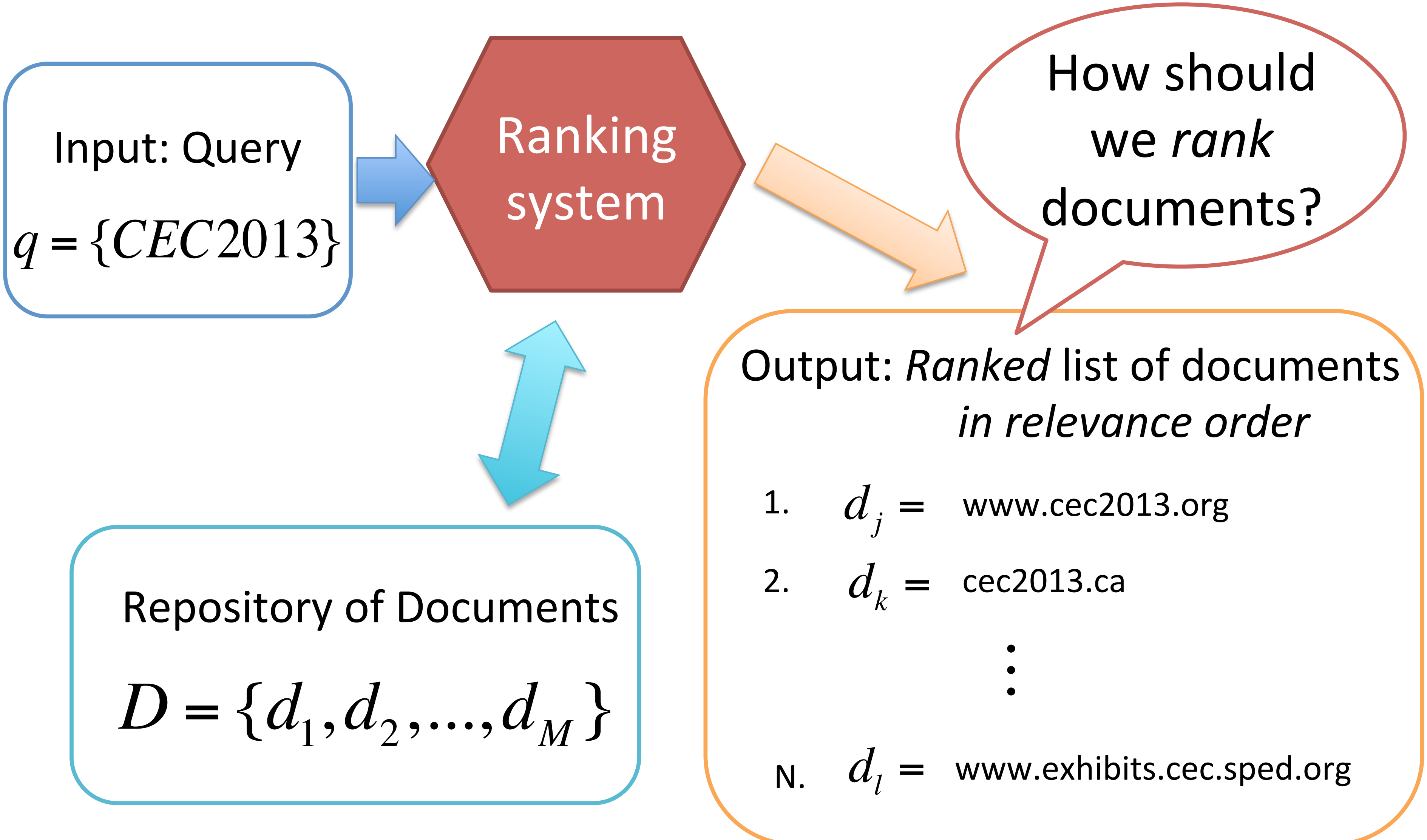
Outline

- Introduction
- Learning to Rank
- Probabilistic Model Building GP
- The Proposed method: **Rank-PMBGP**
- Experiments and discussion
- Conclusion

Search engines

The most efficient way to search documents from Web

The anatomy of a search engine

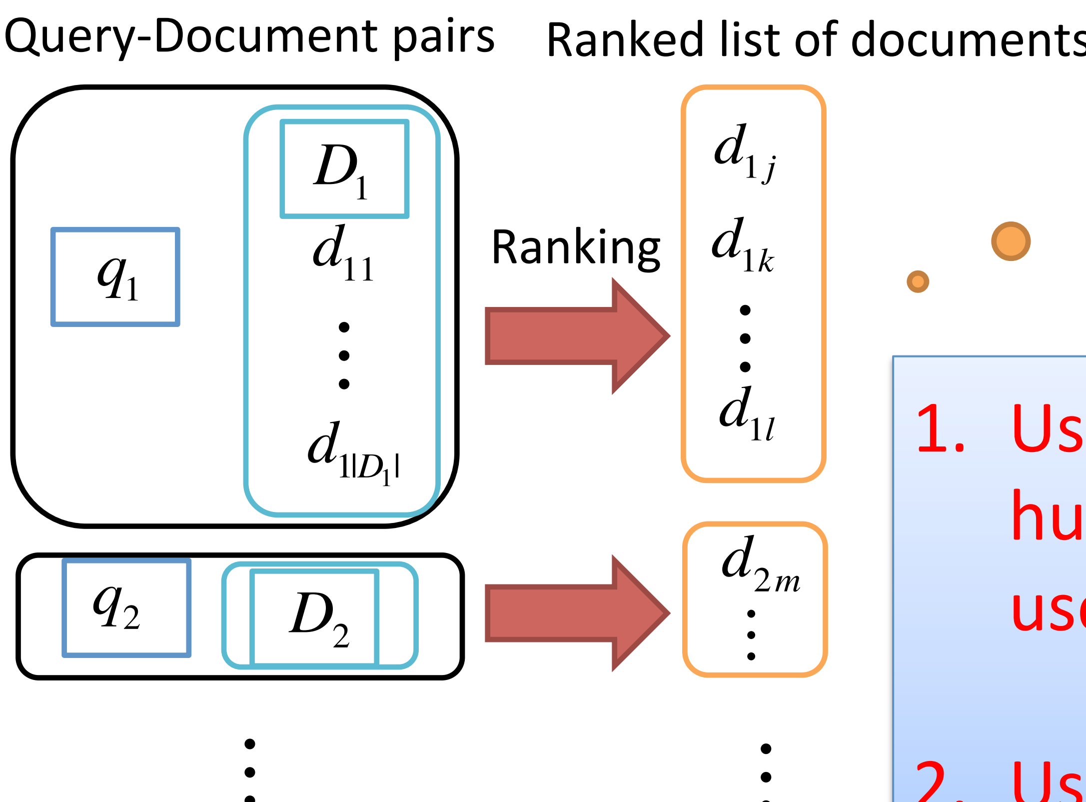


Outline

- Introduction
- Learning to Rank
- Probabilistic Model Building GP
- The Proposed method: **Rank-PMBGP**
- Experiments and discussion
- Conclusion

