10. Learning to Rank

Outline

10.1. Why Learning to Rank?

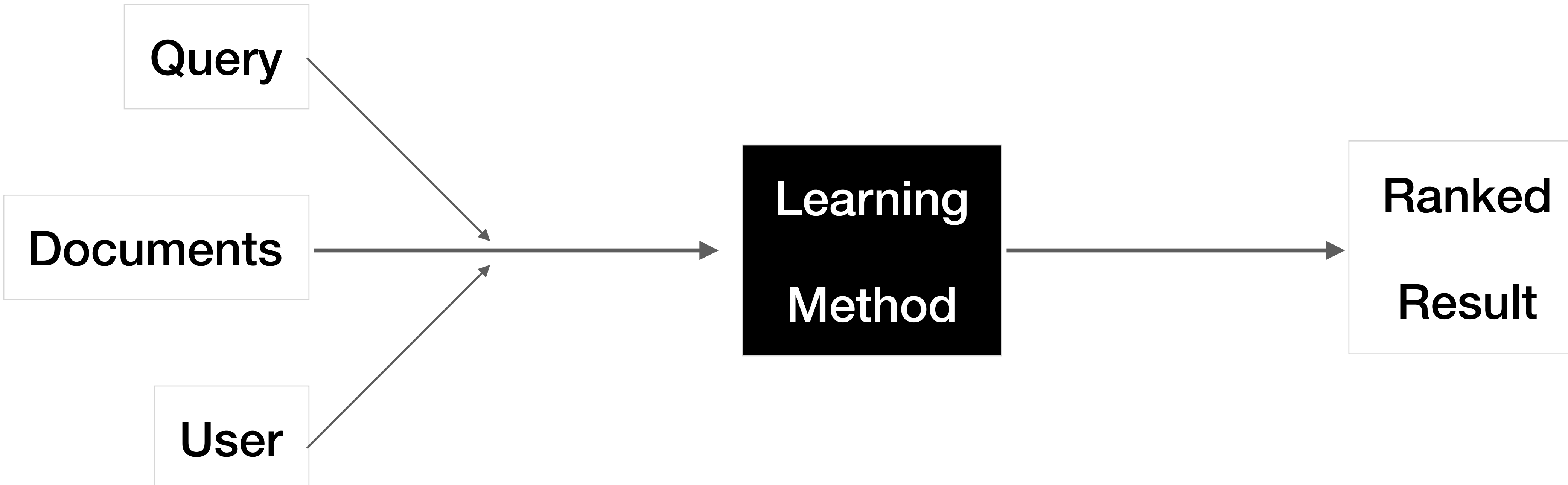
- **Various features** (signals) exist that can be used for ranking
  - **textual relevance** (e.g., determined using a **LM** or **Okapi BM25**)
  - **proximity** of query keywords in document content
  - **link-based importance** (e.g., determined using **PageRank**)
  - **depth of URL** (top-level page vs. leaf page)
  - **spamminess** (e.g., determine using **SpamRank**)
  - **host importance** (e.g., determined using **host-level PageRank**)
  - **readability** of content
  - …

Why Learning to Rank?

- **Traditional approach** to combining different features
  - **normalize features** (zero mean, unit standard deviation)
  - **feature combination function** (typically: weighted sum)
  - **tune weights** (either manually or exhaustively via grid search)

- **Learning to rank** makes combining features more systematic
  - builds on **established methods** from Machine Learning
  - allows different targets derived from **different kinds of user input**
  - active area of research for **past ~10 years**
  - early work by Norbert Fuhr [1] from **1989**

