

Learning to Rank

Kevin Duh

kevinduh@cs.jhu.edu

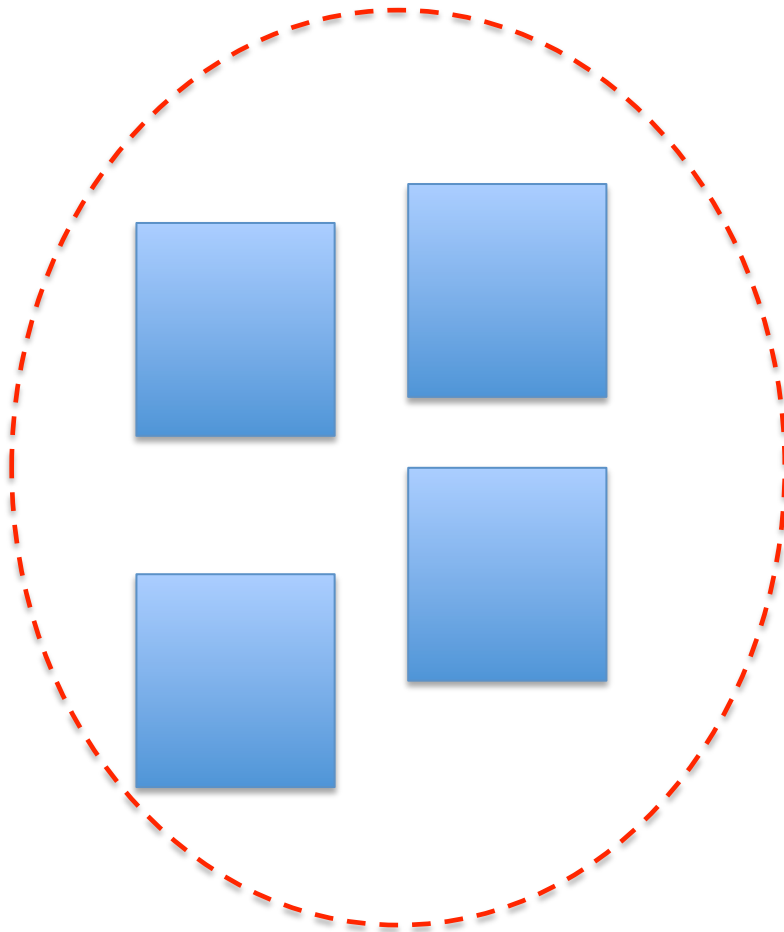
June 2018

Slides are online @

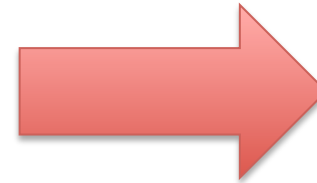
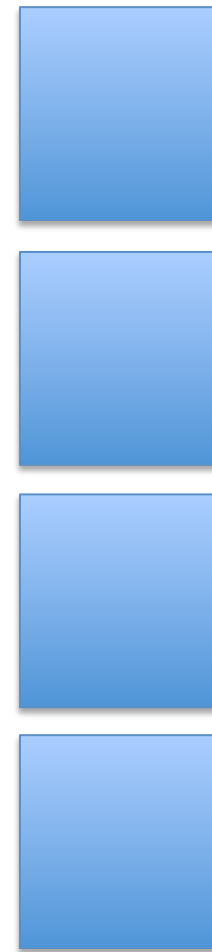
<http://cs.jhu.edu/~kevinduh/t/ltr.pdf>

What is Ranking?