Problem Settings

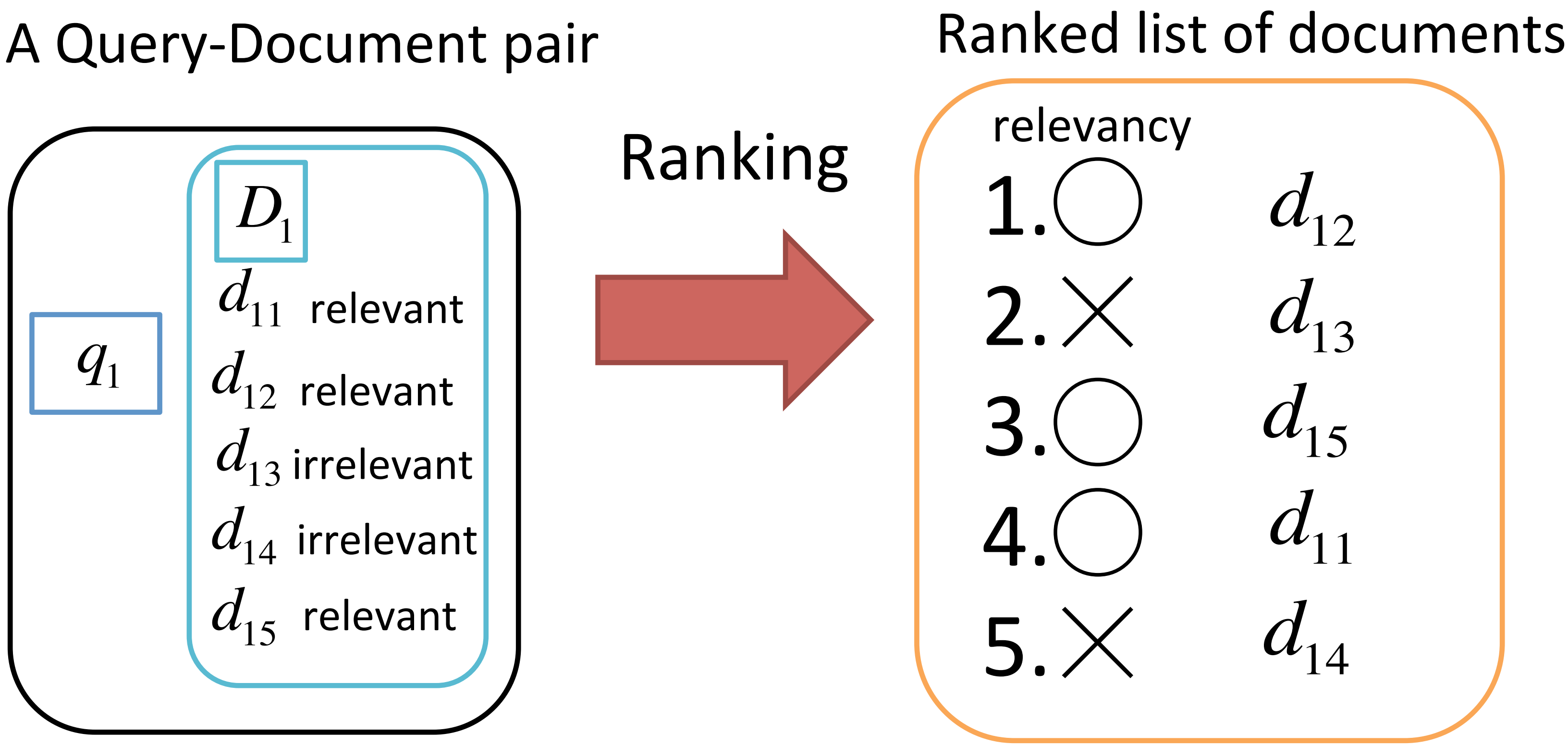
- When Query-Document pairs are given, we want Ranking System which outputs *proper* ranked list of documents



1. Use dataset tagged by human (gotten from real users)
   Ex. d_{1j} is relevant to q_1
   d_{1k} is not relevant to q_1
2. Use evaluate measures

P@n (Precision at position n)

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$



P@n (Precision at position n)

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$

Ranked list of documents

relevancy

1. ○ d_{12}
2. × d_{13}
3. ○ d_{15}
4. ○ d_{11}
5. × d_{14}

P@1 = 1 / 1 =1

P@n (Precision at position n)

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$

Ranked list of documents

relevancy

1. ○ d_{12}
2. × d_{13}
3. ○ d_{15}
4. ○ d_{11}
5. × d_{14}

P@1 = 1 / 1 = 1

P@2 = 1 / 2 = 0.5

P@n (Precision at position n)

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$

Ranked list of documents

relevancy

1. ○ d_{12}
2. × d_{13}
3. ○ d_{15}
4. ○ d_{11}
5. × d_{14}

P@1 = 1 / 1 = 1

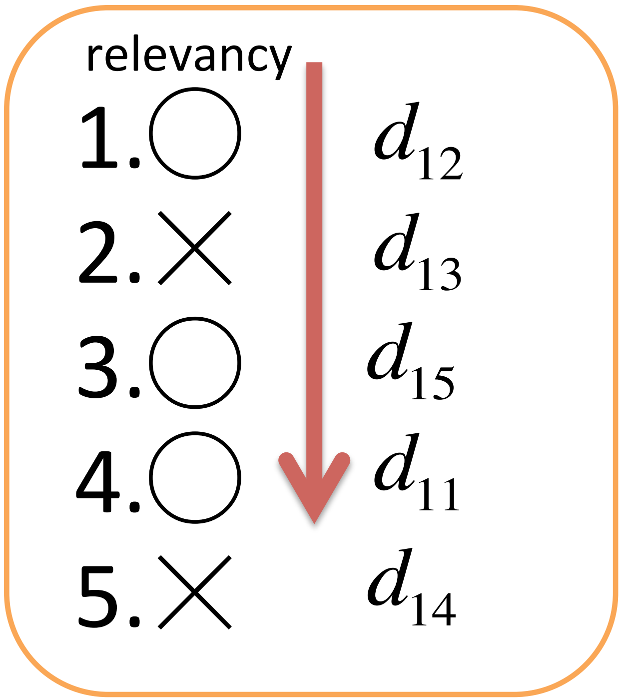
P@2 = 1 / 2 = 0.5

P@3 = 2 / 3 = 0.67

P@n (Precision at position n)

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$

Ranked list of documents



P@1 = 1 / 1 = 1

P@2 = 1 / 2 = 0.5

P@3 = 2 / 3 = 0.67

P@4 = 3 / 4 = 0.75

P@n (Precision at position n)

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$

Ranked list of documents

relevancy

1. ○ d_{12}
2. × d_{13}
3. ○ d_{15}
4. ○ d_{11}
5. × d_{14}

P@1 = 1 / 1 = 1

P@2 = 1 / 2 = 0.5

P@3 = 2 / 3 = 0.67

P@4 = 3 / 4 = 0.75

P@5 = 3 / 5 = 0.6

AP (Average Precision)

※ Usually, N = 10

Ranked list of documents

$$AP = \frac{\sum_{n=1}^{N}(P@n \times rel(n))}{\text{No. of relevant docs for this query}}$$