10,000 ft. View



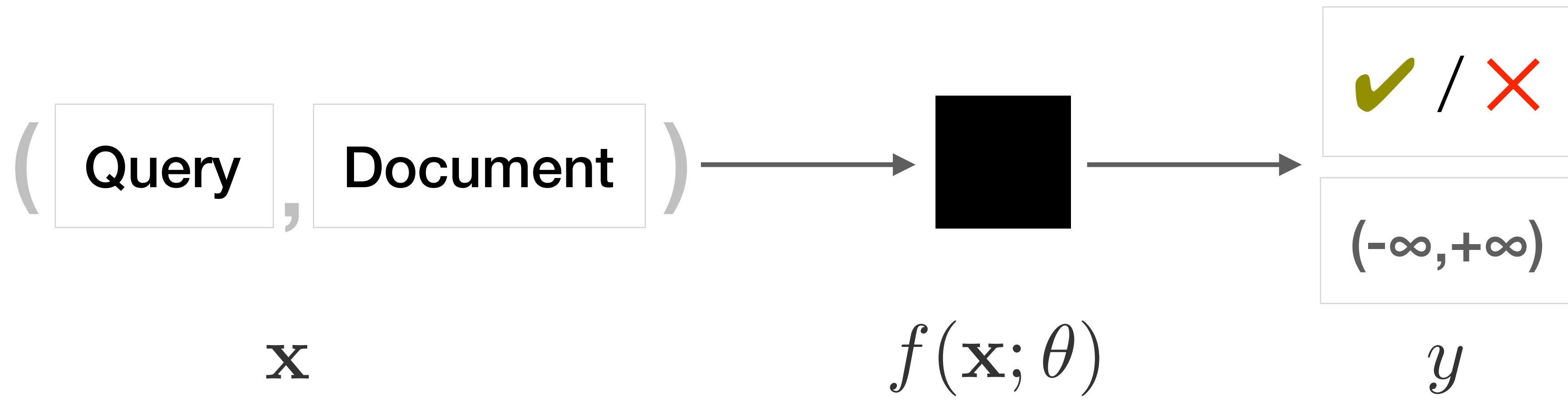
- <u>Open Issues</u>:
  - how do we **model** the problem?
  - is it a **regression or classification** problem?
  - what is our **prediction target**?

10.2. Pointwise, Pairwise, Listwise

- Learning to rank problem can be **modeled** in three different ways
  - predict goodness of **individual documents** (pointwise)
  - predict users' relative preference for **pairs of documents** (pairwise)
  - predict goodness of **entire query result** (listwise)

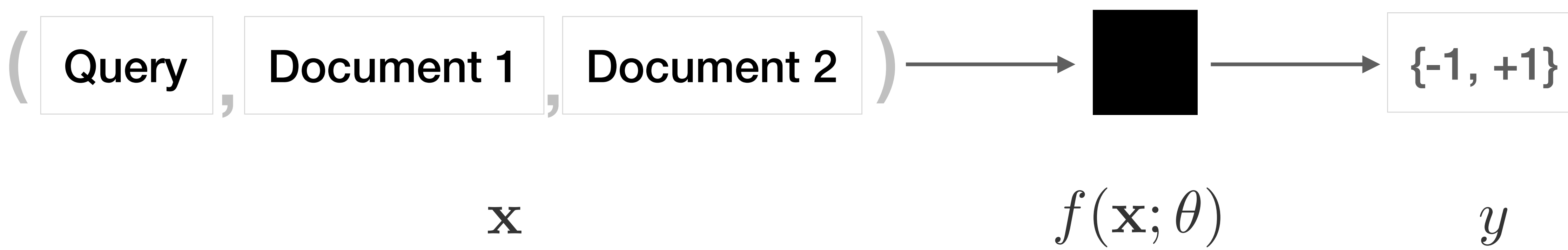
- Each way of modeling has **advantages and disadvantages**; for each of them **several (many) concrete approaches** exist
  - we'll stay at a **conceptual level**
  - for an **in-depth discussion** of concrete approaches see Liu [3]