UNORDERED SET



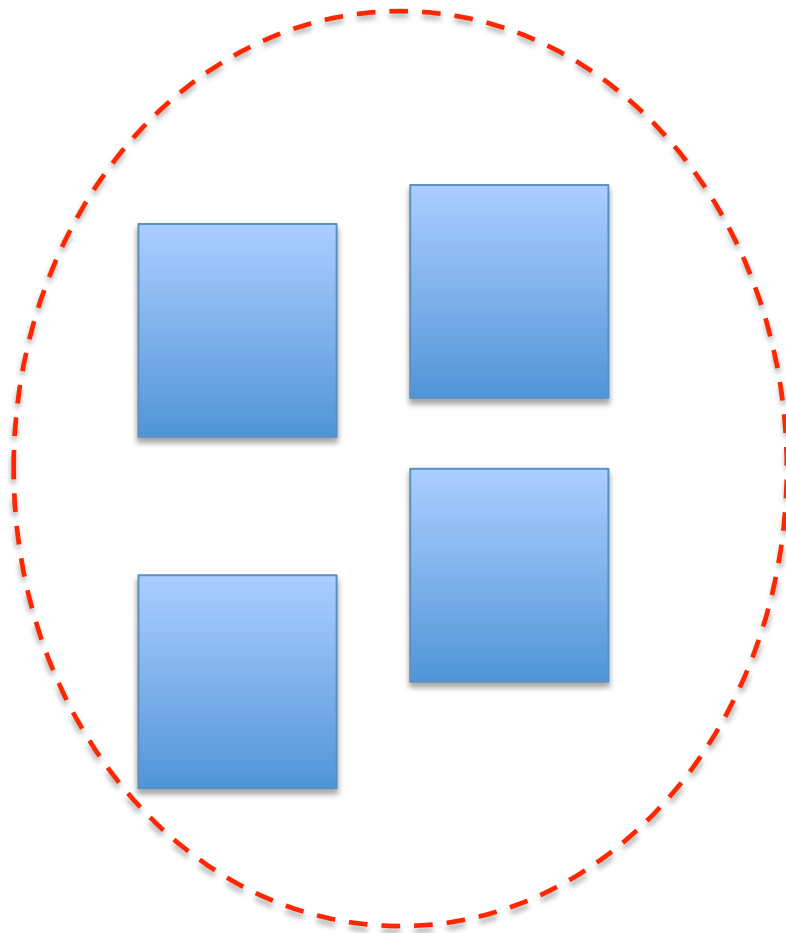
ORDERED LIST



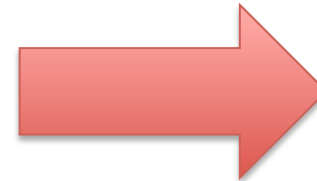
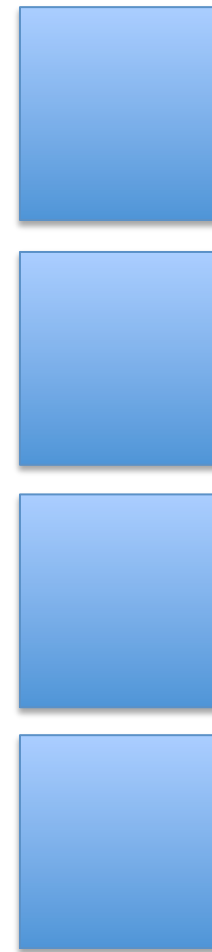
**RANKING
FUNCTION**

What is Learning to Rank?

UNORDERED SET



ORDERED LIST



**RANKING
FUNCTION**
*trained by
Machine
Learning*

Common in Search Engines



learning to rank



Web

Images

Videos

Maps

News

Explore

76,000,000 RESULTS

Any time ▾

Learning to rank - Wikipedia

https://en.wikipedia.org/wiki/Learning_to_rank ▾

Learning to rank or machine-learned ranking (MLR) is the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the ...

[PDF] Learning to Rank using Gradient Descent

research.microsoft.com/en-us/um/people/cburgess/papers/ICML_ranking.pdf

Learning to Rank using Gradient Descent that taken together, they need not specify a complete ranking of the training data). or even consistent.

