CSCE 670 - Information Storage and Retrieval

Week 5: Learning to Rank

Yu Zhang

yuzhang@tamu.edu

Course Website: https://yuzhang-teaching.github.io/CSCE670-S26.html

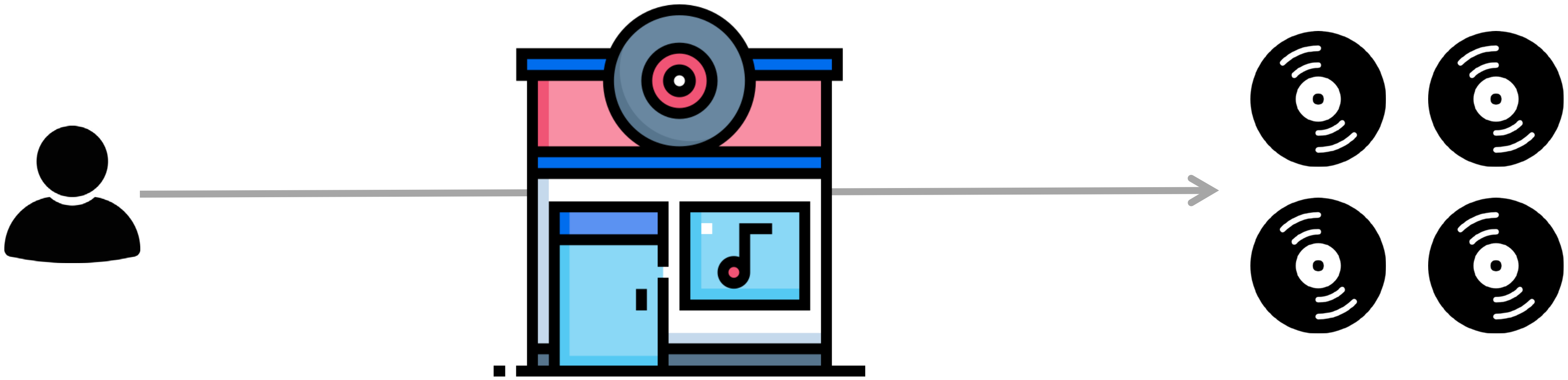
Adapted from the slides by Prof. James Caverlee

Ranking should consider multiple factors

- YouTube videos: view, subscribers, video length, user profile factors (e.g., age, location), title relevance, video quality, recency, …

- LinkedIn job postings: posting popularity, company popularity, number of openings, skill match with the user, nearness, recency, salary, …

- Our record store: record popularity, singer popularity, language, keyword match, …



Hand-tuning a Ranking Function

- Score(q, d) =

  $a_1 \times$ TF-IDF(q, d) +

  $a_2 \times$ BM25(q, d) +

  $a_3 \times$ # views in the last day(d) +

  $a_4 \times$ # views in the last week(d) +

  $a_5 \times$ recency(d) +

  $a_6 \times$ PageRank(d) +

  $\ldots$

- After checking some examples, you set a_1 as 0.5, a_2 as 0.8, $\ldots$
- Problems with this strategy?

Instead, let's learn a good ranker!

- Rough Idea (not 100% accurately framed): Learn the value of $a_1, a_2, \ldots$ from data (e.g., relevant query-document pairs according to user clickthrough history)
  - A very natural idea (especially these days)

- But it took a while for ML and IR to be good friends
  - Wong et al., *Linear structure in information retrieval.* SIGIR 1988.
  - Fuhr, *Probabilistic methods in information retrieval.* Computer Journal 1992.
  - Gey, *Inferring probability of relevance using the method of logistic regression.* SIGIR 1994.
  - Herbrich et al., *Large margin rank boundaries for ordinal regression.* Advances in Large Margin Classifiers 2000.

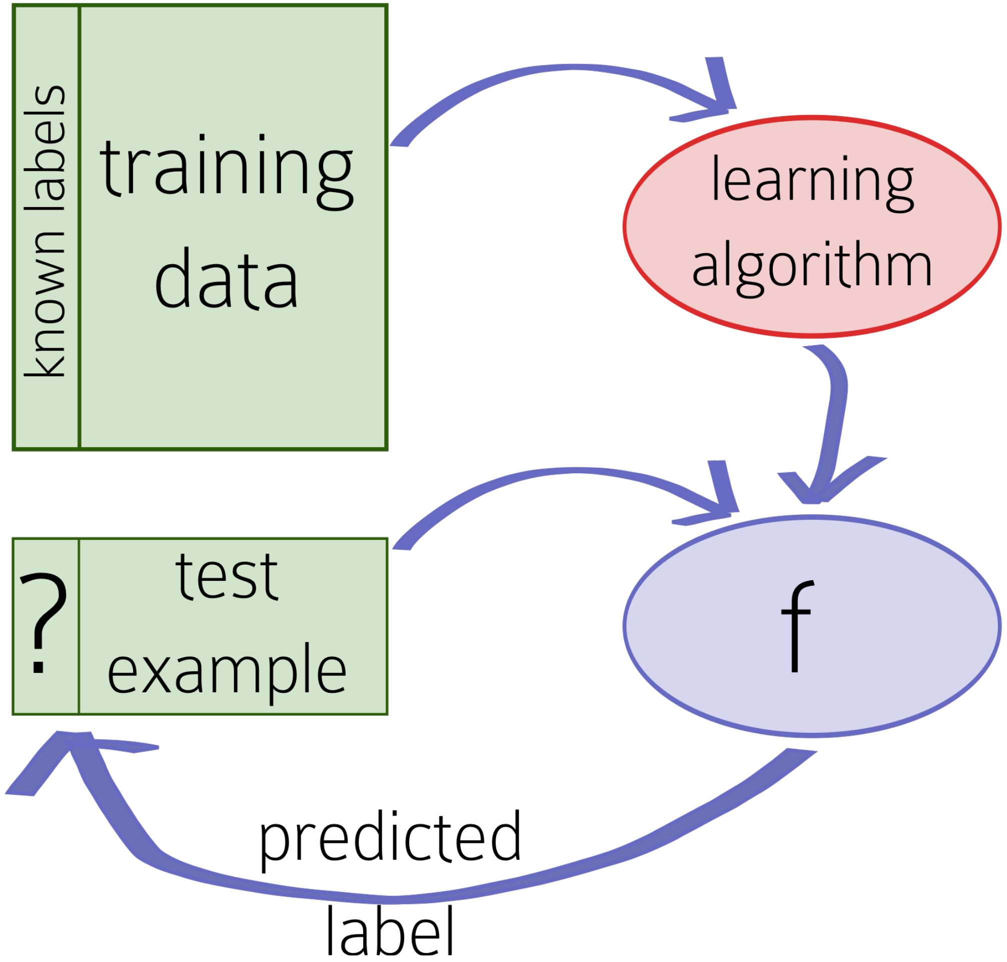
Background: Text Classification

- Given:
  - A document space $\mathcal{X}$
  - A fixed set of classes $\mathcal{C} = \{c_1, c_2, \dots\}$
  - A training set of labeled documents:
    - E.g., $d_1 \rightarrow c_1, \quad d_2 \rightarrow c_1, \quad d_3 \rightarrow c_2, \quad \dots$
- Use a learning algorithm to learn a classifier f that maps documents to classes $\mathrm{f}: \mathcal{X} \rightarrow \mathcal{C}$
- Examples
  - Paper Topic Classification: $\mathcal{X}$ = academic papers, $\mathcal{C}$ = {math, physics, chemistry, …}
  - Review Sentiment Analysis: $\mathcal{X}$ = food reviews, $\mathcal{C}$ = {1-star, 2-star, 3-star, 4-star, 5-star}
  - Songwriter Prediction: $\mathcal{X}$ = lyrics, $\mathcal{C}$ = songwriters

Background: Text Classification

- Training: Use a learning algorithm to learn a classifier f that maps documents to classes f: $\mathcal{X} \rightarrow \mathcal{C}$

- Testing/Inference: Given an unseen document d_{test}
  - Apply our classifier function $\mathrm{f}(d_{\text{test}})$ to determine the most appropriate class in $\mathcal{C}$



A Couple of Simple Text Classifiers: Rocchio

- Training: "Learn" class centers for each class by finding the centroid of all the training examples from each class

- Testing/Inference: Assign a new example to the class of the nearest class center

- Example:
  - 2-class classification (*chemistry* paper vs. *history* paper)
  - Training samples
    - *chemistry*: $d_1 = (1.0, 0.9), d_2 = (0.9, 1.0)$
    - *history*: $d_3 = (0.2, 0.3), d_4 = (0.3, 0.2)$

A Couple of Simple Text Classifiers: Rocchio

- Example:
  - 2-class classification (*chemistry* paper vs. *history* paper)
  - Training samples
    - *chemistry*: $d_1 = (1.0, 0.9), d_2 = (0.9, 1.0)$
    - *history*: $d_3 = (0.2, 0.3), d_4 = (0.3, 0.2)$

- Step 1: Compute class centroids
  - *chemistry*: $c_{\text{chemistry}} = \frac{d_1+d_2}{2} = (0.95, 0.95)$
  - *history*: $c_{\text{history}} = \frac{d_3+d_4}{2} = (0.25, 0.25)$