Pointwise



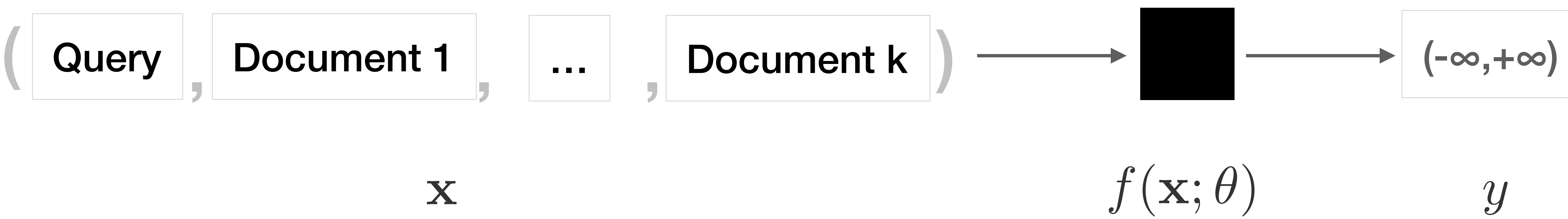
- **Pointwise approaches** predict
  - for **every document** based on its feature vector **x**
  - document **goodness y** (e.g., a label or measure of engagement)
  - training determines the **parameter** θ based on a **loss function** (e.g., root-mean-square error)

Pairwise



- **Pairwise approaches** predict
  - for every **pair of documents** based on a feature vector **x**
  - users' **relative preference** regarding the documents (**+1** shows preference for Document 1; **-1** for Document 2)
  - training determines the **parameter** θ based on a **loss function** (e.g., the number of inverted pairs)

Listwise



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

Typical Learning-to-Rank Pipeline

- Learning to rank is typically deployed as a **re-ranking step**, since it is infeasible to apply it to entire document collection

- Step 1: Determine a top-K result (K ~ 1,000) using a **proven baseline retrieval method** (e.g., Okapi BM25 + PageRank)

- Step 2: Re-rank documents from top-K using **learning to rank approach**, then return top-k (k ~ 100) to user

10.3. Gathering User Input

- Regardless of whether a pointwise, pairwise, or listwise approach is employed, **some input from the user is required** to determine prediction target **y**
  - **explicit user input** (e.g., relevance assessments)
  - **implicit user input** (e.g., by analyzing their behavior)

Relevance Assessments

- Construct a **collection of (difficult) queries**, pool results from different baselines, and gather **graded relevance assessments** from human assessors

- Problems:
  - **hard to represent query workload** within 50, 500, 5K queries
  - difficult for queries that require **personalization or localization**
  - expensive, time-consuming, and subject to Web dynamics

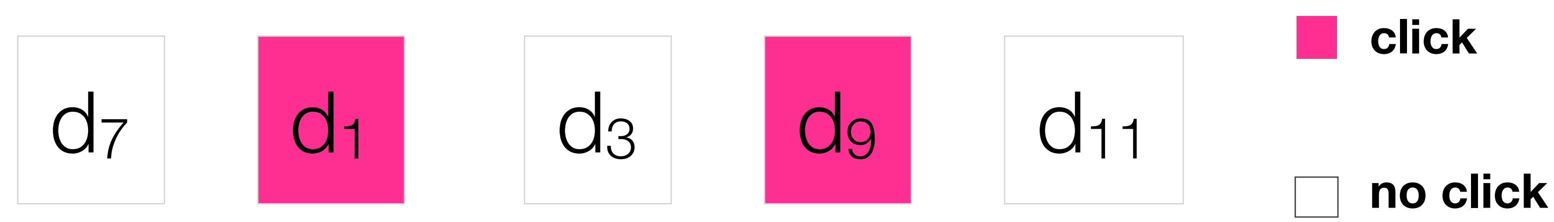
Clicks

- Track **user behavior** and measure their **engagement** with results
  - **click-through rate** of document when shown for query
  - **dwell time**, i.e., how much time did the user spend on the document
- Problems:
  - **position bias** (consider only first result shown)
  - **spurious clicks** (consider only clicks with **dwell time** above threshold)
  - **feedback loop** (add some randomness to results)
- Joachims et al. [2] and Radlinksi et al. [4] study the **reliability of click data**

Skips

- Joachims et al. [2] propose to **use skips in addition to clicks** as a source of **implicit feedback** based on user behavior