1

2

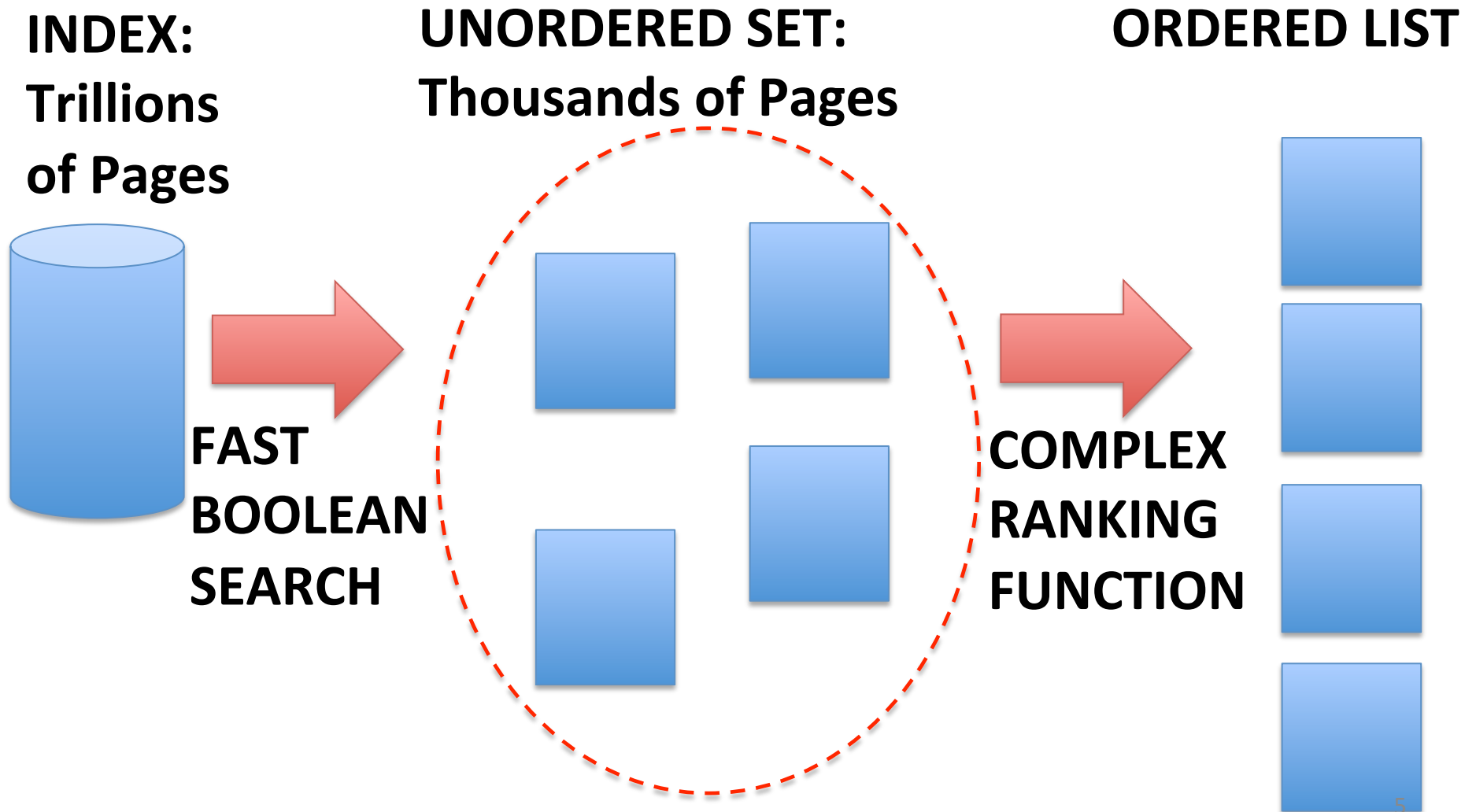
3

4

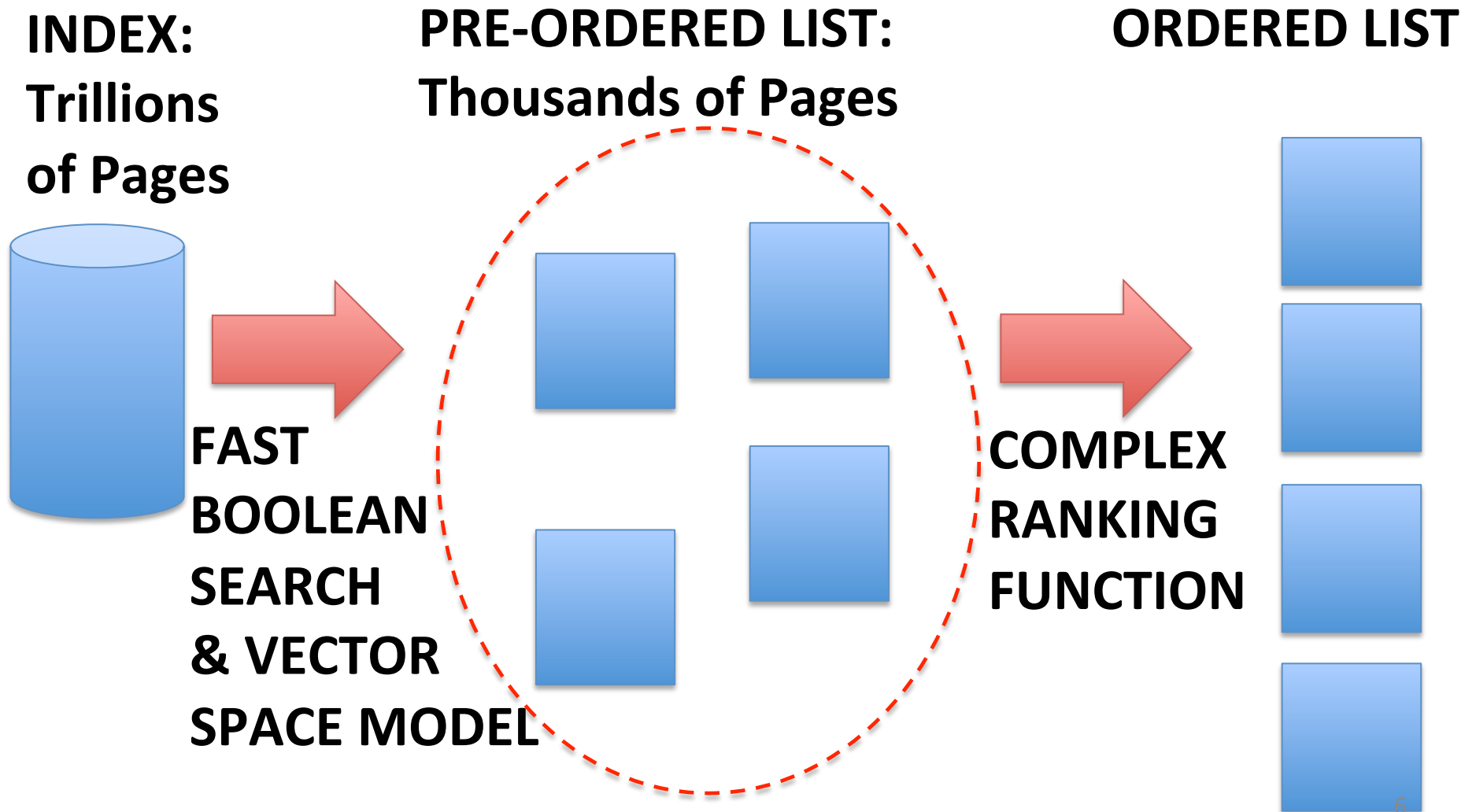
5



Anatomy of a Search Engine



Anatomy of a Search Engine



Motivation

- In search engines, only the top results matter
- Machine learning approach:
 - Enables more **features** (signal sources)
 - Improves over Boolean search, vector space models like tf-idf