relevancy

1. ○ d_{12}
2. × d_{13}
3. ○ d_{15}
4. ○ d_{11}
5. × d_{14}

P@1 = 1

P@2 = 0.5

P@3 = 0.67

P@4 = 0.75

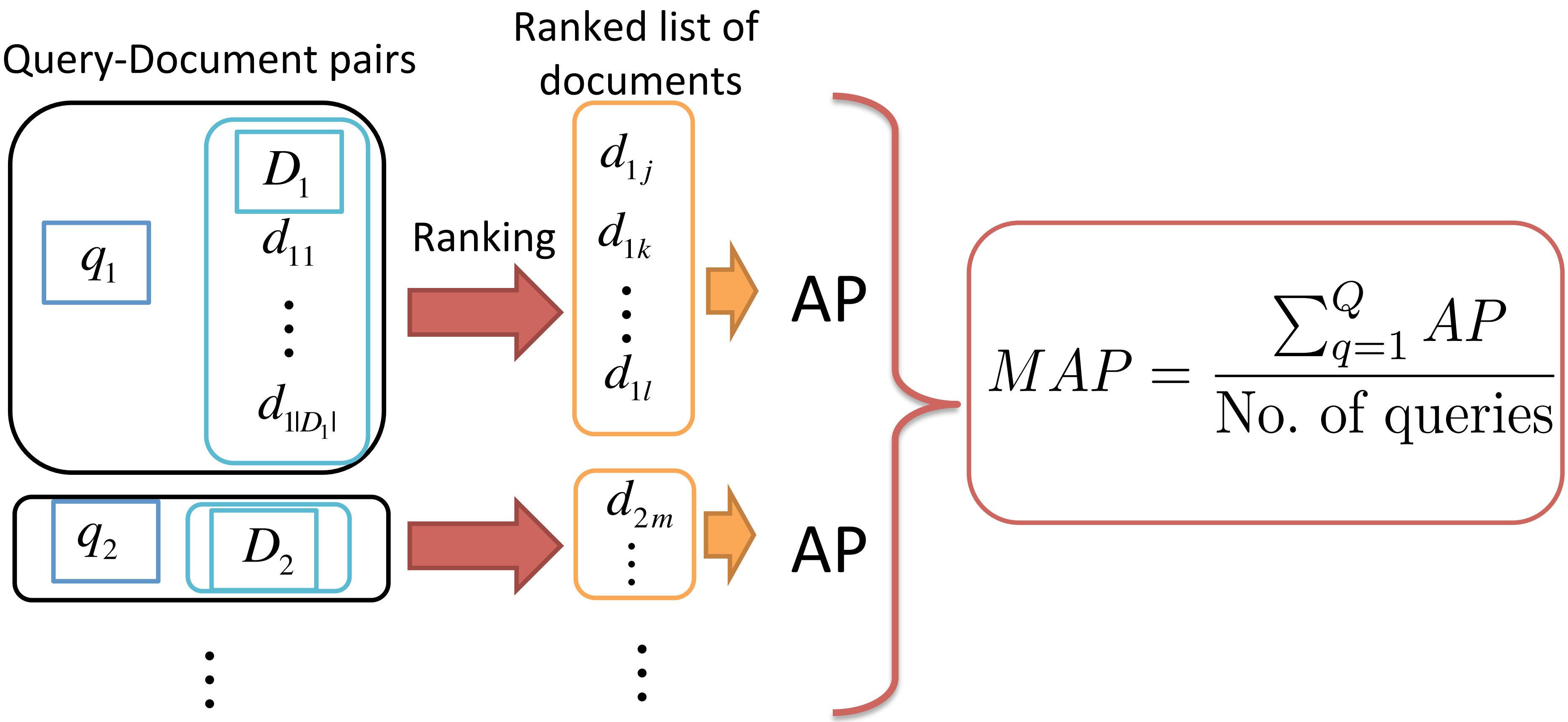
P@5 = 0.6

AP = (P@1+P@2+P@3+P@4+P@5) / 5

= 0.70

MAP (Mean Average Precision)

- Popular evaluation method for ranking, but time consuming
- Employed as fitness function in the proposed method



Features that search engines must consider

- Relevancy between query and document: depends on <u>both query and document</u>
  - term frequency (tf)
  - inverse document frequency(idf)
  - tf-idf
  - BM25: normalized tf-idf by document length
- Importance of documents: depends <u>only on document</u>
  - Page rank
  - HITS

Can a combination of these features define more accurate relevancy and importance?

Ranking function & Learning to Rank

- Ranking function
  - Combination of relevancy and importance features
  - Returns higher real values for more relevant query and document pairs
- *Linear* ranking function *was* commonly used

$$F(query, document) = \sum \omega_i f_i$$

  - Can be easily optimized
  - Fast for large queries
- Learning to Rank
  - To learn and optimize ranking function

Non-linear Ranking Function

- Generally
  - More degrees of freedom, possible to fit the actual ranking function better
- Experimental Results
  - Yahoo! Learning to Rank Challenge Overview [O. Chapelle et al. 2011]
    "The results of the challenge clearly showed that nonlinear models such as trees and ensemble learning methods are powerful techniques. "
  - Non-linear baseline, GBDT (Gradient Boosted Decision Tree) [J. Freedman 2002], beats many linear challengers

The attention to Non-linear learning to rank is ever increasing!
However, Non-linear search space is vast…

Outline

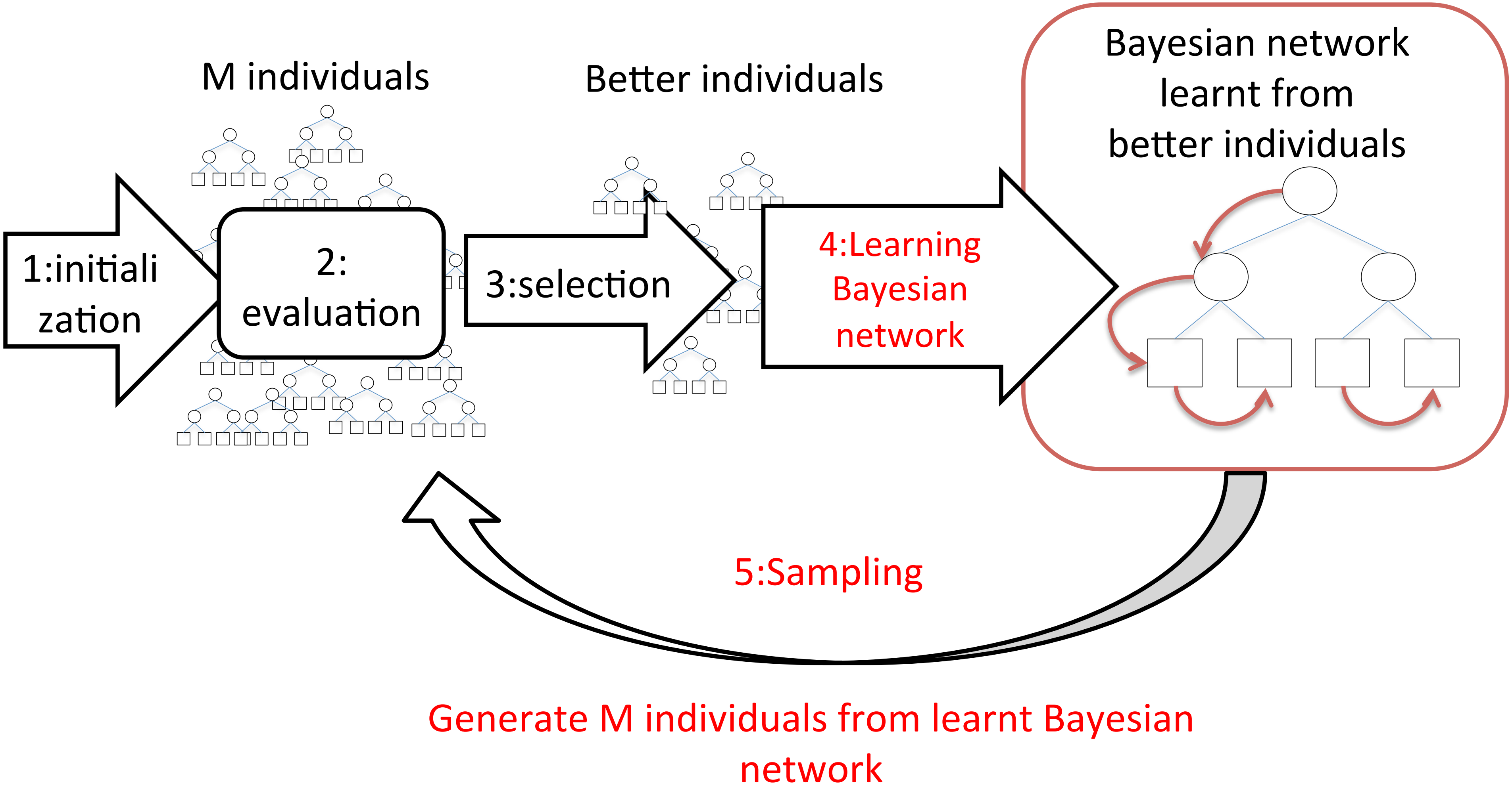
- Introduction
- Learning to Rank
- Probabilistic Model Building GP
- The Proposed method: **Rank-PMBGP**
- Experiments and discussion
- Conclusion

PMBGP (Probabilistic Model Building GP)

- Extension of EDAs (Estimation of Distribution Algorithms) to tree structures, functions or programs
- Estimate subtrees or other building blocks using Probabilistic models

In non-linear vast search space, it is considered efficient to estimate building blocks for searching good ranking functions

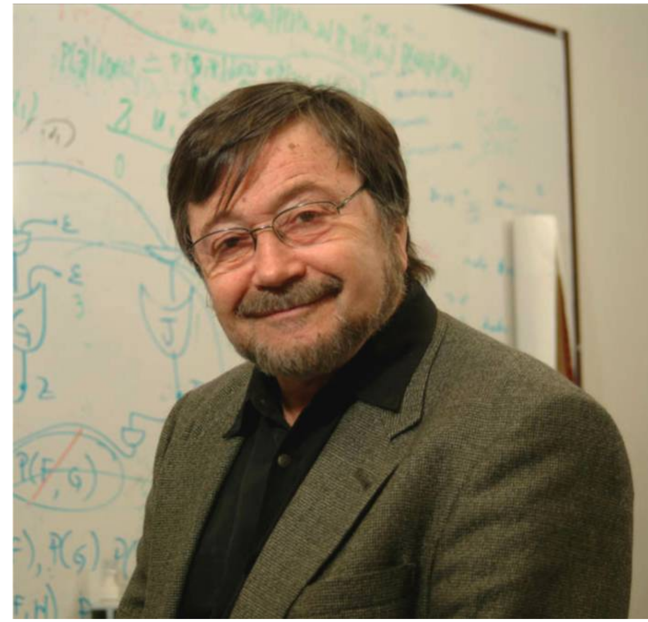
The Flow of Probabilistic Model Building GP (Using Bayesian network)