Top-5:



  - **skip previous**: $d_1 > d_7$ and $d_9 > d_3$ (i.e., user prefers d_1 over d_7)
  - **skip above**: $d_1 > d_7$ and $d_9 > d_3$, $d_9 > d_7$

- Users study reported in [2] shows that derived relative preferences
  - are **less biased** than measures merely based on clicks
  - show **moderate agreement** with explicit relevance assessments

10.4. Learning to Rank Evaluation

- Several **benchmark datasets** have been released to allow for a comparison of different learning-to-rank methods
  - **LETOR** 2.0 (2007), 3.0 (2008), 4.0 (2009) by Microsoft Research Asia based on publicly available document collections, comes with precomputed low-level features, relevance assessments
  - **Yahoo! Learning to Rank Challenge** (2010) by Yahoo! Labs comes with precomputed low-level features and relevance assessments
  - **Microsoft Learning to Rank Datasets** by Microsoft Research U.S. comes with precomputed low-level features and relevance assessments

- Examples of typical features:

Feature List of Microsoft Learning to Rank Datasets			
feature id	**feature description**	**stream**	**comments**
1		body	
2		anchor	
3	covered query term number	title	
4		url	
5		whole document	
6		body	
7		anchor	
8	covered query term ratio	title	
9		url	
10		whole document	
11		body	
12		anchor	

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

- Examples of typical features:

12	stream length	anchor	
13		title	
14		url	
15		whole document	
16	IDF(Inverse document frequency)	body	
17		anchor	
18		title	
19		url	
20		whole document	
21	sum of term frequency	body	
22		anchor	
23		title	
24		url	
25		whole document	
26		body	

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

- Examples of typical features:

26	min of term frequency	body
27		anchor
28		title
29		url
30		whole document
31	max of term frequency	body
32		anchor
33		title
34		url
35		whole document
36	mean of term frequency	body
37		anchor
38		title
39		url
40		whole document

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

- Examples of typical features:

41	variance of term frequency	body	
42		anchor	
43		title	
44		url	
45		whole document	
46	sum of stream length normalized term frequency	body	
47		anchor	
48		title	
49		url	
50		whole document	
51	min of stream length normalized term frequency	body	
52		anchor	
53		title	
54		url	
55		whole document	

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

- Examples of typical features:

96	boolean model	body	
97		anchor	
98		title	
99		url	
100		whole document	
101	vector space model	body	
102		anchor	
103		title	
104		url	
105		whole document	
106	BM25	body	
107		anchor	
108		title	
109		url	
110		whole document	

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

- Examples of typical features:

111		body	Language model approach for information retrieval (IR) with absolute discounting smoothing
112		anchor	
113	LMIR.ABS	title	
114		url	
115		whole document	
116		body	Language model approach for IR with Bayesian smoothing using Dirichlet priors
117		anchor	
118	LMIR.DIR	title	
119		url	
120		whole document	
121		body	Language model approach for IR with Jelinek-Mercer smoothing
122		anchor	
123	LMIR.JM	title	
124		url	
125		whole document	

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

- Examples of typical features:

126	Number of slash in URL		
127	Length of URL		
128	Inlink number		
129	Outlink number		
130	PageRank		
131	SiteRank		Site level PageRank
132	QualityScore		The quality score of a web page. The score is outputted by a web page quality classifier.
133	QualityScore2		The quality score of a web page. The score is outputted by a web page quality classifier, which measures the badness of

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

- Examples of typical features:

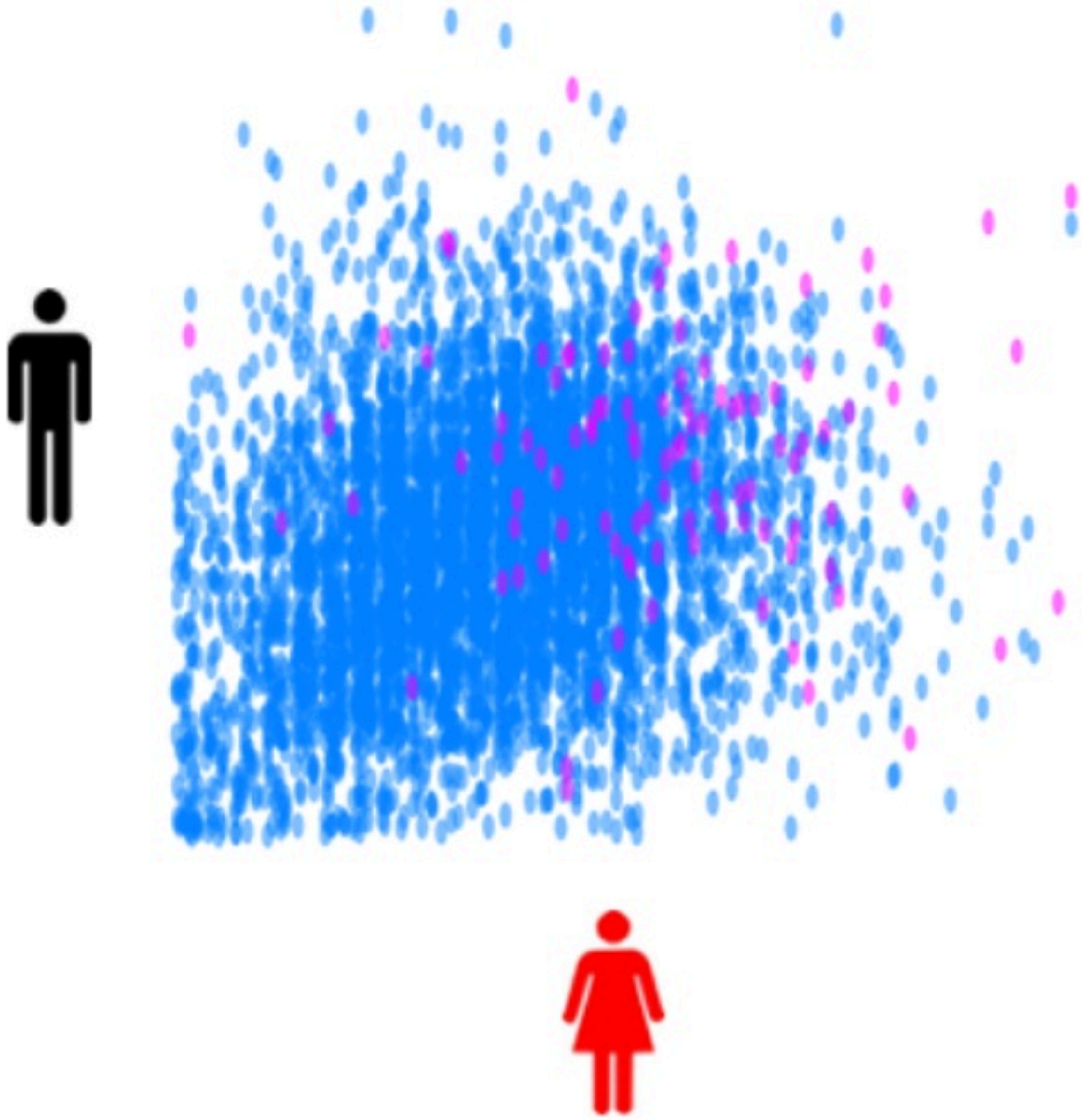
133	QualityScore2		web page. The score is outputted by a web page quality classifier, which measures the badness of a web page.
134	Query-url click count		The click count of a query-url pair at a search engine in a period
135	url click count		The click count of a url aggregated from user browsing data in a period
136	url dwell time		The average dwell time of a url aggregated from user browsing data in a period