Features of a top search result?



jhu cs courses 2016



- Word match?** **URL match?** **Popular page?** **Recently updated?**
- Johns Hopkins University Computer Science | Facebook**
<https://www.facebook.com/compscijhu> ▾
Johns Hopkins University Computer Science, Baltimore. 407 likes · 5 talking about this. Computer Science at Johns Hopkins University (CS@JHU) is a...
- 600.318/418: Operating Systems - Johns Hopkins University**
srl.cs.jhu.edu/courses/600.418/index.html ▾
Computer science majors and graduate students will be admitted regardless of enrollment limits. ... This course provides an introduction to operating systems.
- Department of Computer Science | Course Information**
www.cs.jhu.edu/course-info ▾
CS Course Catalog – complete list of departmental courses with descriptions. Note that not all courses are offered every year. Course Area Designators – a chart ...
- Johns Hopkins University • Free Online Courses and ...**
www.class-central.com ▸ Universities ▸ Johns Hopkins University ▾
Discover free online courses taught by Johns Hopkins University. Watch videos, do assignments, earn a certificate while learning from some of the best Professors.
- User intention match?**
Not spam?
Clickthrough log?

Useful Features

- Based on Query (q) and Document (d)
 - Various Boolean search and vector space model results, applied to document text, URL, title
 - Click-through: e.g. How many times d is clicked given q vs. How many times d is skipped
 - Results after Query-expansion
- Based on Document only (static)
 - Popularity of page: #Likes, #inlinks, Pagerank
 - Domain structure: main page or subpage
- Many, many more

Re-cap: Goal is to improve top ranked results via many features



jhu cs courses 2016



- Word match?** **URL match?** **Popular page?** **Recently updated?**
- Johns Hopkins University Computer Science | Facebook**
<https://www.facebook.com/compscijhu> ▾
Johns Hopkins University Computer Science, Baltimore. 407 likes · 5 talking about this. Computer Science at Johns Hopkins University (CS@JHU) is a...
- 600.318/418: Operating Systems - Johns Hopkins University**
srl.cs.jhu.edu/courses/600.418/index.html ▾
Computer science majors and graduate students will be admitted regardless of enrollment limits. ... This course provides an introduction to operating systems.
- Department of Computer Science | Course Information**
www.cs.jhu.edu/course-info ▾
CS Course Catalog – complete list of departmental courses with descriptions. Note that not all courses are offered every year. Course Area Designators – a chart ...
- Johns Hopkins University • Free Online Courses and ...**
www.class-central.com ▸ Universities ▸ Johns Hopkins University ▾
Discover free online courses taught by Johns Hopkins University. Watch videos, do assignments, earn a certificate while learning from some of the best Professors.
- User intention match?**
Not spam?
Clickthrough log?

Challenge: How to weight features?

- If there's only a few, we can manually tune.

e.g. score = 3 x #wordmatch + 1 x #inlink

- If there are many, we rely on machine learning (i.e. Learning to Rank)

Summary so far

1. Ranking: Unordered set → Ordered list
2. Search engines: only top results matter
3. Good ranking requires many features
4. Next: how to weight features

Problem Formulation



jhu cs courses 2016



1. Feature extraction

$\langle \text{query}, \text{doc}_1 \rangle \rightarrow \text{vector } x_1$

$\langle \text{query}, \text{doc}_2 \rangle \rightarrow \text{vector } x_2$

$\langle \text{query}, \text{doc}_3 \rangle \rightarrow \text{vector } x_3$

2. Apply ranking function & sort

$F(x_1) = 3 \rightarrow \text{Rank 2}$

$F(x_2) = 1 \rightarrow \text{Rank 3 (worst)}$

$F(x_3) = 4 \rightarrow \text{Rank 1 (best)}$

What function class for $F()$?

Assume linear weights: $F(x_i) = w^T x_i$

Learn **weights w** that replicate ranking on training set

Training Set

Query 1

$\langle \text{query}, \text{doc}_1 \rangle \rightarrow \text{vector } x_1$

$\langle \text{query}, \text{doc}_2 \rangle \rightarrow \text{vector } x_2$

$\langle \text{query}, \text{doc}_3 \rangle \rightarrow \text{vector } x_3$

Labels for each query-doc pair

$\text{label}(x_1) = 3$

$\text{label}(x_2) = 1$

$\text{label}(x_3) = 4$

Query 2

$\langle \text{query}, \text{doc}_1 \rangle \rightarrow \text{vector } x_1$

$\langle \text{query}, \text{doc}_2 \rangle \rightarrow \text{vector } x_2$

$\langle \text{query}, \text{doc}_3 \rangle \rightarrow \text{vector } x_3$

$\langle \text{query}, \text{doc}_4 \rangle \rightarrow \text{vector } x_4$

Labels for each query-doc pair

$\text{label}(x_1) = 3$ (Very relevant)

$\text{label}(x_2) = 2$ (Relevant)

$\text{label}(x_3) = 1$ (Slightly relevant)

$\text{label}(x_4) = 0$ (Irrelevant)

Where does the label come from?