※ M: population size

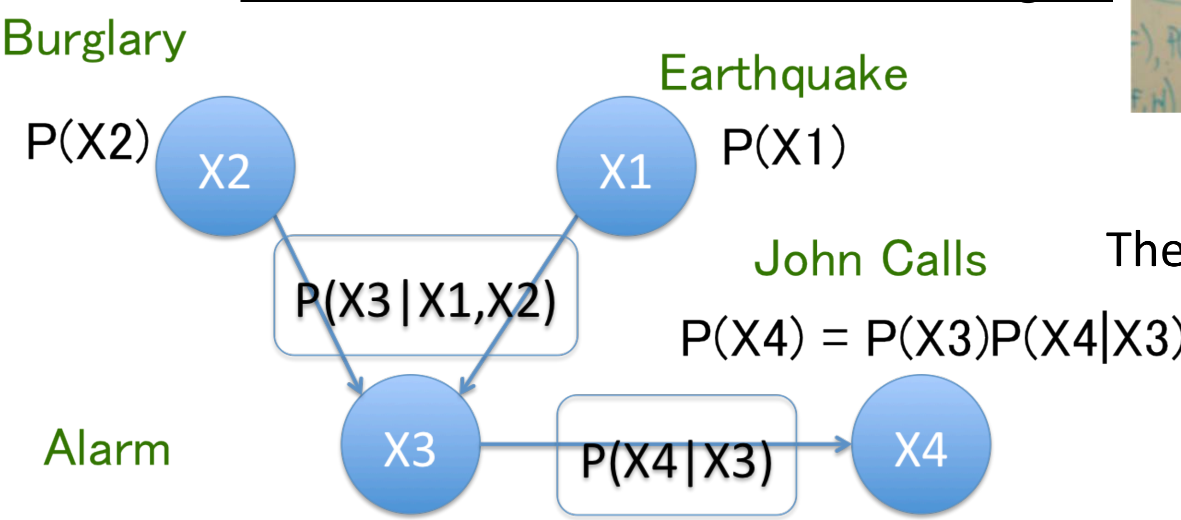
Bayesian network

- Probabilistic model to describe conditional dependencies
- Many applications
  - Disease detection
  - Machine trouble detection
  - Evolutionary Computation
    - EDA and Probabilistic Model Building GP



Judea Pearl

The 2011 winner of Turing Award

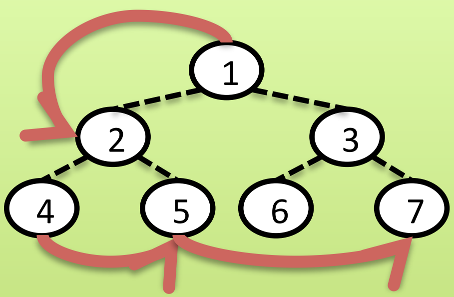


$P(X3) = P(X1)P(X2)P(X3|X1,X2)$

4 : Learning Bayesian network

Better individuals at generation g : B_g

Graph structure G



MAP (Maximum a posteriori) estimation

$$\hat{G} = \arg\max_{G}(P(G \mid B_g))$$
$$= \arg\max_{G}(P(B_g \mid G)P(G))$$

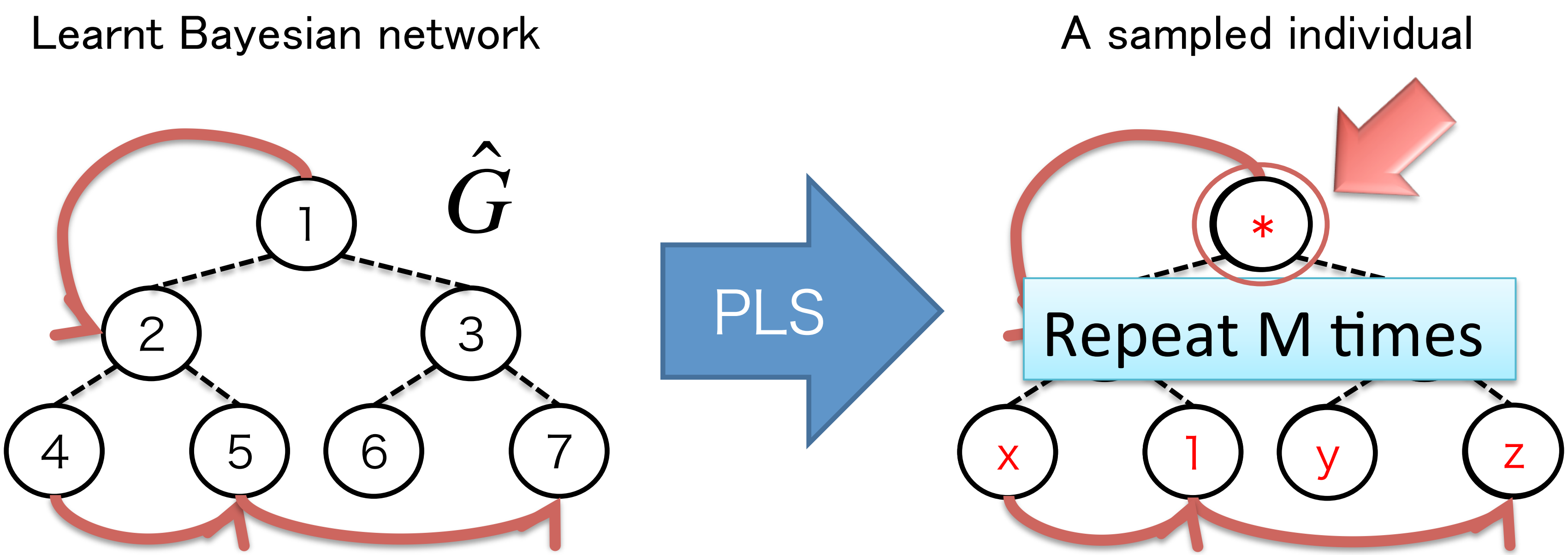
Greedy search for graph structure with maximize

$$P(B_g \mid G)P(G)$$

ex. BD score, BIC score

MAP (Maximum a posteriori) estimated Bayesian network $\hat{G}$

5：Sampling (generation of new individuals)
PLS: Probabilistic Logic Sampling



※ M: population size

Outline

- Introduction
- Learning to Rank
- Probabilistic Model Building GP
- The Proposed method: **Rank-PMBGP**
- Experiments and discussion
- Conclusion

The proposed method: **Rank-PMBGP**
Non-linear Learning to Rank using PMBGP

- Base: POLE (Program Optimization with Linkage Estimation) [Y. Hasegawa et al. 2007]
- Function nodes: {+, -, *}

Create non-linear elements

- Terminal nodes:
  - Variable nodes: features
  - Constant node: weights for features [0,1]
- Fitness: MAP