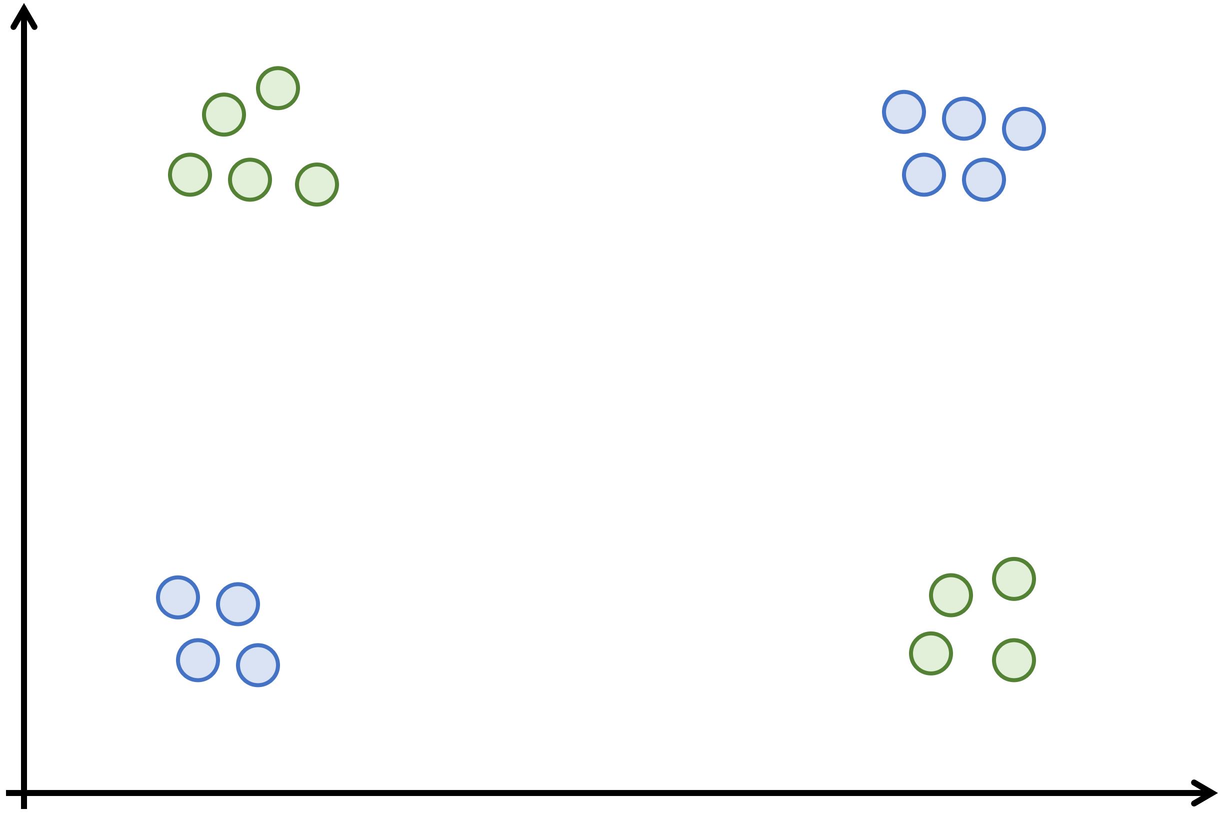
A Couple of Simple Text Classifiers: Rocchio

- Step 1: Compute class centroids
  - *chemistry*: $c_{\text{chemistry}} = \frac{d_1+d_2}{2} = (0.95, 0.95)$
  - *history*: $c_{\text{history}} = \frac{d_3+d_4}{2} = (0.25, 0.25)$

- Step 2: Classify a new document
  - New document: $d_5 = (0.8, 0.85)$
  - Compute Euclidean distance:
    - To *chemistry*: $\text{dist}(d_5, c_{\text{chemistry}}) = \sqrt{(0.8-0.95)^2+(0.85-0.95)^2} \approx 0.1803$
    - To *history*: $\text{dist}(d_5, c_{\text{history}}) = \sqrt{(0.8-0.25)^2+(0.85-0.25)^2} \approx 0.8124$
  - d_5 is closer to *chemistry*, so we classify it as a *chemistry* paper.

A Couple of Simple Text Classifiers: Rocchio

- Can you raise an example where Rocchio does NOT work?



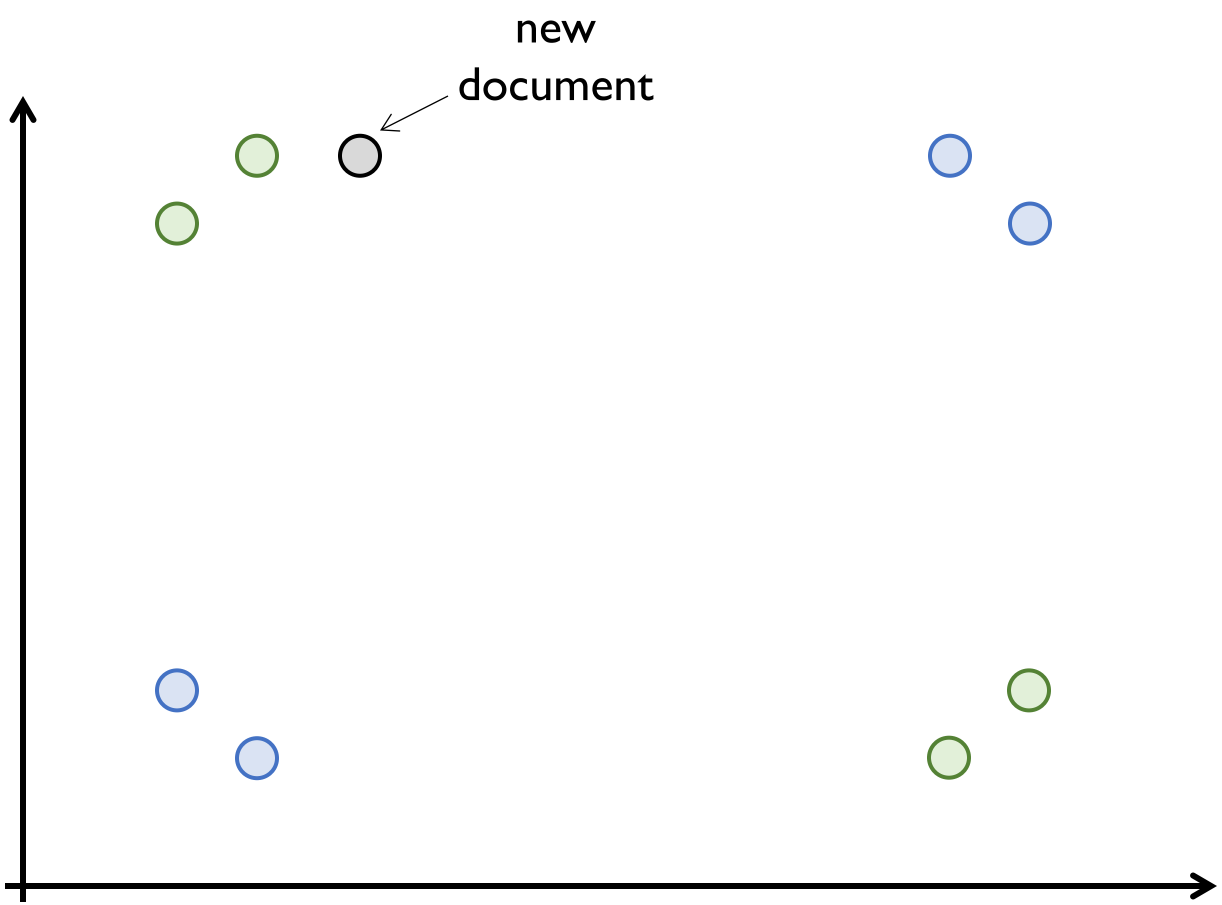
A Couple of Simple Text Classifiers: k Nearest Neighbors (kNN)

- Training: None
- Testing/Inference: Assign a new example to the majority class of the k-nearest training examples
- Example:
  - 2-class classification (*chemistry* paper vs. *history* paper)
  - Training samples
    - *chemistry*: $d_1 = (1.0, 0.9), d_2 = (0.9, 1.0), d_3 = (0.2, 0.3), d_4 = (0.3, 0.2)$
    - *history*: $d_5 = (1.0, 0.3), d_6 = (0.9, 0.2), d_7 = (0.3, 1.0), d_8 = (0.2, 0.9)$
  - New document: $d_9 = (0.4, 1.0)$
  - $k = 3$

A Couple of Simple Text Classifiers: k Nearest Neighbors (kNN)

- Nearest neighbor: *history*
- 2nd nearest neighbor: *history*
- 3rd nearest neighbor: *chemistry*