- Full details: http://research.microsoft.com/en-us/um/beijing/projects/letor/

10.5. Beyond Search

- Learning to rank is applicable **beyond web search**
- Example: Matching in eHarmony.com
  - based on WSDM 2014 talk by Vaclav Petricek
  - Step 1: **Compatibility matching** based on 150 questions regarding personality, values, attitudes, beliefs
    ~predict **marital satisfaction**
  - Step 2: **Affinity matching** based on other features such as distance, height difference, zoom level of photo
    ~predict probability of **message exchange**
  - Step 3: **Match distribution** based on graph optimization problem (constrained max flow)
- Slides: http://www.slideshare.net/VaclavPetricek/data-science-of-love

Compatibility Matching › Obstreperousness



ob·strep·er·ous
/əbˈstrepərəs/

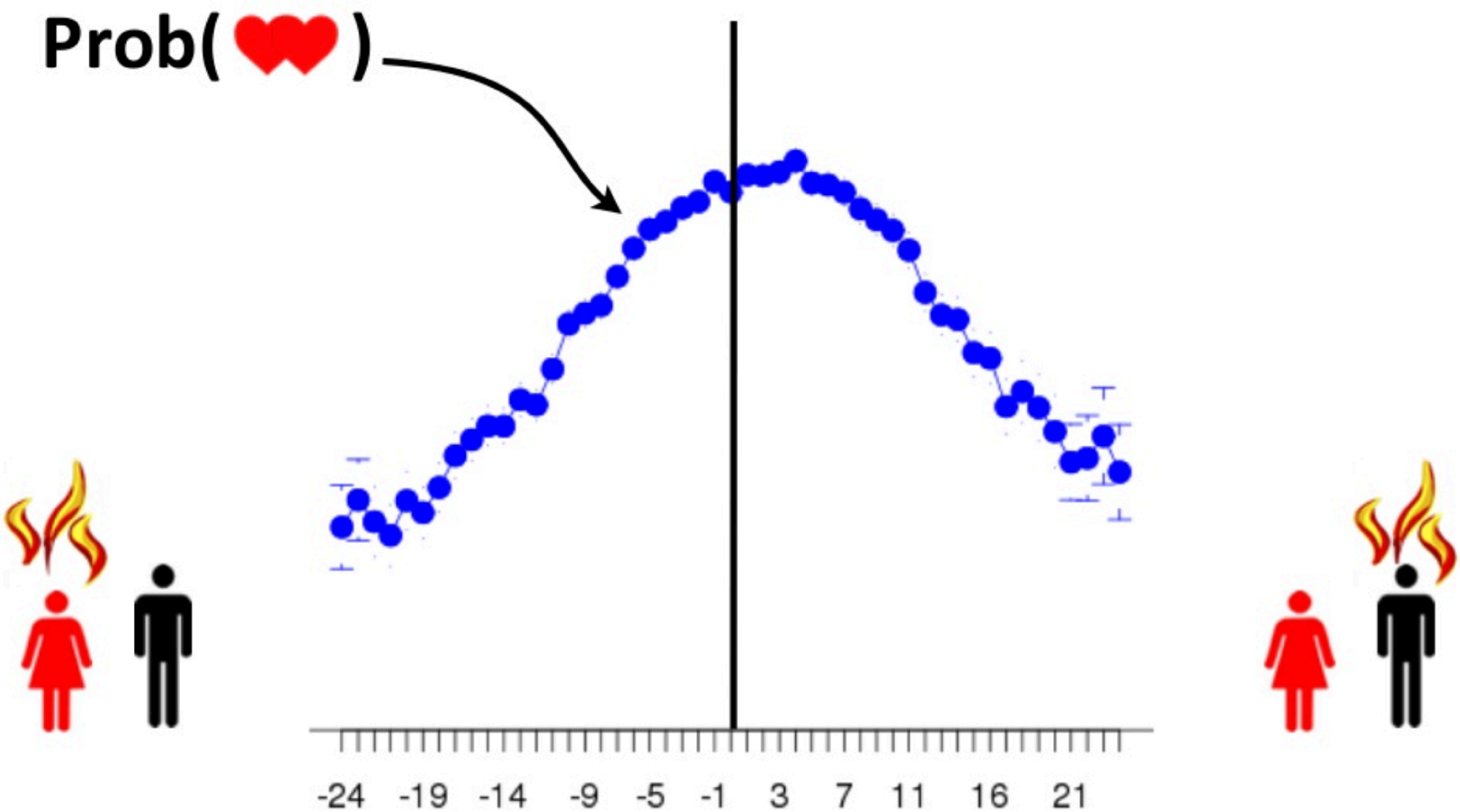
Adjective
Noisy and difficult to control: "the boy is cocky and obstreperous".