- Human annotation
 - High quality, but expensive
- Click-through logs
 - Noisy, but cheap/abundant

Notation

query: $q^{(n)}$ $n = 1, \dots, N$

document for query n: $d_i^{(n)}$ $i = 1, \dots, I_n$

vector of D features per query-doc: $x_i^{(n)} \in \mathcal{R}^D$

label for each query-doc pair: $l_i^{(n)} \in \mathcal{Z}$

Training set: $\{q^{(n)}, \{x_i^{(n)}, l_i^{(n)}\}\}$

Ranking Function: $F(x_i^{(n)}) = w^T x_i^{(n)}$

Different training approaches

- How to optimize something on a set with a sort operation? Reduce to traditional regression/classification problems

Training Approach	Reduction
Point-wise	Document
Pair-wise	Two Documents
List-wise	All Documents per query

Point-wise Approach

Training set: $\{q^{(n)}, \{x_i^{(n)}, l_i^{(n)}\}\}$

Ranking Function: $F(x_i^{(n)}) = w^T x_i^{(n)}$

Find w that makes each $F(x)$ equal to its label

Training Objective: $\sum_n \sum_i (F(x_i^{(n)}) - l_i^{(n)})^2$

Training Objective: $\sum_n \sum_i (F(x_i^{(n)}) - l_i^{(n)})^2$

$\rightarrow \sum_z (F(x_z) - l_z)^2$ where z ranges over all i, n

Solve with linear regression!

Pair-wise Approach

Training set: $\{q^{(n)}, \{x_i^{(n)}, l_i^{(n)}\}\}$

Ranking Function: $F(x_i^{(n)}) = w^T x_i^{(n)}$

Find w that gives every pair the correct ranking

Training Objective:

$$F(x_i^{(n)}) > F(x_j^{(n)}) \quad \forall \quad i, j \quad \text{s.t.} \quad l_i^{(n)} > l_j^{(n)}$$

Training Objective:

$$\begin{aligned} F(x_i^{(n)}) &> F(x_j^{(n)}) \quad \forall \quad i, j \quad \text{s.t.} \quad l_i^{(n)} > l_j^{(n)} \\ \rightarrow F(x_i^{(n)}) - F(x_j^{(n)}) &> 0 \quad \forall \quad i, j \quad \text{s.t.} \quad l_i^{(n)} > l_j^{(n)} \\ \rightarrow w^T x_i^{(n)} - w^T x_j^{(n)} &> 0 \quad \forall \quad i, j \quad \text{s.t.} \quad l_i^{(n)} > l_j^{(n)} \\ \rightarrow w^T (x_i^{(n)} - x_j^{(n)}) &> 0 \quad \forall \quad i, j \quad \text{s.t.} \quad l_i^{(n)} > l_j^{(n)} \\ \rightarrow w^T (\delta_{ij}^{(n)}) &> 0 \quad \forall \quad i, j \quad \text{s.t.} \quad l_i^{(n)} > l_j^{(n)} \end{aligned}$$

Solve with binary classification!

Make a new sample out of every pair

Give new label: Positive for i,j pairs

Negative for j,i pairs

Disclaimer

- We've focused on very simple ranking functions (linear) for simplicity
- In practice, more complex functions (e.g. decision trees, neural nets) are common
- Recommend further reading:
 - Dawei Yin, et. al. "Ranking Relevance in Yahoo Search", Proceedings of KDD2016

Ranking Relevance in Yahoo Search

Dawei Yin[†], Yuening Hu[†], Jiliang Tang[†], Tim Daly Jr., Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, Jean-Marc Langlois, Yi Chang[†]