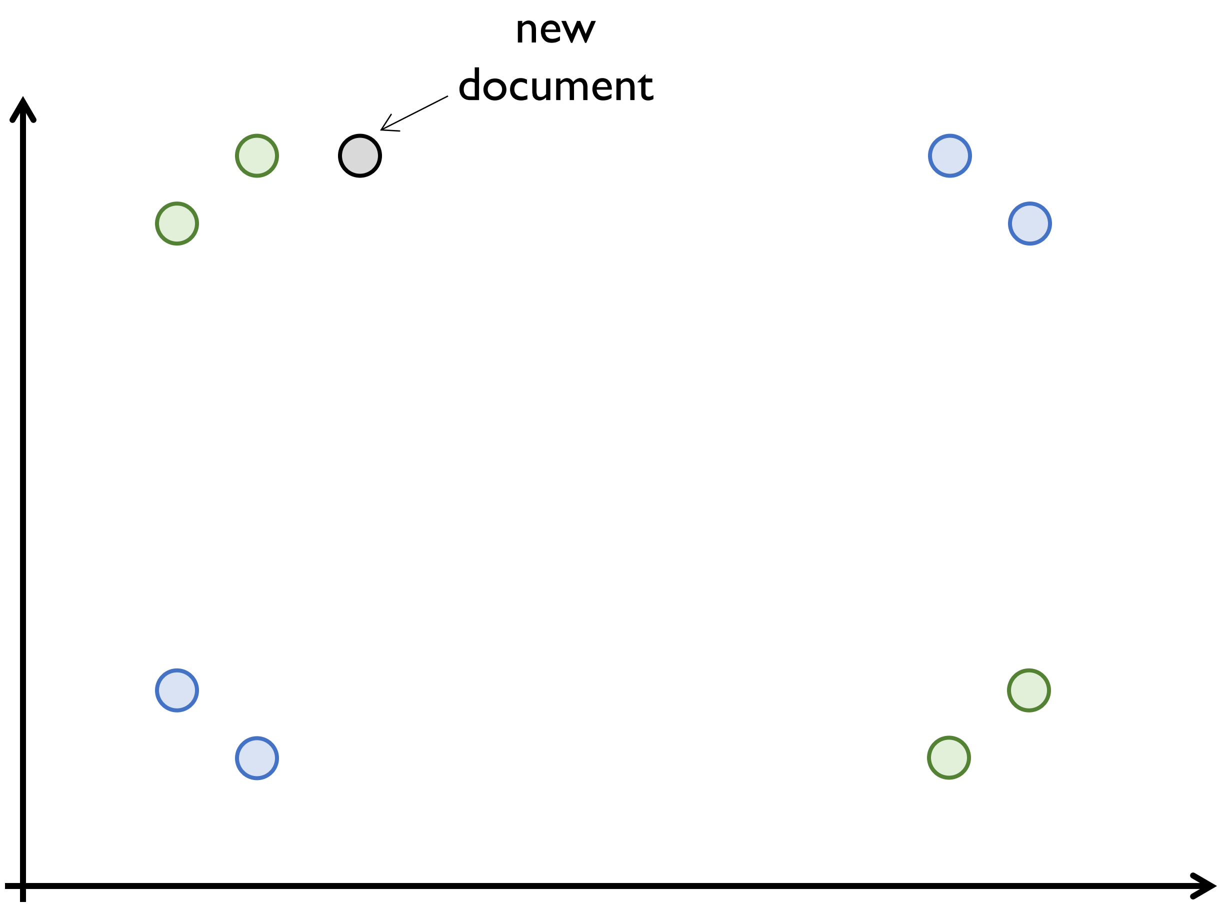
- Majority voting: the new document has more *history* neighbors, so we classify it as a *history* paper.



A Couple of Simple Text Classifiers: k Nearest Neighbors (kNN)

- Can you raise an example where kNN does NOT work?

- How to determine k? What if $k = 5$?
- Nearest neighbor: *history*
- 2nd nearest neighbor: *history*
- 3rd nearest neighbor: *chemistry*
- 4th nearest neighbor: *chemistry*
- 5th nearest neighbor: *chemistry*



In practice: Which features?

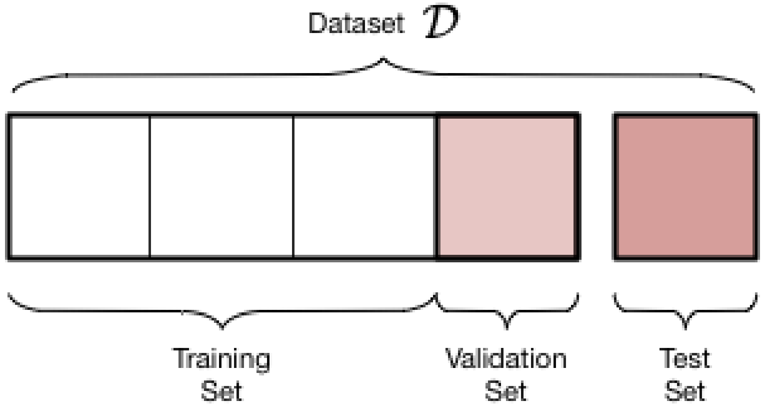
- Very important to select good features to represent our documents

- Features we know about:
  - TF-IDF score of each word (one feature per word)
  - PageRank/Hub/Authority score of the document
  - Popularity, # of clicks, freshness, …

In practice: Which classifier?

- Many, many ways to learn a good classifier
  - Rocchio
  - kNN
  - Support Vector Machine
  - Naive Bayes
  - Decision Tree
  - Random Forest
  - Gradient-Boosted Decision Tree
  - …

In practice: How to evaluate?



- Need a way to evaluate how well we do
- Classification accuracy is one way
  - For a held-out test set (for which we know the correct labels), calculate how many labels our classifier correctly predicts
- Many others (some we may talk about later)

- Keep part of the labeled data separate as a validation set
- Train a model over the training data and "test" over the validation set
- Train another model over the training data and "test" over the validation set (and so on and so on)
- Choose model that minimizes error on the validation set

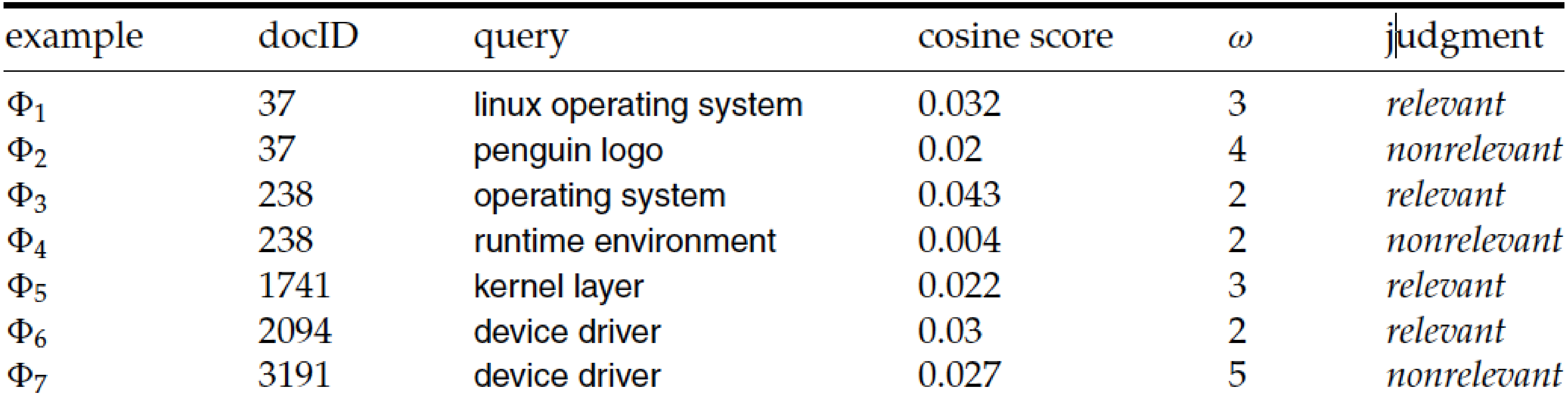
Back to Ranking

- Assume we have a test collection:
  - A benchmark document collection
  - A benchmark suite of queries
  - A binary assessment of either Relevant or Non-relevant for each query and each document

- Sounds like classification!
  - Classification Training: Given a training set of (*query*, *document* → *relevance*) triples, learn a model f that outputs Relevant or Non-relevant
  - Classification Testing: Given unseen (*query*, *document*), apply f(*query*, *document*) and output Relevant or Non-relevant
  - NOTE: Now our input is not just a *document* but both a *document* and a *query*!