Synonyms
noisy - loud - clamorous - rumbustious - boisterous

10.5. Beyond Search

- Learning to rank is applicable **beyond web search**
- Example: Matching in eHarmony.com
  - based on WSDM 2014 talk by Vaclav Petricek
  - Step 1: **Compatibility matching** based on 150 questions regarding personality, values, attitudes, beliefs
~predict **marital satisfaction**
  - Step 2: **Affinity matching** based on other features such as distance, height difference, zoom level of photo
~predict probability of **message exchange**
  - Step 3: **Match distribution** based on graph optimization problem (constrained max flow)
- Slides: http://www.slideshare.net/VaclavPetricek/data-science-of-love

Affinity Matching › "Attractiveness"



10.5. Beyond Search

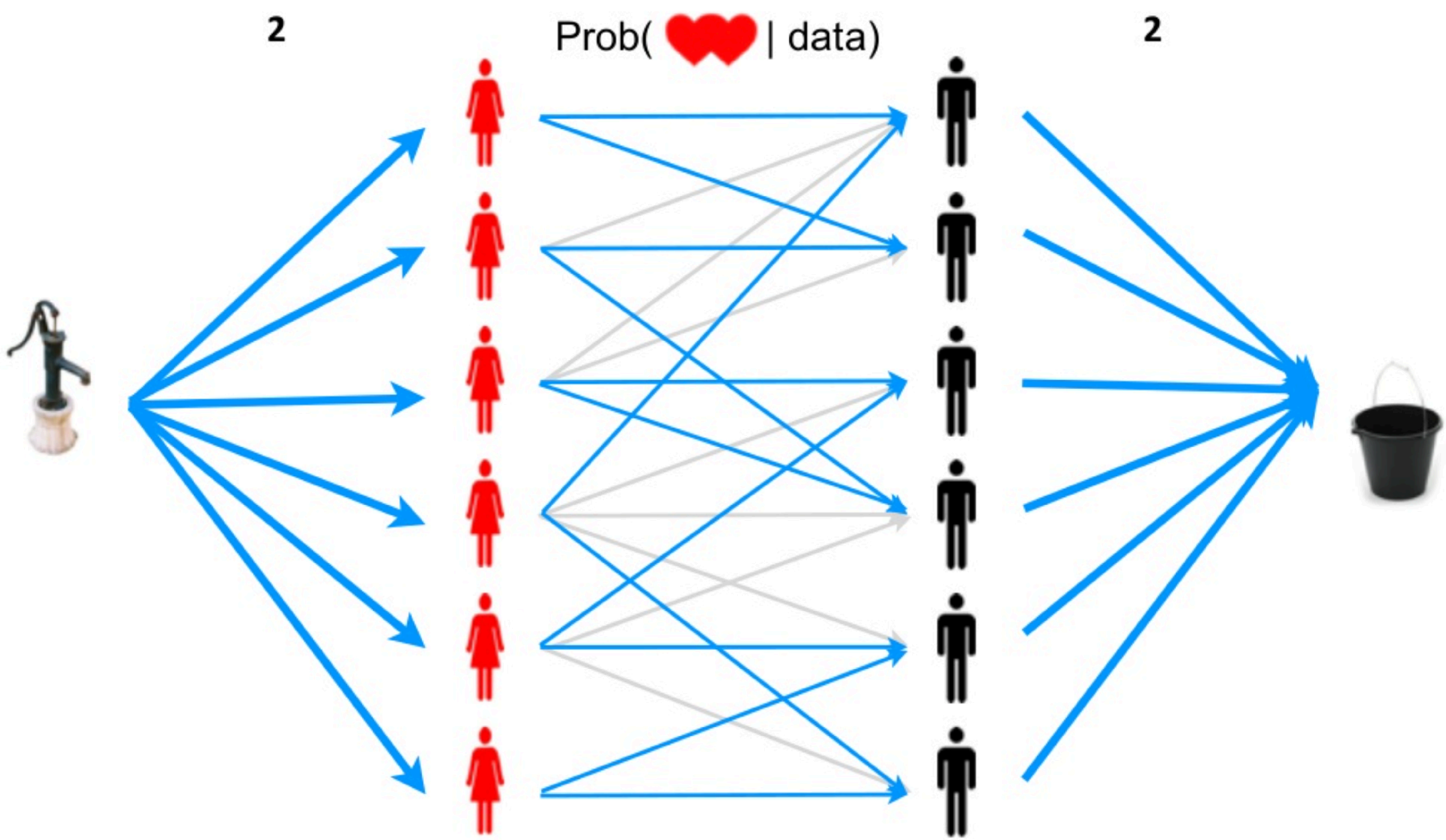
- Learning to rank is applicable **beyond web search**
- Example: Matching in eHarmony.com
  - based on WSDM 2014 talk by Vaclav Petricek
  - Step 1: **Compatibility matching** based on 150 questions regarding personality, values, attitudes, beliefs
    ~predict **marital satisfaction**
  - Step 2: **Affinity matching** based on other features such as distance, height difference, zoom level of photo
    ~predict probability of **message exchange**
  - Step 3: **Match distribution** based on graph optimization problem (constrained max flow)
- Slides: http://www.slideshare.net/VaclavPetricek/data-science-of-love