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$

$$MAP = \frac{\sum_{q=1}^{Q} AP}{\text{No. of queries}}$$

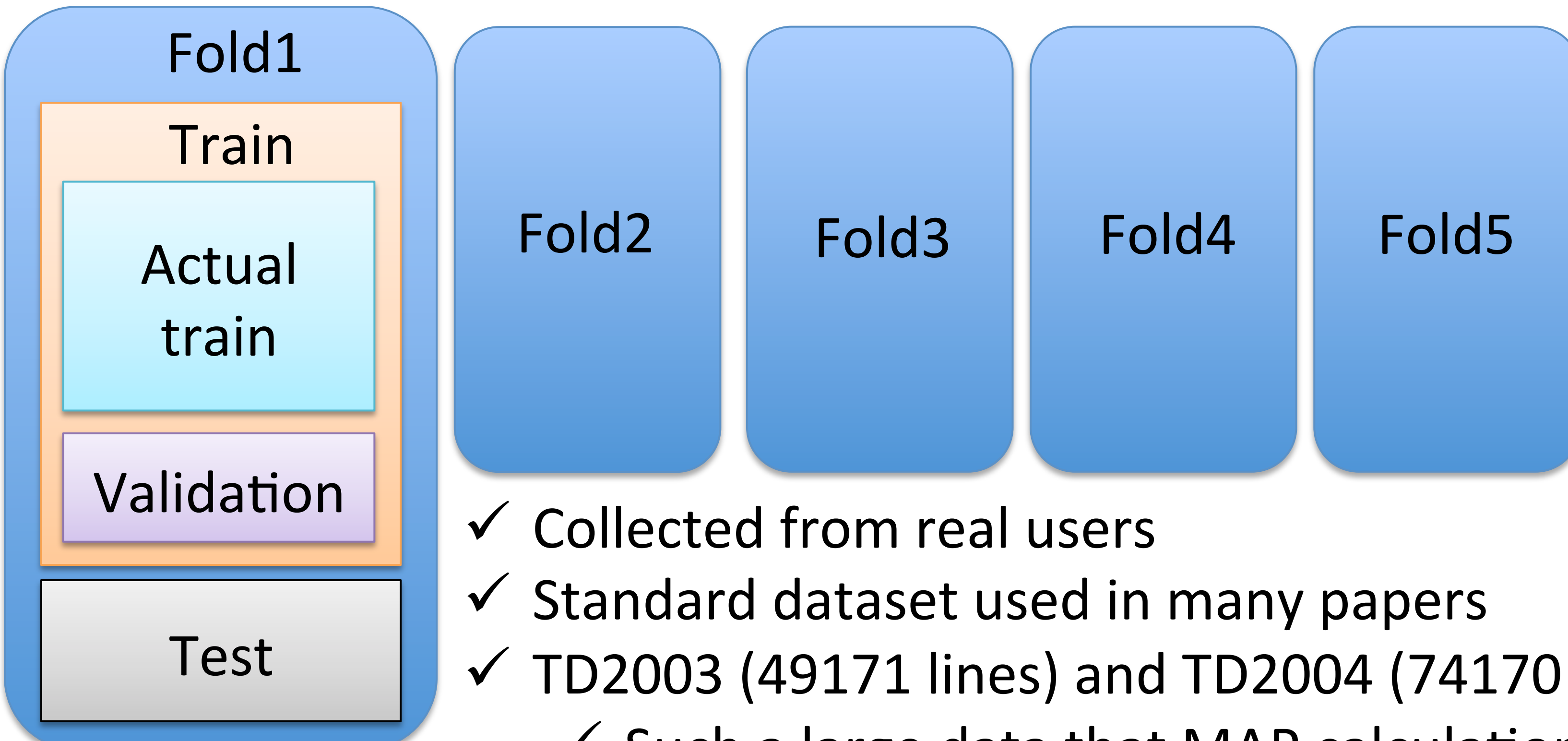
$$AP = \frac{\sum_{n=1}^{N}(P@n \times rel(n))}{\text{No. of relevant docs for this query}}$$

Outline

- Introduction
- Learning to Rank
- Probabilistic Model Building GP
- The Proposed method: **Rank-PMBGP**
- Experiments and discussion
- Conclusion

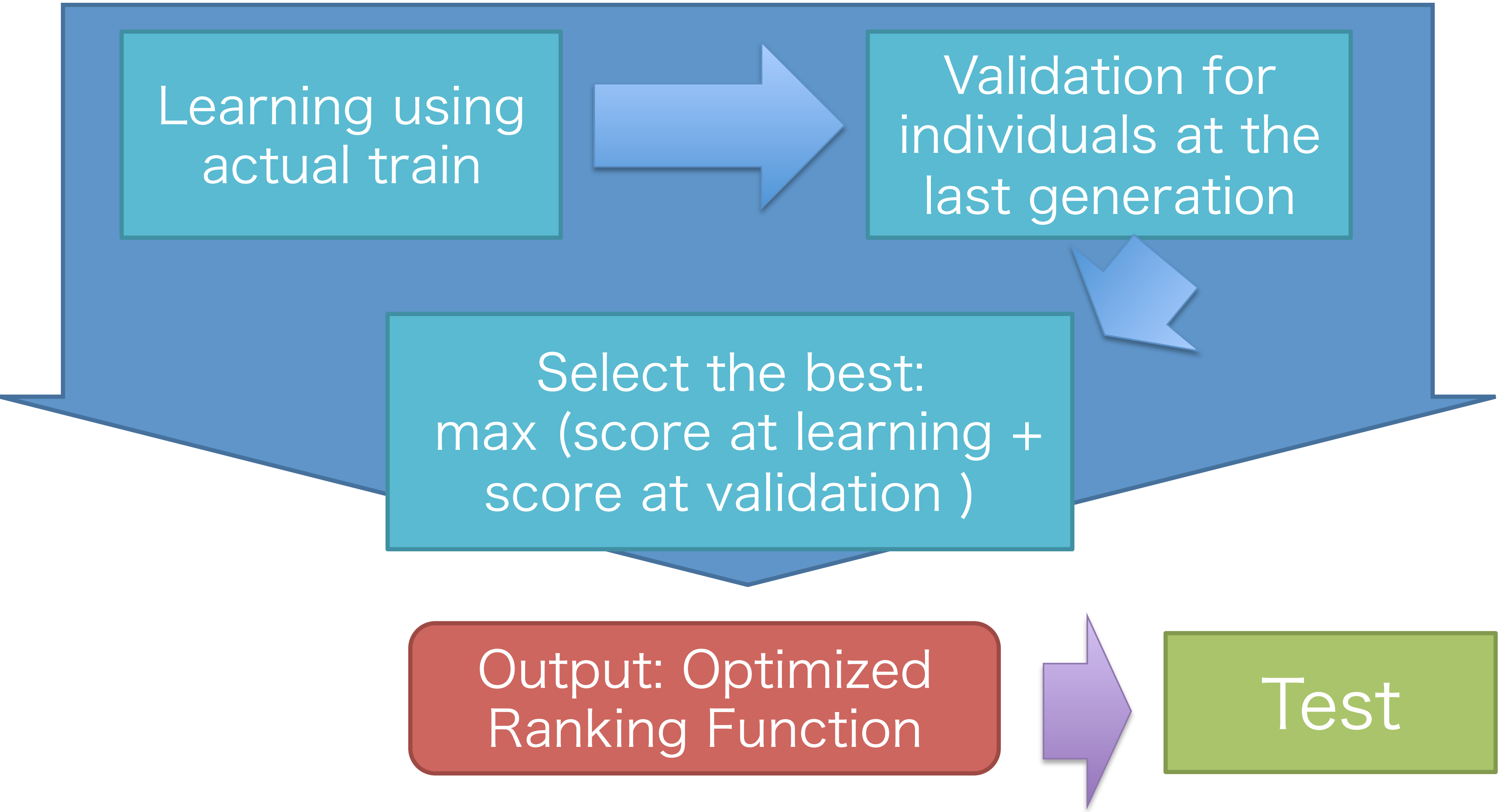
Dataset

LETOR : Released by Microsoft Research Asia



- ✓ Collected from real users
- ✓ Standard dataset used in many papers
- ✓ TD2003 (49171 lines) and TD2004 (74170 lines)
  - ✓ Such a large data that MAP calculation (fitness evaluation) is very time consuming
- ✓ 44 features