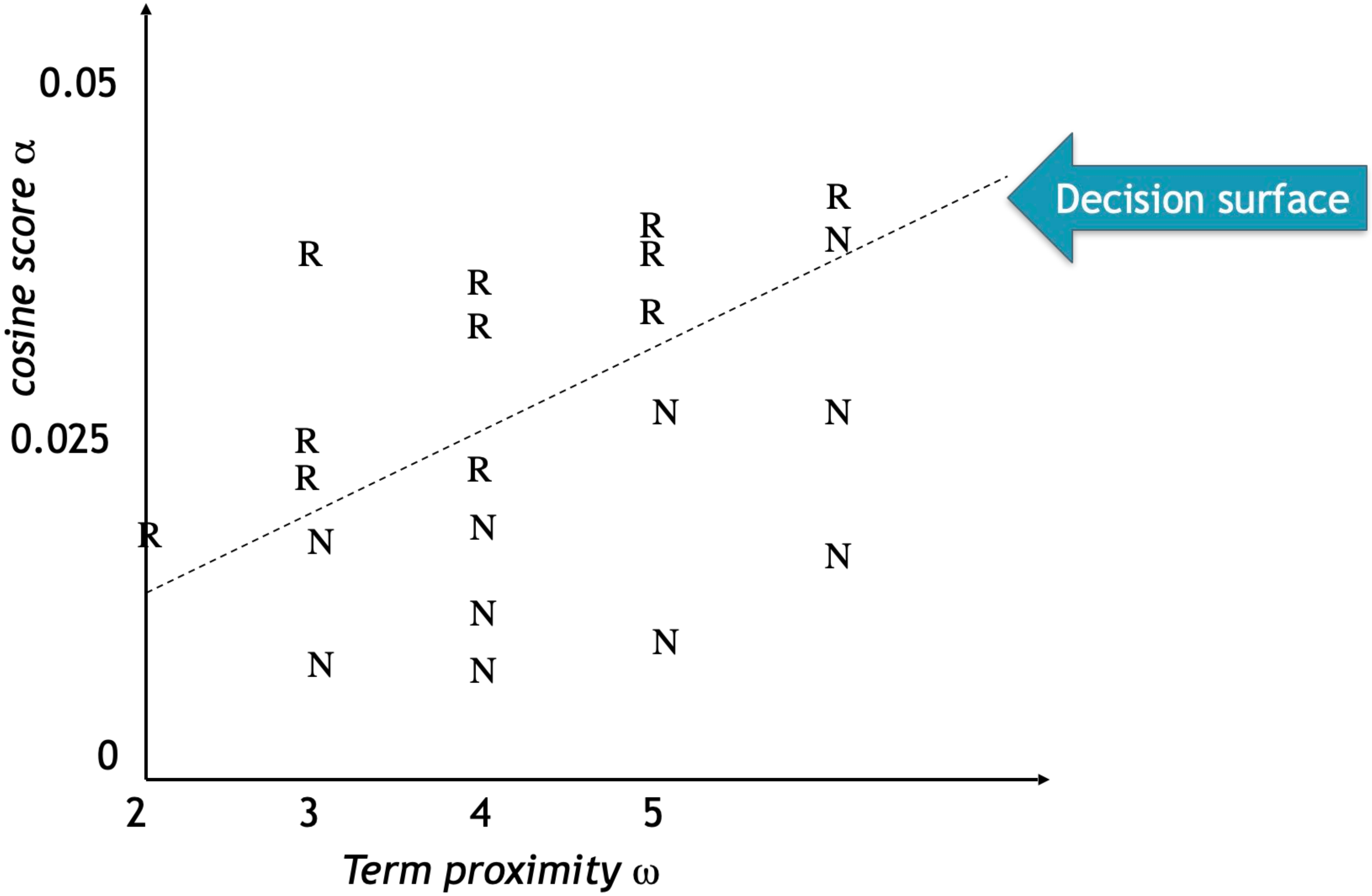
Relevance Classification: Example

term proximity

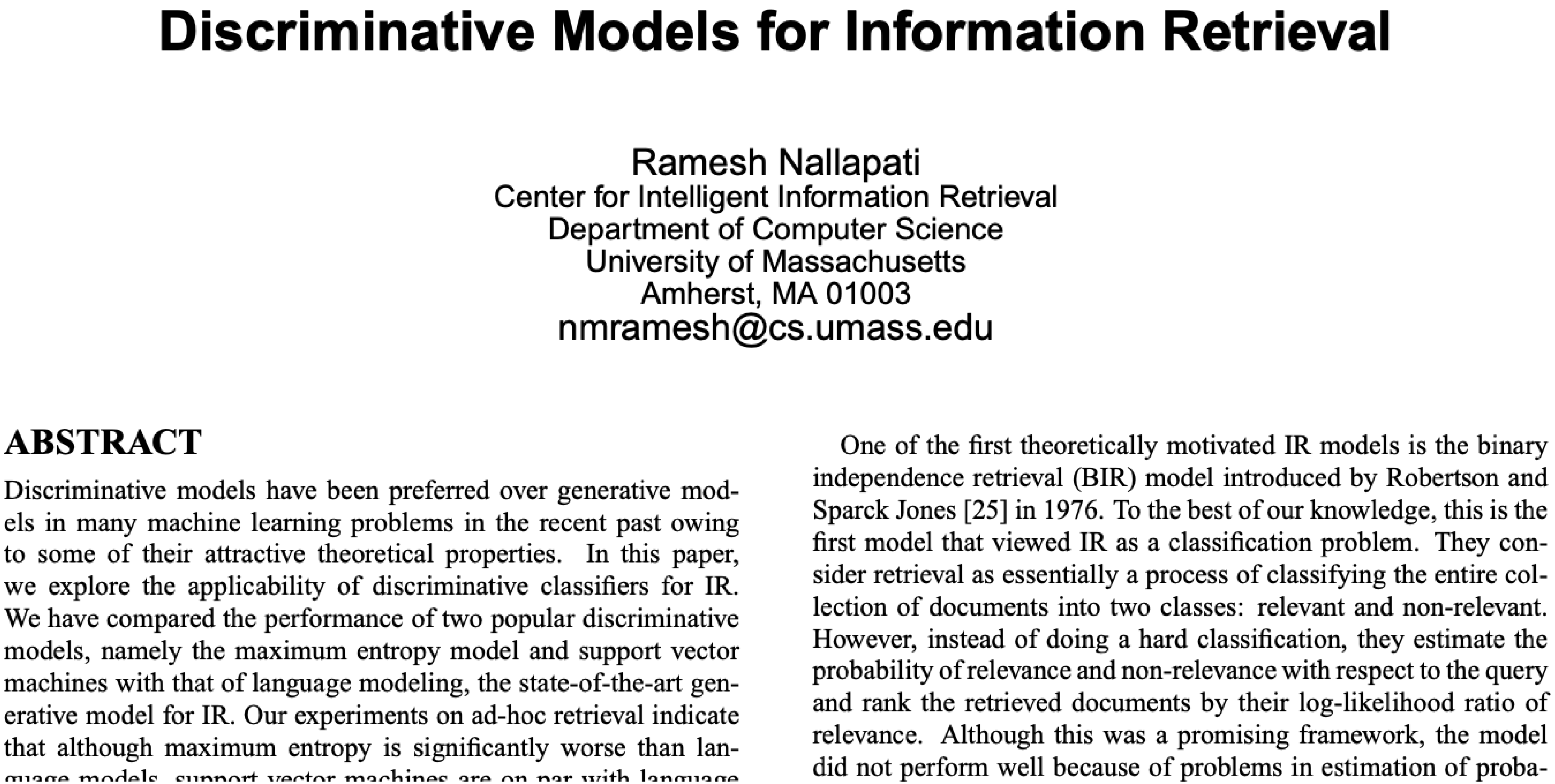


example	docID	query	cosine score	ω	judgment
Φ_1	37	linux operating system	0.032	3	*relevant*
Φ_2	37	penguin logo	0.02	4	*nonrelevant*
Φ_3	238	operating system	0.043	2	*relevant*
Φ_4	238	runtime environment	0.004	2	*nonrelevant*
Φ_5	1741	kernel layer	0.022	3	*relevant*
Φ_6	2094	device driver	0.03	2	*relevant*
Φ_7	3191	device driver	0.027	5	*nonrelevant*

Relevance Classification: Example



[Nallapati, SIGIR 2004]



Discriminative Models for Information Retrieval

Ramesh Nallapati
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
nmramesh@cs.umass.edu

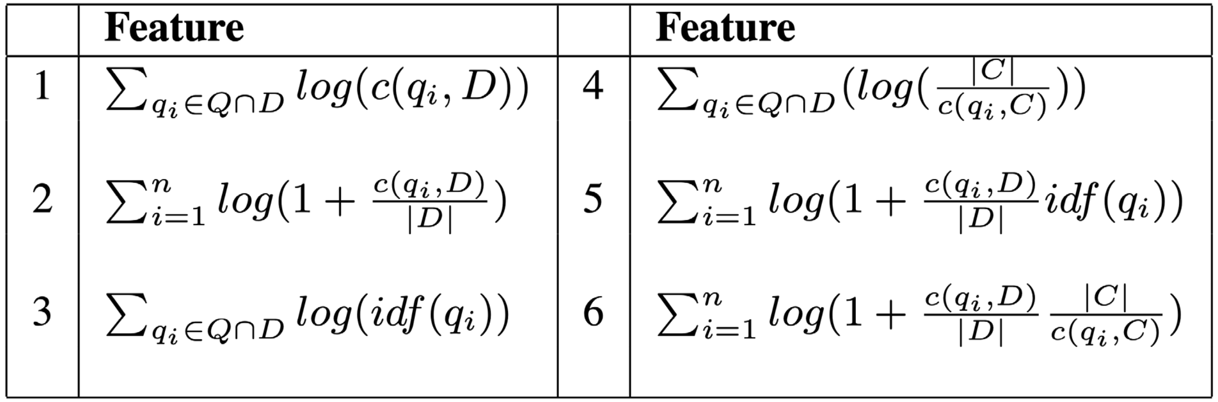
ABSTRACT

Discriminative models have been preferred over generative models in many machine learning problems in the recent past owing to some of their attractive theoretical properties. In this paper, we explore the applicability of discriminative classifiers for IR. We have compared the performance of two popular discriminative models, namely the maximum entropy model and support vector machines with that of language modeling, the state-of-the-art generative model for IR. Our experiments on ad-hoc retrieval indicate that although maximum entropy is significantly worse than language models, support vector machines are on par with language

One of the first theoretically motivated IR models is the binary independence retrieval (BIR) model introduced by Robertson and Sparck Jones [25] in 1976. To the best of our knowledge, this is the first model that viewed IR as a classification problem. They consider retrieval as essentially a process of classifying the entire collection of documents into two classes: relevant and non-relevant. However, instead of doing a hard classification, they estimate the probability of relevance and non-relevance with respect to the query and rank the retrieved documents by their log-likelihood ratio of relevance. Although this was a promising framework, the model did not perform well because of problems in estimation of proba-

[Nallapati, SIGIR 2004]

- Experiments:
  - Comparisons with Lemur (LM), a state-of-the-art open-source IR engine
  - Which classifier? SVM with linear kernel
  - What features? 6 features, all variants of TF, IDF, and TF-IDF scores