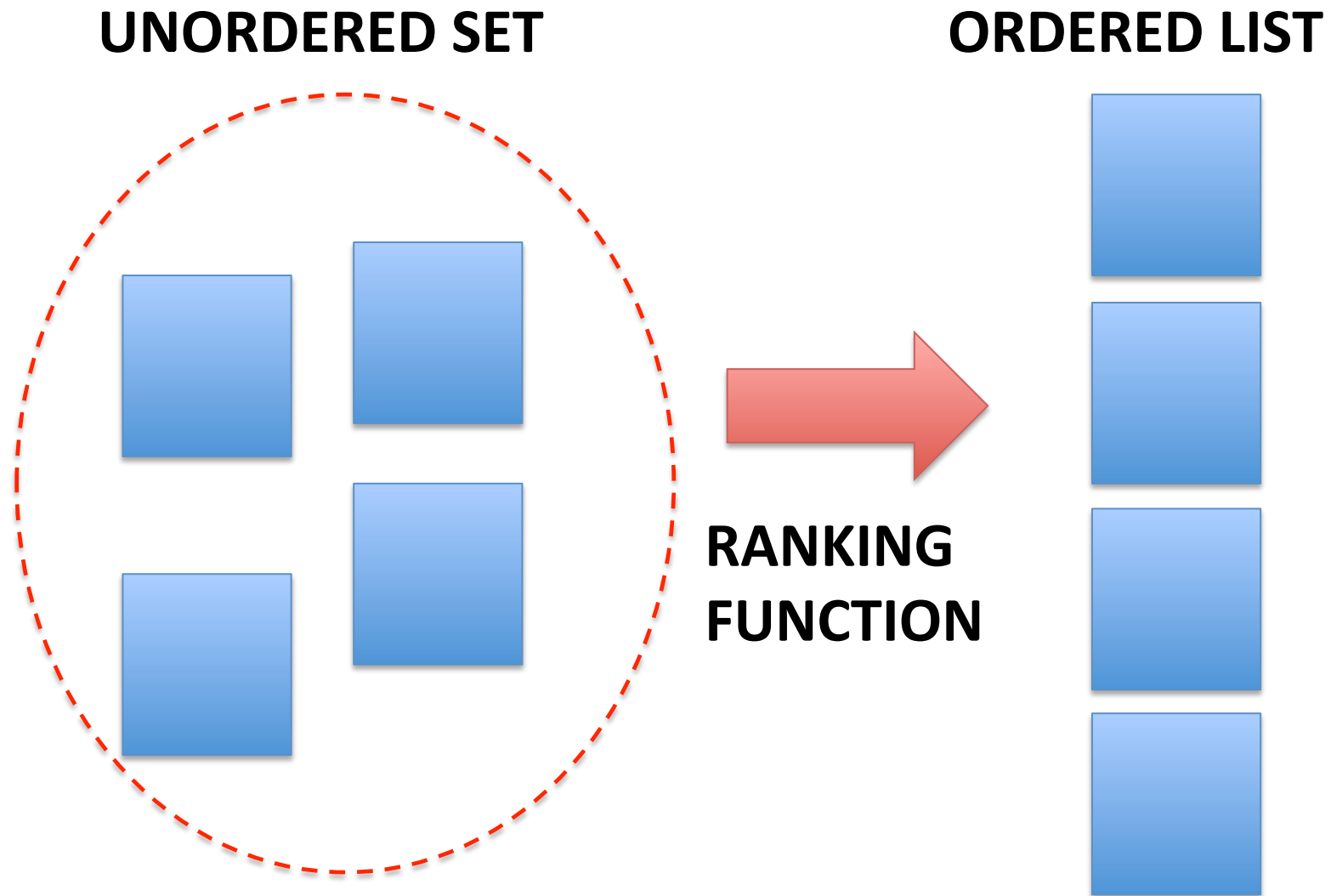
Relevance Science, Yahoo! Inc.

[†]{daweiy,ynhu,jlt,yichang}@yahoo-inc.com

query	methods	DCG1	DCG3	DCG5
all	LogisticRank	4.31	7.74	9.78
	GBRank	4.26 (-1.19%)	7.52 (-2.81%)*	9.51 (-2.81%)*
	LambdaMart	4.24 (-1.60%)	7.36 (-4.84%)*	9.21 (-5.83%)*
top	LogisticRank	5.69	9.67	12.08
	GBRank	5.56 (-2.22%)	9.25 (-4.29%)*	11.51 (-4.67%)*
	LambdaMart	5.59 (-1.72%)	9.08 (-6.04%)*	11.02 (-8.75%)*
torso	LogisticRank	3.88	7.23	9.26
	GBRank	3.88 (-1.77%)	7.065 (-2.30%)*	9.08 (-2.03%)*
	LambdaMart	3.81 (-1.88%)	6.97 (-3.64%)†	8.92 (-3.64%)*
tail	LogisticRank	2.91	5.65	7.16
	GBRank	2.99 (3.06%)	5.65 (0.01%)	7.19 (0.37%)
	LambdaMart	2.88 (-0.71%)	5.42 (-4.15%)†	6.91 (-2.78%)†

Table 1: Performance comparison of models using different learning algorithms. * denotes p-value $\leq$ 0.01; † denotes p-value $\leq$ 0.05.