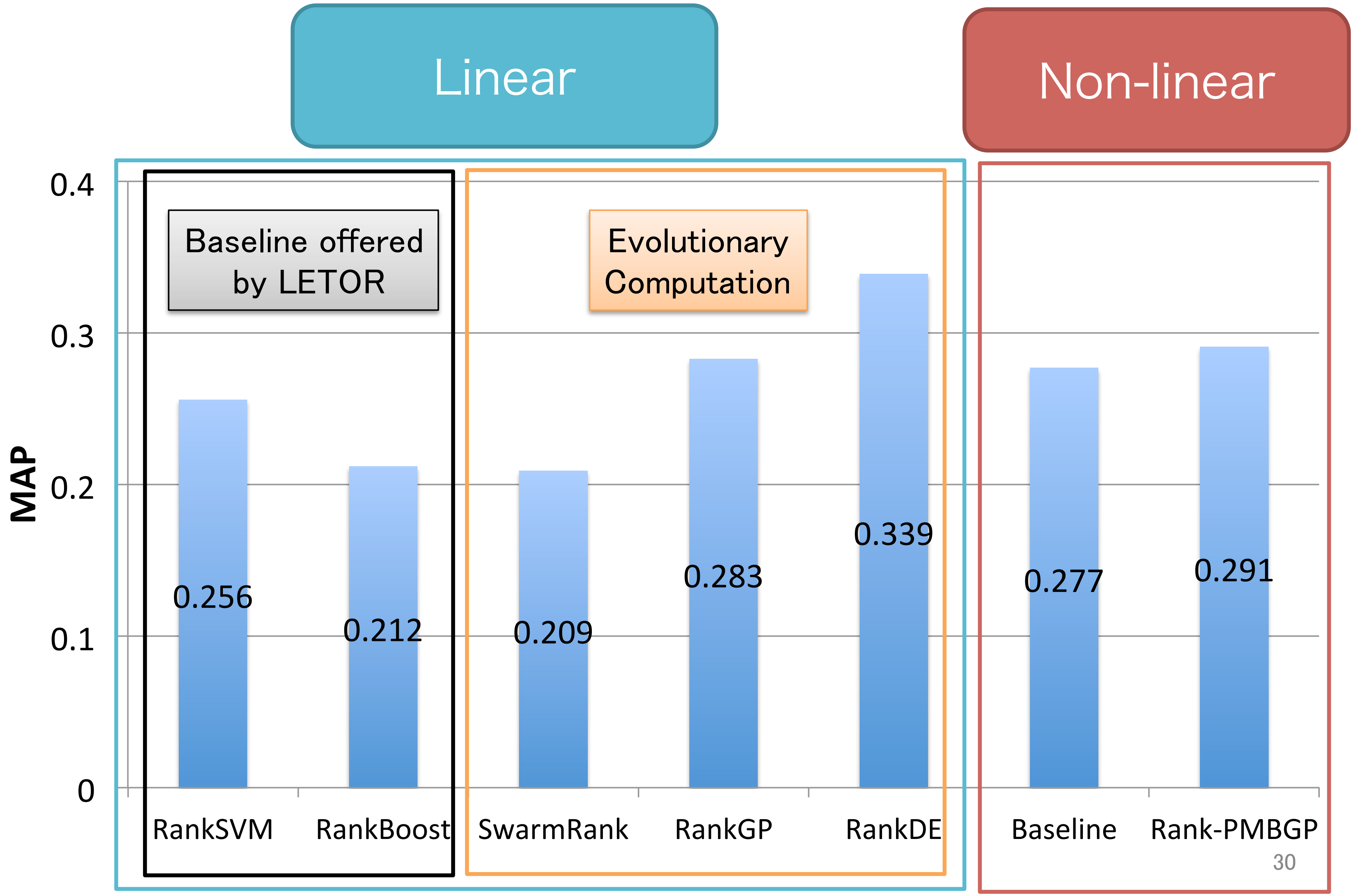
The Flow of experiments



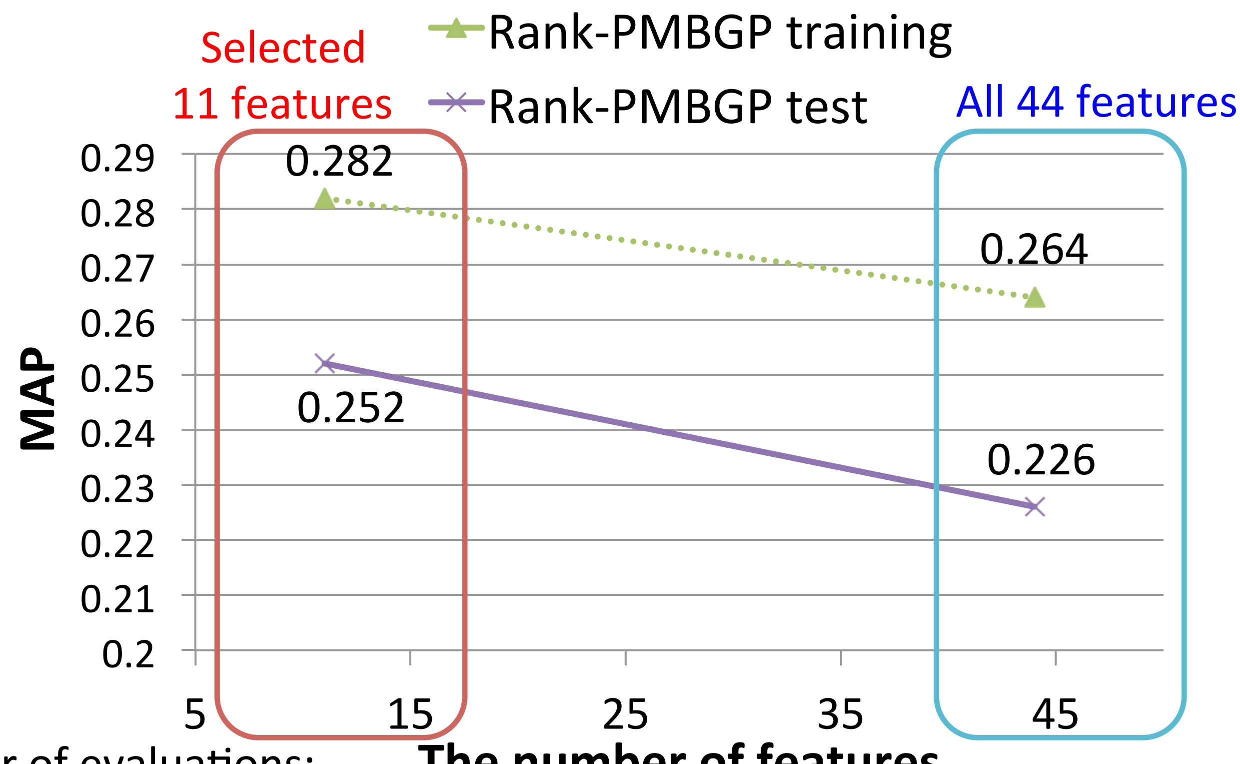
Methods for comparison

- RankSVM [R. Herbrich et al. 1999]
  - Using Support Vector Machine to discriminate relevance or not
- RankBoost [Y. Freund et al. 2003]
  - An application of Adaboost to learning to rank
- SwarmRank [E. Diaz-Aviles et al. 2009]
  - Optimize linear ranking function by PSO (Particle Swarm Optimization)
- RankGP [J. Y. Yeh et al. 2007]
  - Optimize linear ranking function using GP
- RankDE [D. Bollegala et al. 2011]
  - This achieves best score among evolutionary computation based learning to rank
  - Optimize linear ranking function using DE (Differential Evolution)
- Our Baseline
  - Optimize non-linear ranking function using GP
  - Extension of RankGP

Comparison with existing methods



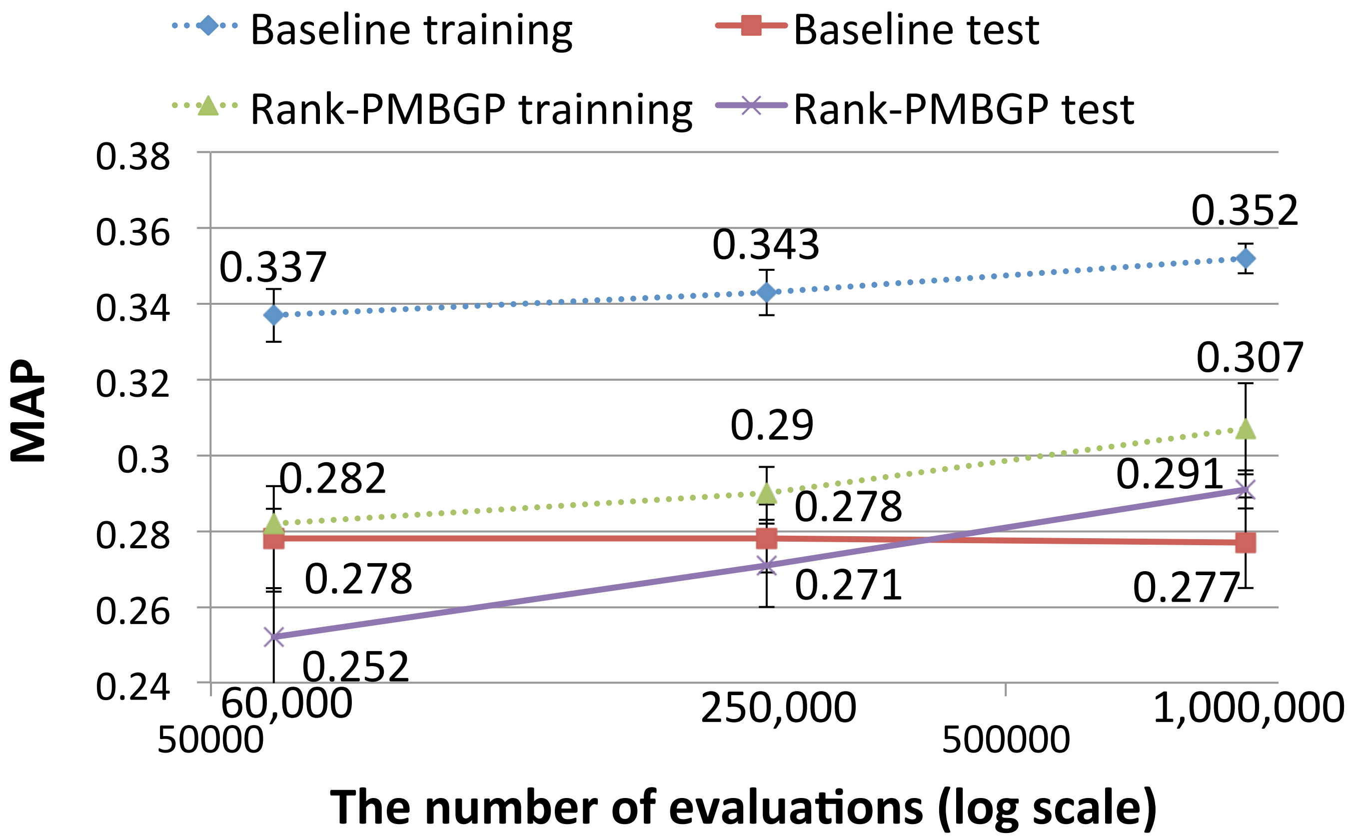
The effect of Differential Evolution based feature selection