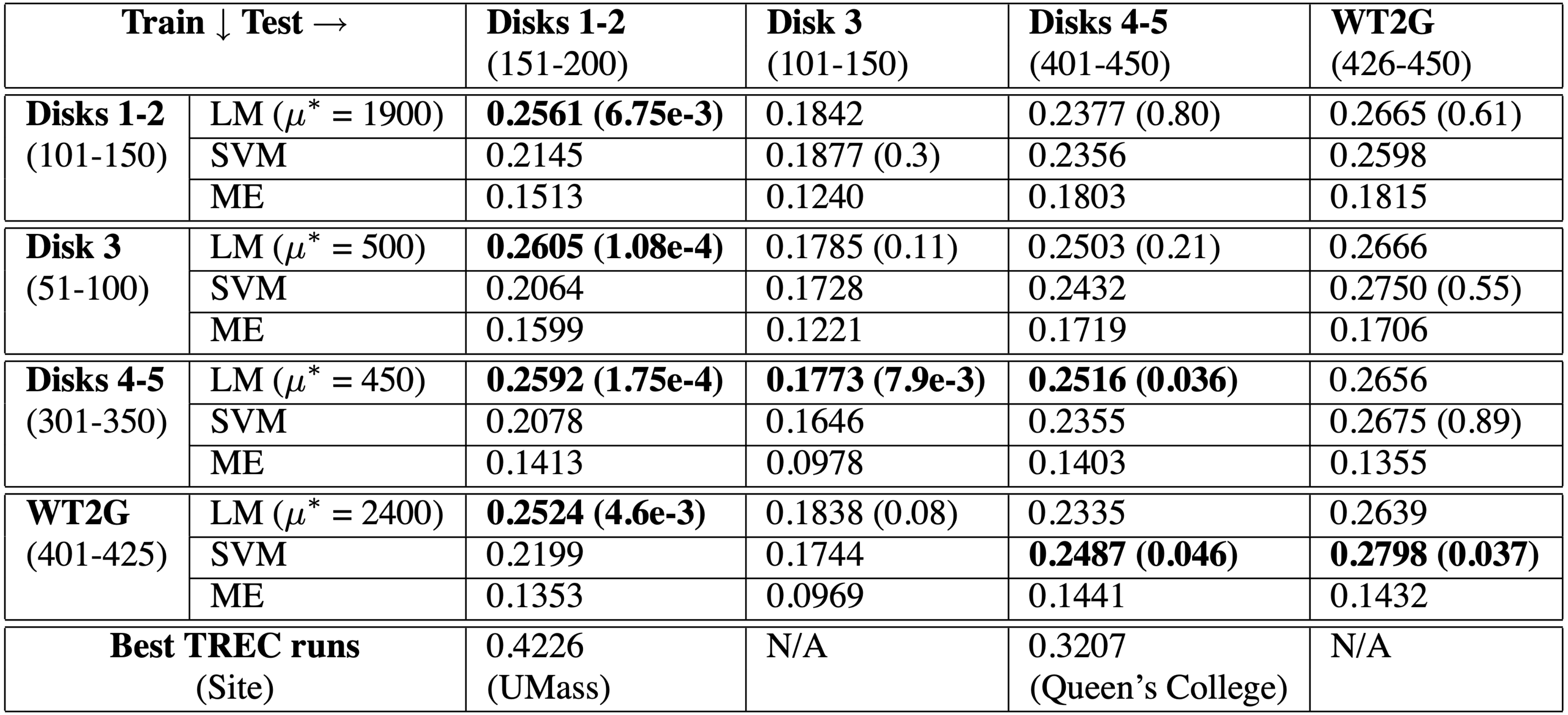


	Feature		**Feature**
1	$\sum_{q_i \in Q \cap D} log(c(q_i, D))$	4	$\sum_{q_i \in Q \cap D} (log(\frac{\lvert C \rvert}{c(q_i, C)}))$
2	$\sum_{i=1}^{n} log(1 + \frac{c(q_i, D)}{\lvert D \rvert})$	5	$\sum_{i=1}^{n} log(1 + \frac{c(q_i, D)}{\lvert D \rvert} idf(q_i))$
3	$\sum_{q_i \in Q \cap D} log(idf(q_i))$	6	$\sum_{i=1}^{n} log(1 + \frac{c(q_i, D)}{\lvert D \rvert} \frac{\lvert C \rvert}{c(q_i, C)})$

Figure 2: Features in the discriminative models: $c(w, D)$ represents the raw count of word w in document D, C represents the collection, n is the number of terms in the query, $|.|$ is the *size-of* **function and $idf(.)$ is the inverse document frequency.**

Experiments on 4 TREC Datasets

- Metric: Mean Average Precision (MAP)



Train ↓ Test →		**Disks 1-2** (151-200)	**Disk 3** (101-150)	**Disks 4-5** (401-450)	**WT2G** (426-450)
Disks 1-2 (101-150)	LM (μ^* = 1900)	**0.2561 (6.75e-3)**	0.1842	0.2377 (0.80)	0.2665 (0.61)
	SVM	0.2145	0.1877 (0.3)	0.2356	0.2598
	ME	0.1513	0.1240	0.1803	0.1815
Disk 3 (51-100)	LM (μ^* = 500)	**0.2605 (1.08e-4)**	0.1785 (0.11)	0.2503 (0.21)	0.2666
	SVM	0.2064	0.1728	0.2432	0.2750 (0.55)
	ME	0.1599	0.1221	0.1719	0.1706
Disks 4-5 (301-350)	LM (μ^* = 450)	**0.2592 (1.75e-4)**	**0.1773 (7.9e-3)**	**0.2516 (0.036)**	0.2656
	SVM	0.2078	0.1646	0.2355	0.2675 (0.89)
	ME	0.1413	0.0978	0.1403	0.1355
WT2G (401-425)	LM (μ^* = 2400)	**0.2524 (4.6e-3)**	0.1838 (0.08)	0.2335	0.2639
	SVM	0.2199	0.1744	**0.2487 (0.046)**	**0.2798 (0.037)**
	ME	0.1353	0.0969	0.1441	0.1432
Best TREC runs (Site)		0.4226 (UMass)	N/A	0.3207 (Queen's College)	N/A

Experiments on 4 TREC Datasets

- At best the results are about equal to Lemur
  - Actually a little bit below

- Paper's advertisement: Easy to add more features
- This is illustrated on a homepage finding task on WT10G:

	Success@10
Lemur	0.52
SVM with text features only	0.58
SVM with URL-depth and in-link features	0.78

Questions?

But Boolean ≠ Ranking!

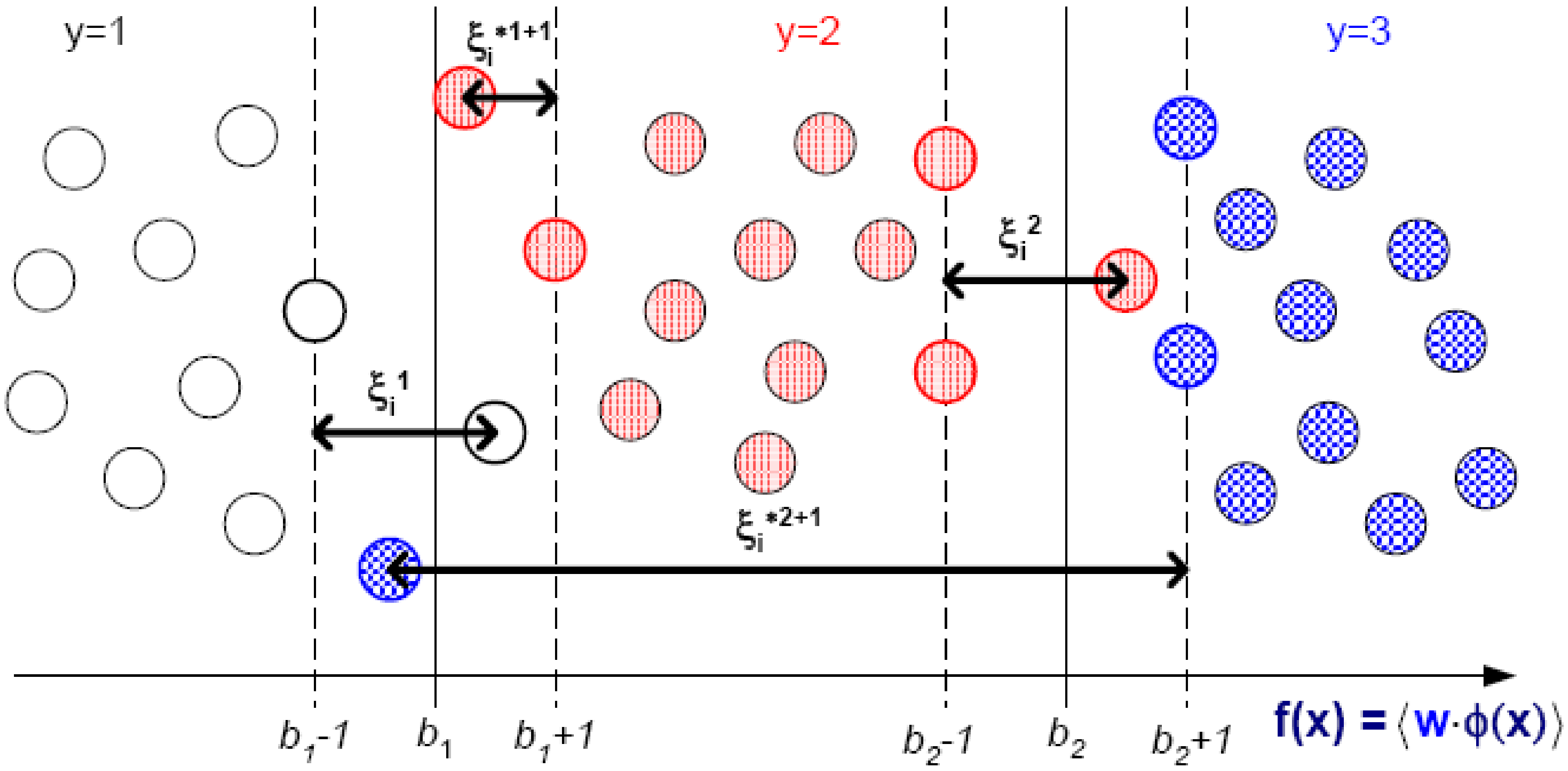
- Assigning (*query*, *document*) to
  - Relevant
  - or
  - Not Relevant
- Is not really what we want when we think about ranking

Pointwise Learning

- Assume we have training data like (*query*, *document*, *score*)
- Here the *score* could be a relevance score like
  - 4 Perfect match
  - 3 Very relevant
  - 2 Relevant
  - 1 Somewhat relevant
  - 0 Not relevant at all
- Our goal is to output a score
  - This is regression (if we can output any value)
  - Or ordinal regression (If we can only output 0, 1, 2, 3, or 4)

Pointwise Learning

- What could be a difference between classification and ordinal regression?