Other applications of Learning to Rank



Other applications:

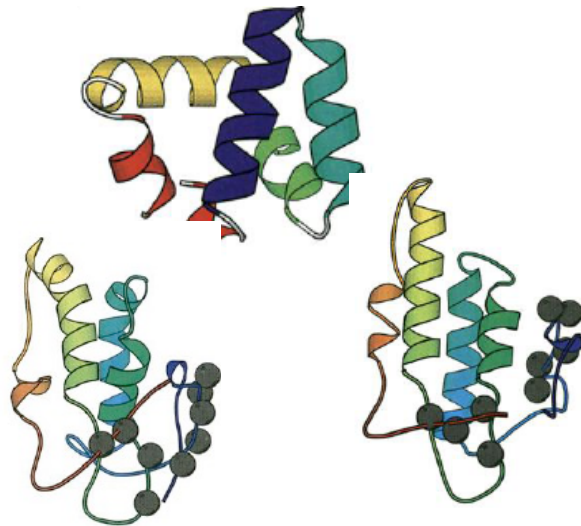
Protein structure prediction

Amino Acid Sequence:

MMKLKSNQTRTYDGDGYKKRAACLCFSE



*various protein
folding simulations*

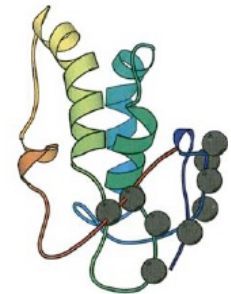


Candidate 3-D Structures

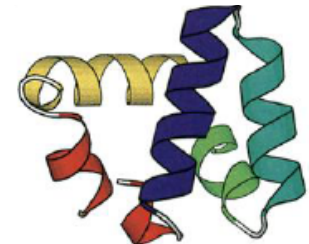
Ranking Function



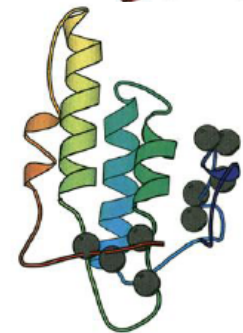
1st



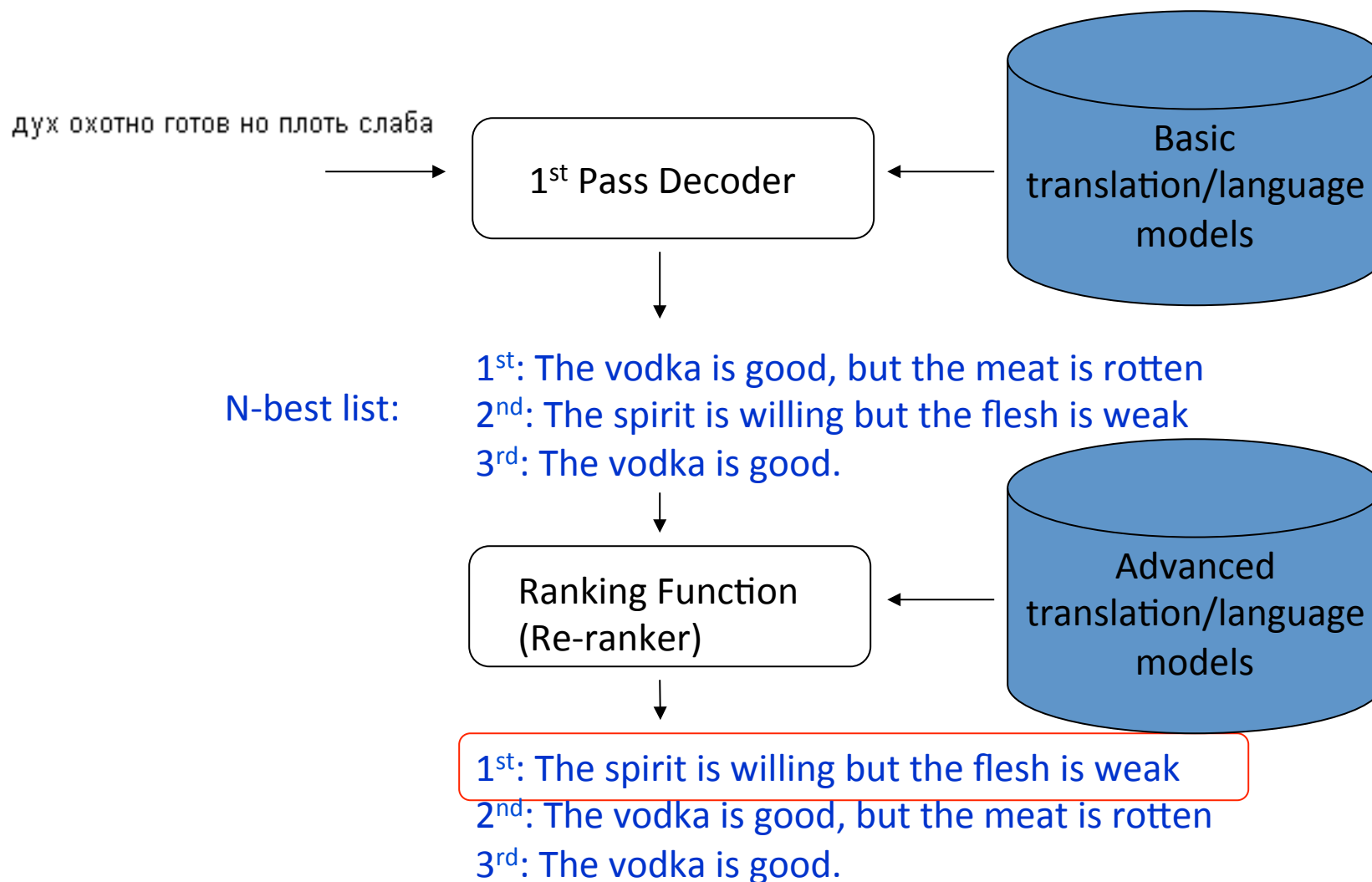
2nd



3rd



Other applications: Machine Translation



Summary

1. Ranking: Unordered set $\rightarrow$ Ordered list
2. Search engines: only top results matter
3. Good ranking requires many features
4. Approaches to learn weights of features
(reduction to classification/regression)