※The number of evaluations: 60,000

Feature selection improves MAP score

Baseline (GP) versus Rank-PMBGP (TD2003)



Although Baseline does not improve, Rank-PMBGP improves as the number of evaluations increases

Conclusion

- We proposed Probabilistic Model Building GP based method to optimize Non-linear Ranking Function
- The proposed method, **Rank-PMBGP**, outperforms GP based Baseline and some of the existing methods
- Although feature selection is effective, further research is required to reduce the search space
- Analysis of optimized ranking function is future work

Thank you!

If you have any questions,
please feel free to e-mail to

sato@iba.t.u-tokyo.ac.jp

Q&A

Why we employ POLE as Probabilistic Model Building GP?

- Learn graph structure and parameter at each generation
  - Better than other Bayesian network based PMBGPs with fixed structure
- Use EPT (Expanded Parse Tree)
  - Special function node push terminal symbols on trunk into leaves. In other words, terminal symbols appear only in leaves
  - Learning of Bayesian network becomes easy since symbols on trunk is reduced (only functions)

Features in LETOR

- low-level content features
  - tf: term frequency
  - idf: inverse document frequency
  - dl: document length
  - tfidf: multiplication of tf and idf
- high-level content features
  - BM25
  - LMIR
- Hyperlink-based features
  - PageRank
  - Topical PageRank
  - HITS
  - Topical HITS
  - HostRank
- Hybrid features
  - Hyperlink-based relevance propagation
  - Site map-based relevance propagation
- Total: 44 features in the LETOR-2 dataset

Why did the non-linear proposed method lose to linear RankDE?

- Could not search non-linear vast search space thoroughly
  - Learning is not saturated at 1,000,000 fitness evaluations
  - We could not increase the number of fitness evaluations more since MAP calculation is very time consuming
    - 1 run (5 folds) takes over 24 hours
    - A future work is to reduce the number of evaluations
- Note that overfitting did not occur otherwise did in GP based baseline

The number of evaluations

- 60,000
  - Population size: 600, maximum generation: 100
- 250,000
  - Population size: 5000, maximum generation: 50
- 1,000,000
  - Population size: 10000, maximum generation: 100

Parameters of Rank-PMBGP

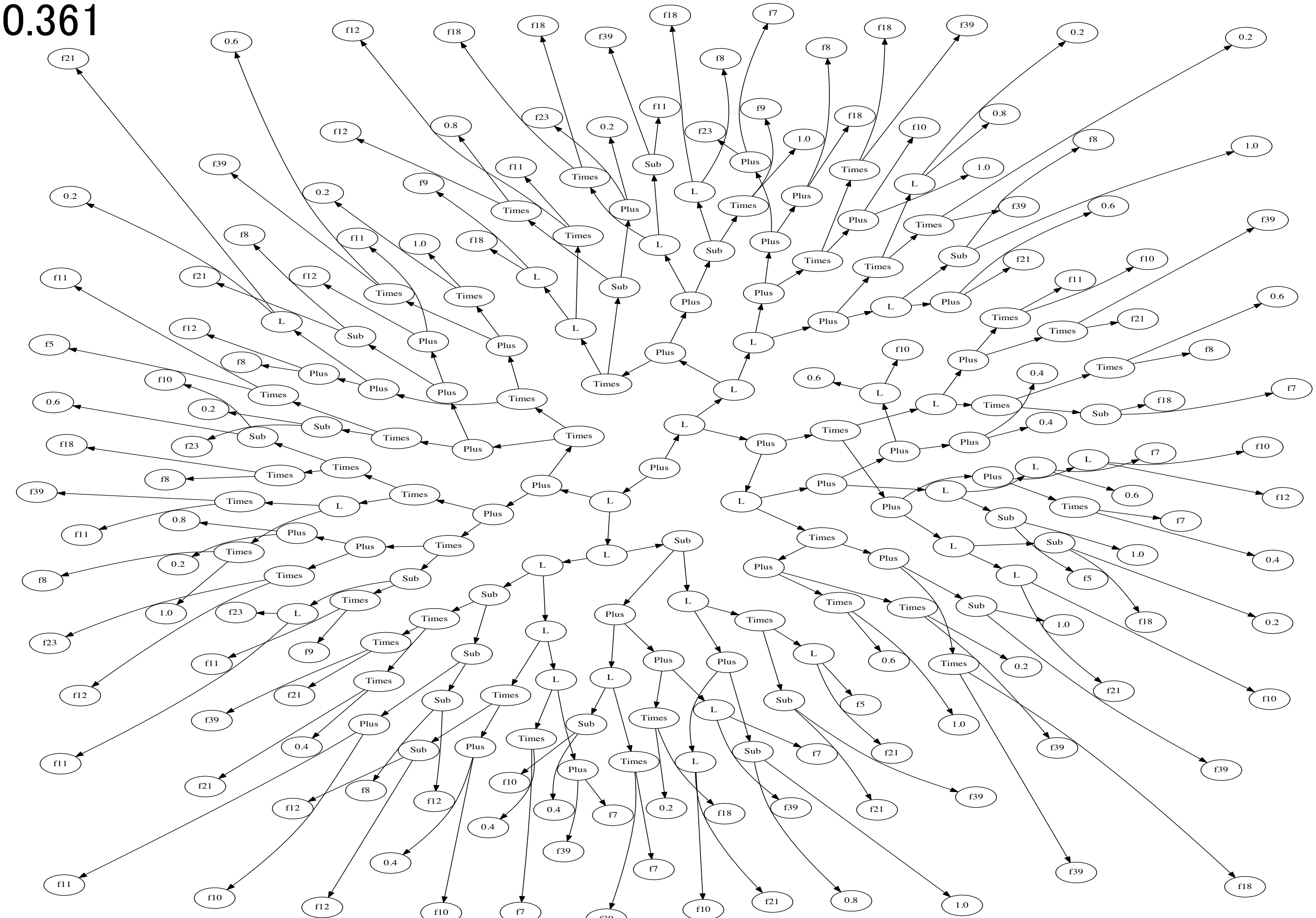
Parameters/Nodes	Settings
P_s	if population size is larger than 5000 use 0.05 otherwise use 0.2
P_e	if population size is larger than 5000 use 1 otherwise use 0.005
P_F	0.9
S_f	{+,-,*} (all function takes two arguments)
S_v	11 features (id : name) 5: dl of URL 7: HITS hub 8: HostRank 9: idf of body 10: idf of anchor 11: idf of title 12: idf of URL 18: LMIRJM of anchor 21: LMIRDIR of extracted title 23: LMIRABS of title 39: Hyperlink base score propagation (weighted in-link) }
S_c	$\{0.2, 0.4, 0.6, 0.8, 1.0\}$
The number of terminal symbols	16
depth limitation	8