Pointwise Learning

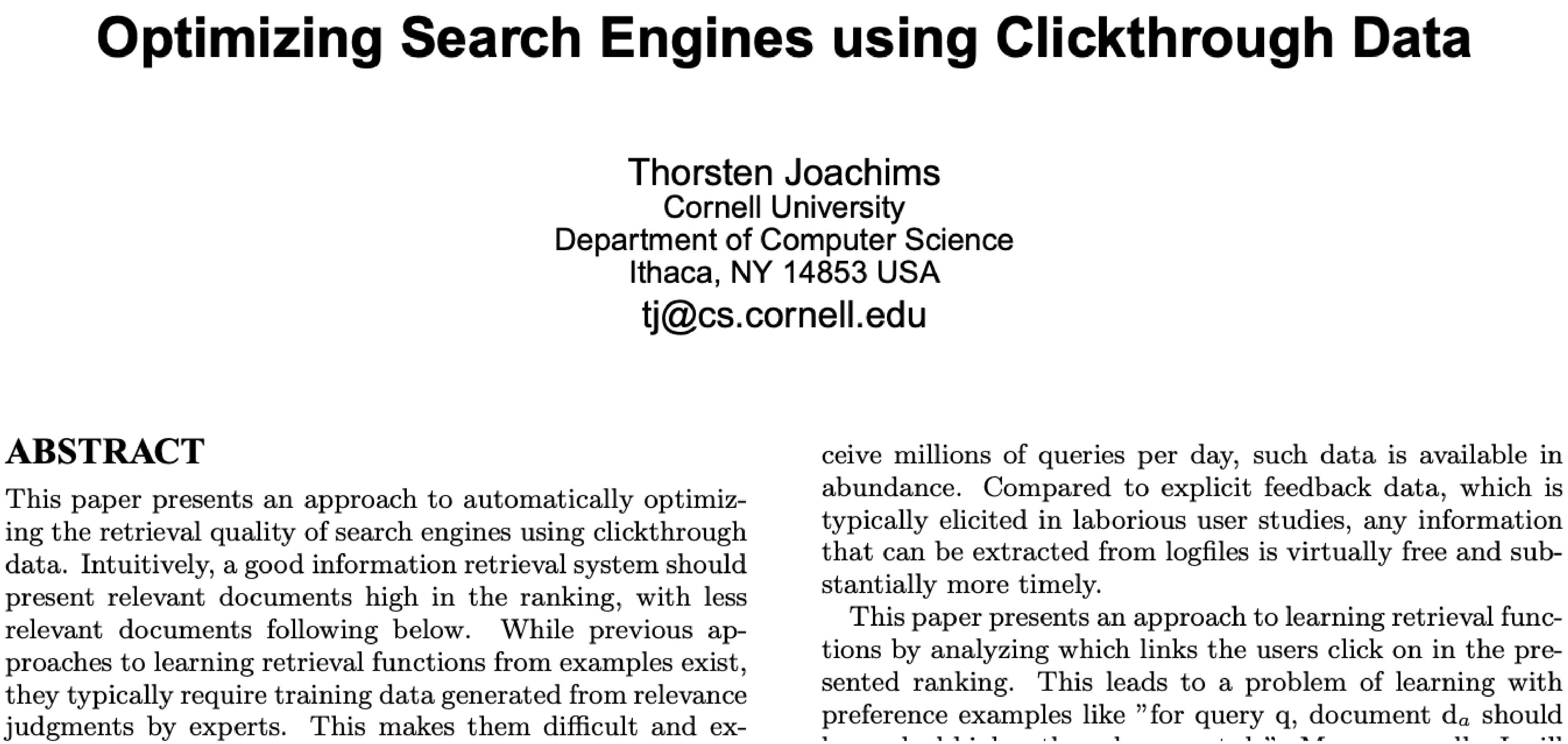
- Regression Training: Given a training set of (*query*, *document* → *score*) triples, learn a model f
- Regression Testing: Given unseen (*query*, *document*), apply f(*query*, *document*) and output the *score*

- Challenges?
  - Expensive to collect labels
  - Focuses on scores, not relative ordering (or relationship to other documents)
  - Bias towards frequent queries
  - …

Pairwise Learning

- Aim is to classify instance pairs as correctly ranked or incorrectly ranked
  - Given the query q and two candidates (c_i, c_k), predict if c_i should be ranked higher than c_k (denoted as $c_i > c_k$)
- This turns an ordinal regression problem back into a binary classification problem in an expanded space
  - We only need lots of (c_i, c_k), where we already know $c_i > c_k$, for training
- Formally, we want a ranking function f such that
  - $c_i > c_k \;\Leftrightarrow \mathrm{f}(\psi_i) > \mathrm{f}(\psi_k)$
  - ψ_i is the feature vector of c_i given the query q (e.g., one entry can be TF-IDF(q, c_i))
- To simplify our discussion, let's suppose that f is a linear function: $\mathrm{f}(\psi_i) = w^T \psi_i$

Ranking SVM [Joachims, KDD 2002]



Optimizing Search Engines using Clickthrough Data

Thorsten Joachims
Cornell University
Department of Computer Science
Ithaca, NY 14853 USA
tj@cs.cornell.edu

ABSTRACT

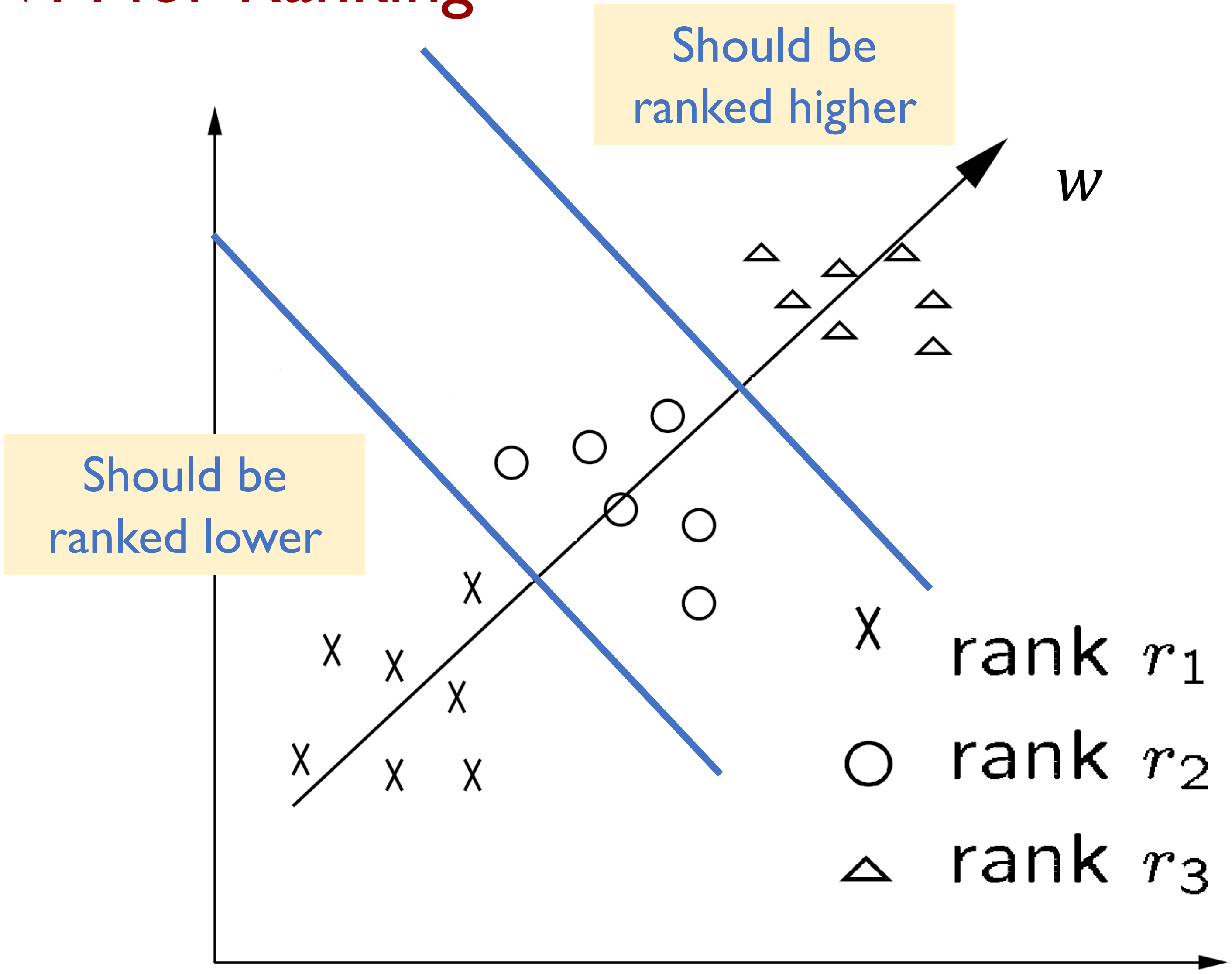
This paper presents an approach to automatically optimizing the retrieval quality of search engines using clickthrough data. Intuitively, a good information retrieval system should present relevant documents high in the ranking, with less relevant documents following below. While previous approaches to learning retrieval functions from examples exist, they typically require training data generated from relevance judgments by experts. This makes them difficult and ex-

ceive millions of queries per day, such data is available in abundance. Compared to explicit feedback data, which is typically elicited in laborious user studies, any information that can be extracted from logfiles is virtually free and substantially more timely.