Lab: Build a learning-to-rank system

- Step 0: Download a Learning to Rank dataset
 - <http://www.cs.jhu.edu/~kevinduh/t/letor.tgz>
 - Source: (OHSUMED in LETOR3.0)
<http://research.microsoft.com/en-us/um/beijing/projects/letor/letor3download.aspx>

```
kduh@a14:~$ mkdir ltr; cd ltr
```

```
kduh@a14:~/ltr$ wget http://www.cs.jhu.edu/~kevinduh/t/letor.tgz
```

```
kduh@a14:~/ltr$ tar -xvf letor.tgz
```

Step 1: Understand the data format

Format: relevance label, query id, features+

```
0 qid:1 1:1.000000 2:1.000000 3:0.833333 4:0.871264 5:0 6:0 7:0 8:0.941842 9:1.000000 10:1.000000 11:1.000000 12:1.000000 13:1.000000 14:1.000000 15:1.000000 16:1.000000 17:1.000000 18:0.719697 19:0.729351 20:0 21:0 22:0 23:0.811565 24:1.000000 25:1.000000 26:1.000000 27:1.000000 28:0.922374 29:0.946654 30:0.938888 31:1.000000 32:1.000000 33:0.711276 34:0.722202 35:1.000000 36:1.000000 37:1.000000 38:1.000000 39:1.000000 40:1.000000 41:1.000000 42:1.000000 43:0.959134 44:0.963919 45:0.971425 #docid = 244338
2 qid:1 1:0.600000 2:0.600000 3:1.000000 4:1.000000 5:0 6:0 7:0 8:1.000000 9:0.624834 10:0.767301 11:0.811565 12:0.811565 13:0.811565 14:0.680222 15:0.686762 16:0.421053 17:0.680904 18:1.000000 19:1.000000 20:0 21:0 22:0 23:1.000000 24:1.000000 25:1.000000 26:1.000000 27:0.949446 28:0.984769 29:0.955266 30:0.997786 31:0.441860 32:0.687033 33:1.000000 34:1.000000 35:1.000000 36:1.000000 37:1.000000 38:1.000000 39:0.425450 40:0.975968 41:0.928785 42:0.978524 43:0.979553 44:1.000000 45:1.000000 #docid = 143821
```

- Questions:
 - How many documents in train.txt?
 - How many queries?
 - How many documents/query?

Step 2: Try out existing ranker

- SVMrank:

https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

```
kduh@a14:~/ltr$ mkdir svm_rank
kduh@a14:~/ltr$ cd svm_rank/
kduh@a14:~/ltr/svm_rank$ wget http://download.joachims.org/svm_rank/current/svm_rank.tar.gz
```

```
make: *** No targets specified and no makefile found.
kduh@a14:~/ltr/svm_rank$ tar -xvf svm_rank.tar.gz
```

```
kduh@a14:~/ltr/svm_rank$ make
```

- HELP: `svm_rank_learn -?`
- TRAIN: `svm_rank_learn -c 20 data model`

```
kduh@a14:~/ltr$ ./svm_rank/svm_rank_learn -c 20 letor/train.txt model1
```

Step 3. Evaluate

INFERENCE: svm_rank_classify data model output

```
kduh@a14:~/ltr$ ./svm_rank/svm_rank_classify letor/vali.txt model1 vali.pred
Reading model...done.
Reading test examples...done.
Classifying test examples...done
Runtime (without IO) in cpu-seconds: 0.00
Average loss on test set: 0.3840
Zero/one-error on test set: 100.00% (0 correct, 21 incorrect, 21 total)
NOTE: The loss reported above is the fraction of swapped pairs averaged over
      all rankings. The zero/one-error is fraction of perfectly correct
      rankings!
Total Num Swappedpairs : 51224
Avg Swappedpairs Percent: 38.40
```

Eval script

```
kduh@a14:~/ltr$ perl letor/Eval-Score-3.0.pl letor/vali.txt vali.pred vali.pred.result 1
kduh@a14:~/ltr$ grep MAP vali.pred.result
MAP: 0.458439033060933
```

Step 4

- Implement your own:
 - point-wise method with linear regression, using `sklearn.linear_model.LinearRegression`
 - Implement pair-wise method with binary classification, using `sklearn.linear_model.LogisticRegression`
- Learn more about existing toolkits:
 - **SVMrank**: https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
 - **RankLib**: <https://people.cs.umass.edu/~vdang/ranklib.html>
- Evaluate and compare MAP results