Affinity Matching › Zoom level

10.5. Beyond Search

- Learning to rank is applicable **beyond web search**
- Example: Matching in eHarmony.com
  - based on WSDM 2014 talk by Vaclav Petricek
  - Step 1: **Compatibility matching** based on 150 questions regarding personality, values, attitudes, beliefs
    ~predict **marital satisfaction**
  - Step 2: **Affinity matching** based on other features such as distance, height difference, zoom level of photo
    ~predict probability of **message exchange**
  - Step 3: **Match distribution** based on graph optimization problem (constrained max flow)
- Slides: http://www.slideshare.net/VaclavPetricek/data-science-of-love

Match Distribution › Graph optimization



10.5. Beyond Search

- Learning to rank is applicable **beyond web search**
- Example: Matching in eHarmony.com
  - based on WSDM 2014 talk by Vaclav Petricek
  - Step 1: **Compatibility matching** based on 150 questions regarding personality, values, attitudes, beliefs
    ~predict **marital satisfaction**
  - Step 2: **Affinity matching** based on other features such as distance, height difference, zoom level of photo
    ~predict probability of **message exchange**
  - Step 3: **Match distribution** based on graph optimization problem (constrained max flow)
- Slides: http://www.slideshare.net/VaclavPetricek/data-science-of-love

Summary

- **Learning to rank** provides systematic ways to **combine features**
- **Pointwise approaches**
  predict goodness of individual document
- **Pairwise approaches**
  predict relative preference for document pairs
- **Listwise approaches**
  predict effectiveness of ranked list of documents
- **Explicit and implicit user inputs**
  include relevance assessments, clicks, and skips
- Learning to rank is applicable **beyond web search**

References

[1] **N. Fuhr:** *Optimum Polynomial Retrieval Functions based on the the Probability Ranking Principle*, ACM TOIS 7(3), 1989

[2] **T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radklinski, G. Gay:** *Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search,* ACM TOIS 25(2), 2007

[3] **T.-Y. Liu:** *Learning to Rank for Information Retrieval*, Foundations and Trends in Information Retrieval 3(3):225–331, 2009

[4] **F. Radlinski and T. Joachims:** *Query Chains: Learning to Rank from Implicit* Feedback, KDD 2005