This paper presents an approach to learning retrieval functions by analyzing which links the users click on in the presented ranking. This leads to a problem of learning with preference examples like "for query q, document d_a should

Training a Linear SVM for Ranking

Ranking function:
$f(\psi_i) = w^T \psi_i$



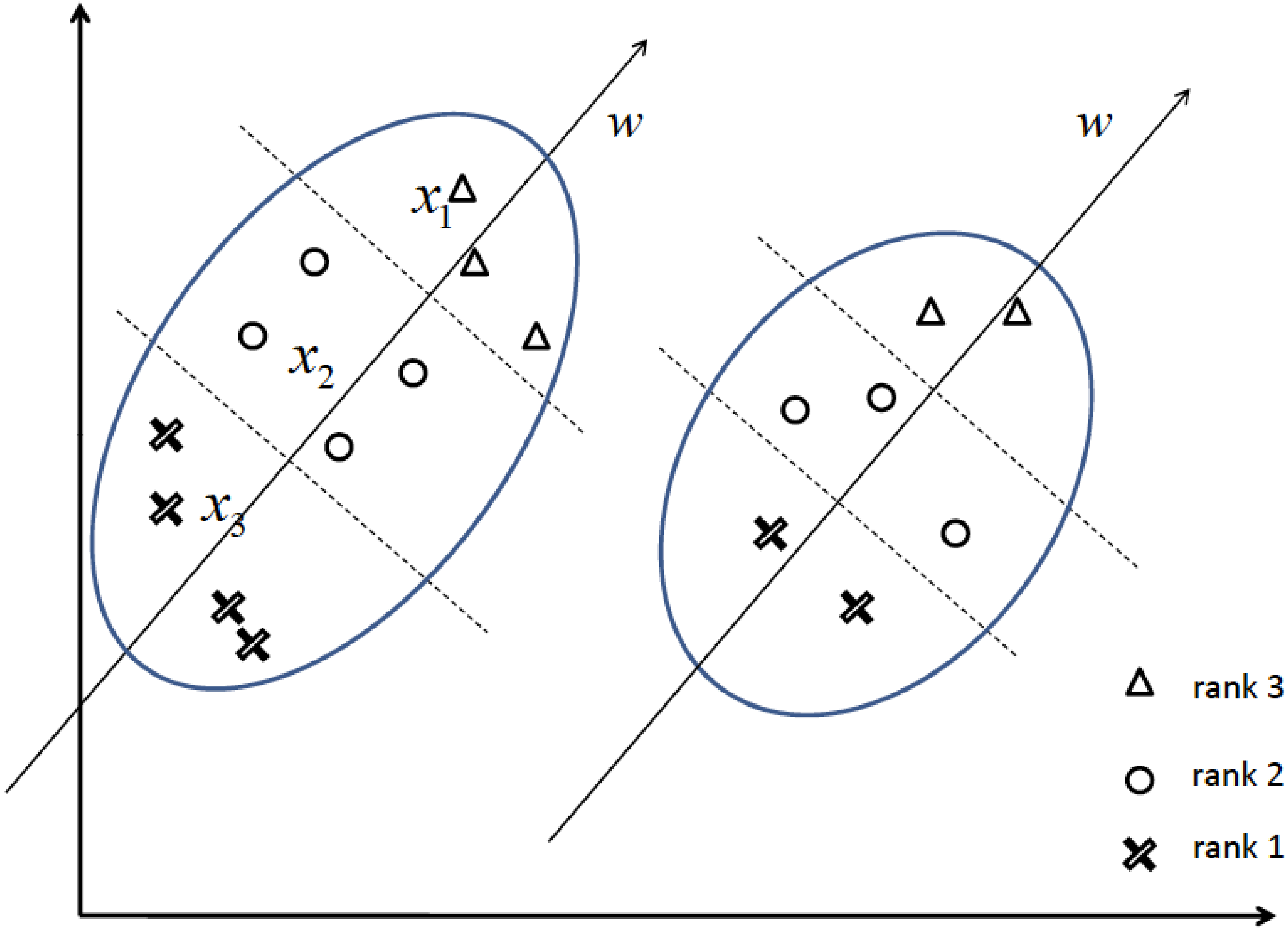
Should be ranked higher

Should be ranked lower

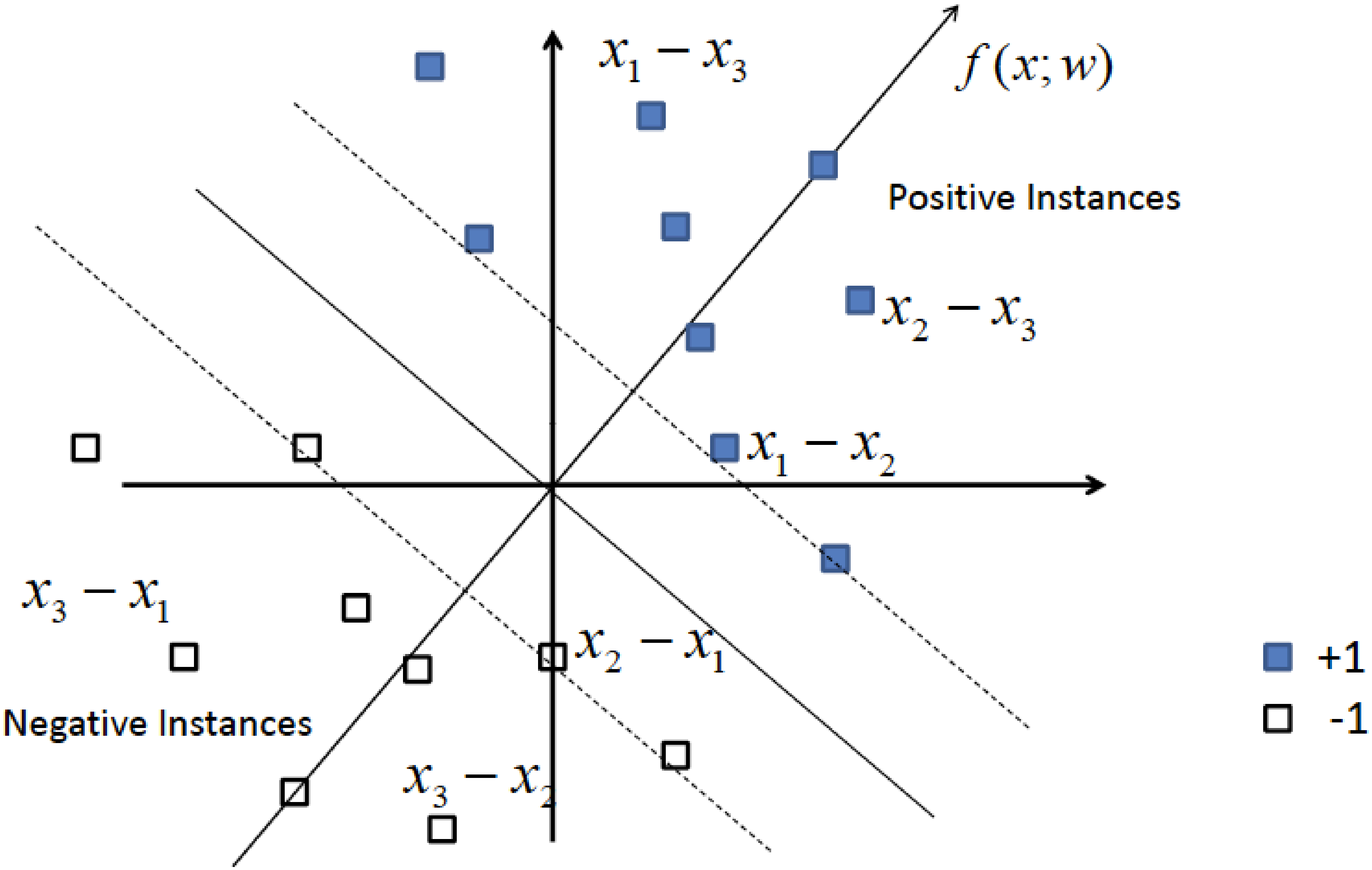
Training a Linear SVM for Ranking

- But we don't have pointwise training data!
  - Remember we only have lots of (c_i, c_k), where we already know $\mathrm{f}(\psi_i) > \mathrm{f}(\psi_k)$
  - We don't know the value of $\mathrm{f}(\psi_i)$ or $\mathrm{f}(\psi_k)$
- Idea: Create a new instance space from pairwise learning
  - We have $c_i > c_k \;\Leftrightarrow \mathrm{f}(\psi_i) > \mathrm{f}(\psi_k)$
  - We also have $\mathrm{f}(\psi_i) = w^T\psi_i$ and $\mathrm{f}(\psi_k) = w^T\psi_k$
  - So $c_i > c_k \;\Leftrightarrow w^T\psi_i > w^T\psi_k \Leftrightarrow w^T(\psi_i - \psi_k) > 0$
  - Let's create a new instance $\phi_u = \psi_i - \psi_k$
  - And $z_u = +1, 0, -1$ as $c_i >, =, < c_k$
  - From training data $\mathcal{S} = \{\phi_u\}$, we train an SVM