Parameters of Baseline using GP

Parameter	Definition	Value
P_e	Elitist Reproduction Rate	Only 1 individual
P_c	Crossover Rate	Initial value = 0.95, then change dynamically using AMRT
P_m	Mutation Rate	Initial value = 0.05, then change dynamically using AMRT
size_t	Tournament Size	5
P_F	Functional Selection Rate	0.9

An example of optimized ranking function by Rank-PMBGP（Fold4）

MAP：0.361



Some measures to evaluate ranked list of documents

$$P@n = \frac{\text{No. of relevant docs in top } n \text{ results}}{n}$$

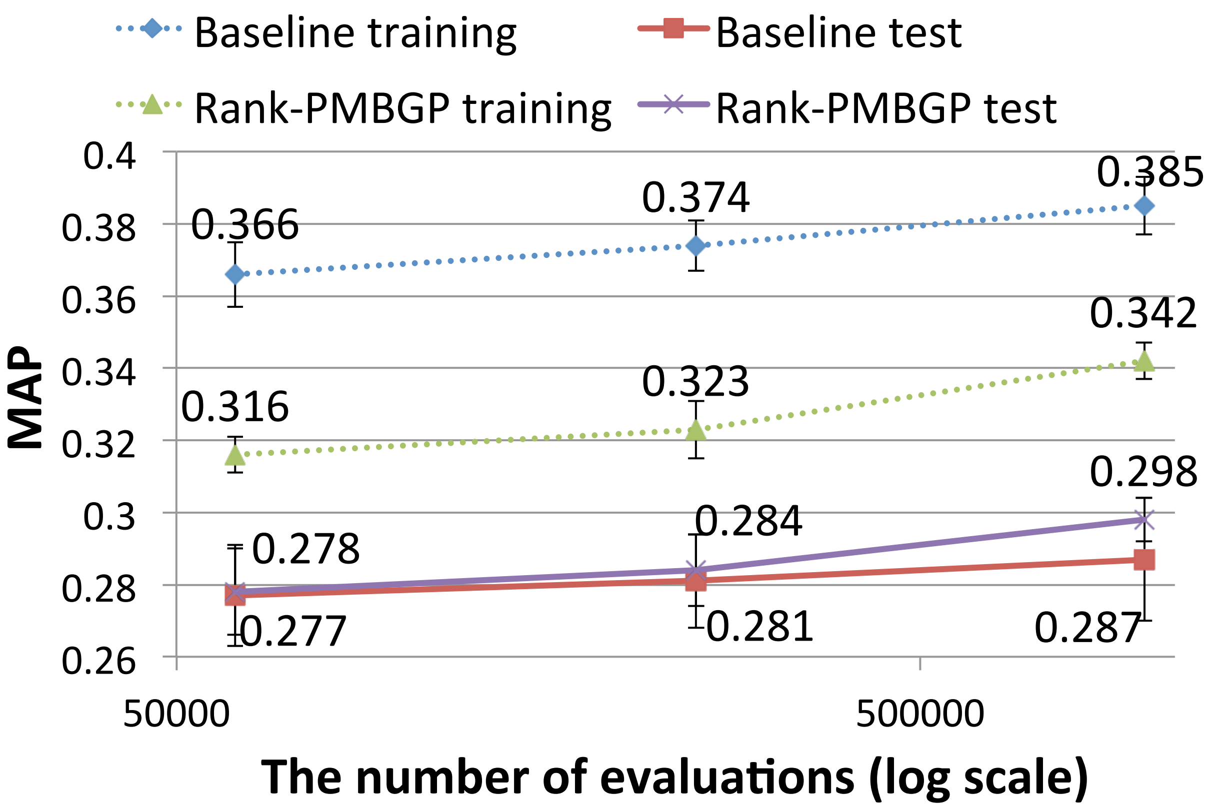
$$MAP = \frac{\sum_{q=1}^{Q} AP}{\text{No. of queries}}$$

$$AP = \frac{\sum_{n=1}^{N} (P@n \times rel(n))}{\text{No. of relevant docs for this query}}.$$

$$NDCG@n = Z_n \sum_{j=1}^{n} \frac{2^{rel(j)} - 1}{\log(1+j)}$$

$$Z_n = \frac{1}{\sum_{j=1}^{n} \frac{1}{\log(1+j)}}$$

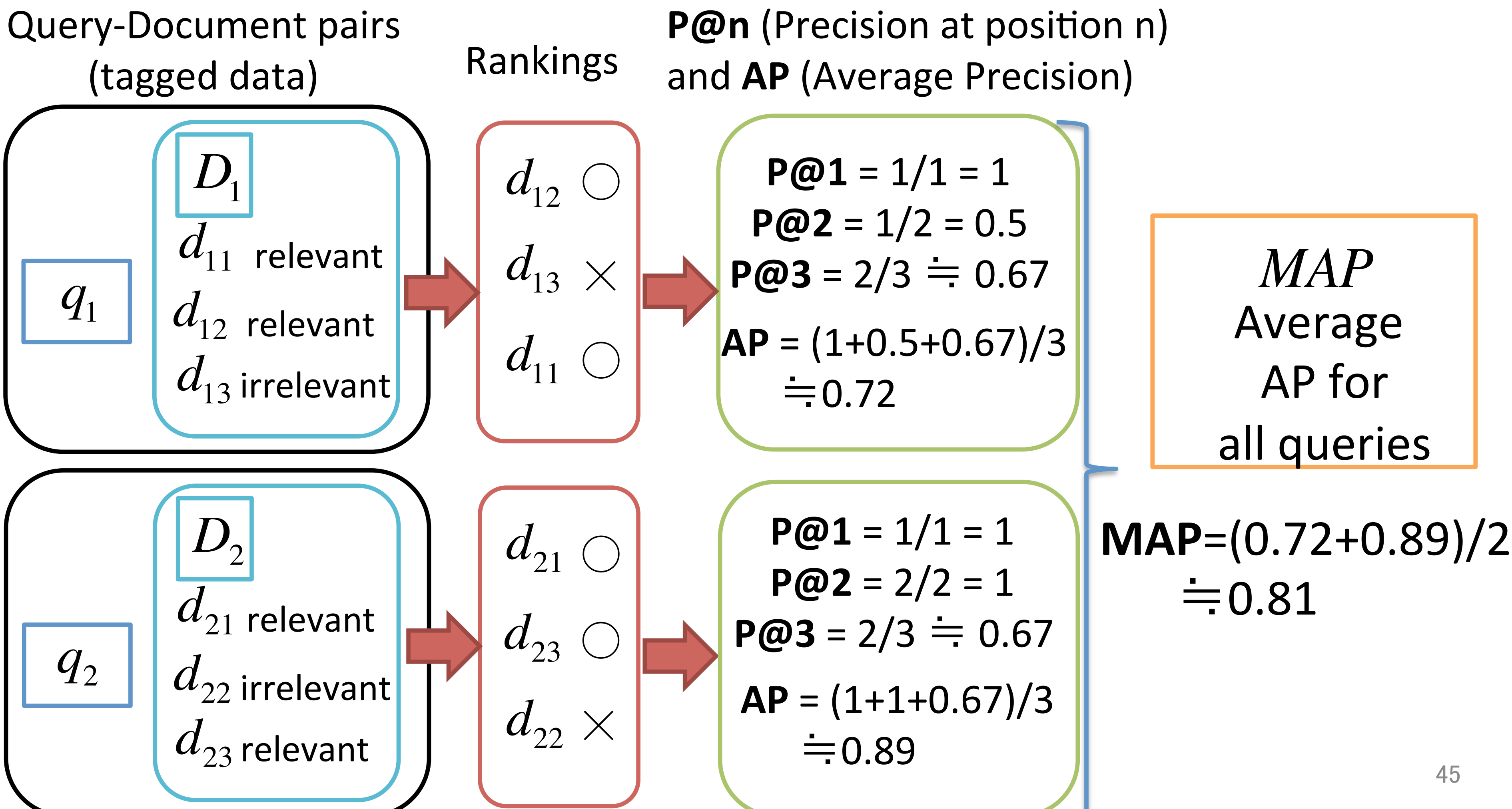
Result on TD2004



The same tendency is observed

MAP (Mean Average Precision)

- Popular evaluation method for ranking, but time consuming
- Employed as <u>fitness function in the proposed method</u>



Problem Settings

- When query q and documents D are given, output *proper* ranked list of documents
- 問題設定
- 学習データの集め方
- 特徴量