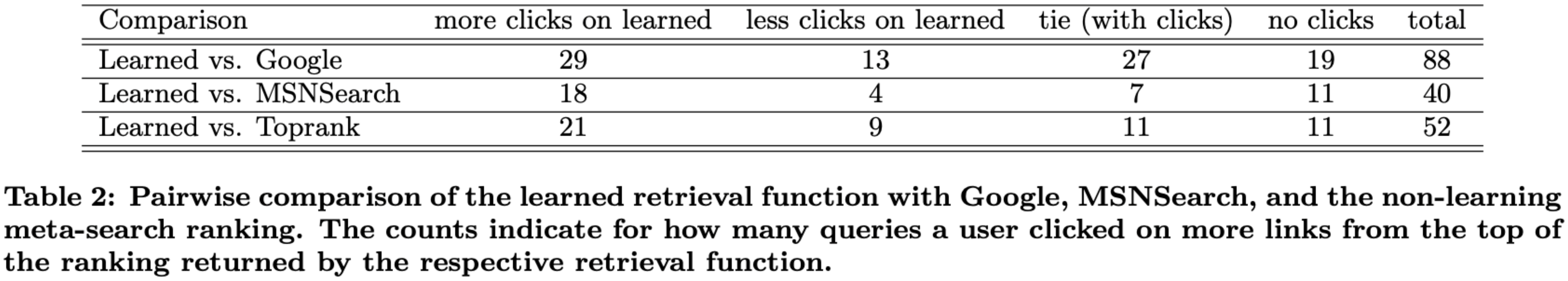
Two Queries in the Original Space



Two Queries in the Pairwise Space

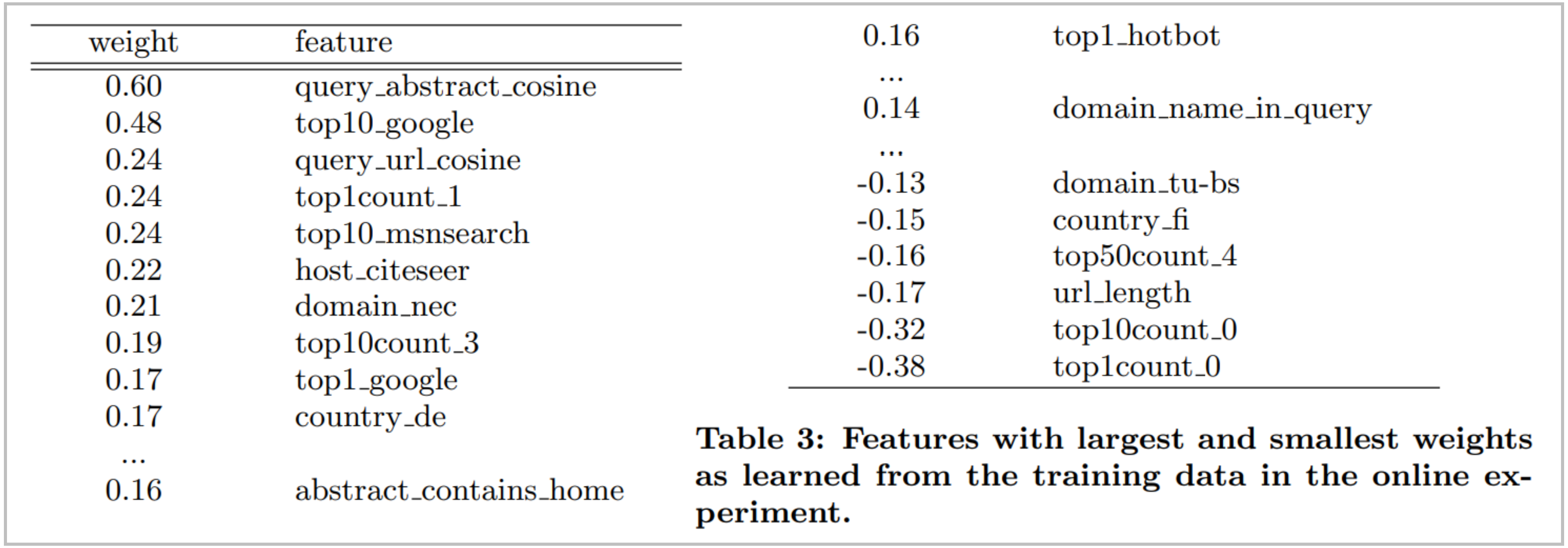


Performance of Ranking SVM



Comparison	more clicks on learned	less clicks on learned	tie (with clicks)	no clicks	total
Learned vs. Google	29	13	27	19	88
Learned vs. MSNSearch	18	4	7	11	40
Learned vs. Toprank	21	9	11	11	52

Table 2: Pairwise comparison of the learned retrieval function with Google, MSNSearch, and the non-learning meta-search ranking. The counts indicate for how many queries a user clicked on more links from the top of the ranking returned by the respective retrieval function.

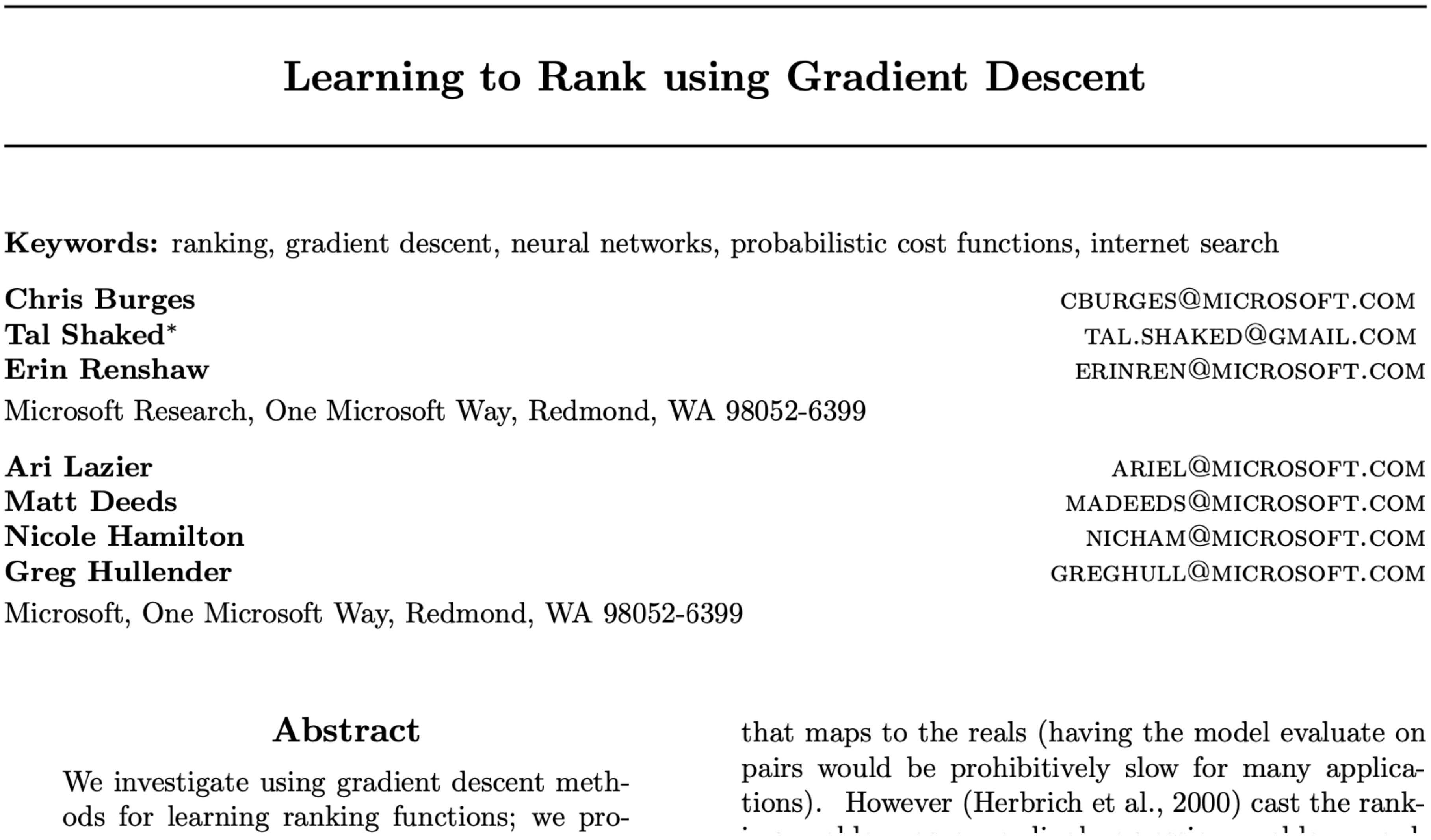


weight	feature
0.60	query_abstract_cosine
0.48	top10_google
0.24	query_url_cosine
0.24	top1count_1
0.24	top10_msnsearch
0.22	host_citeseer
0.21	domain_nec
0.19	top10count_3
0.17	top1_google
0.17	country_de
...	
0.16	abstract_contains_home
0.16	top1_hotbot
...	
0.14	domain_name_in_query
...	
-0.13	domain_tu-bs
-0.15	country_fi
-0.16	top50count_4
-0.17	url_length
-0.32	top10count_0
-0.38	top1count_0

Table 3: Features with largest and smallest weights as learned from the training data in the online experiment.

Questions?

RankNet [Burges et al., ICML 2005]



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

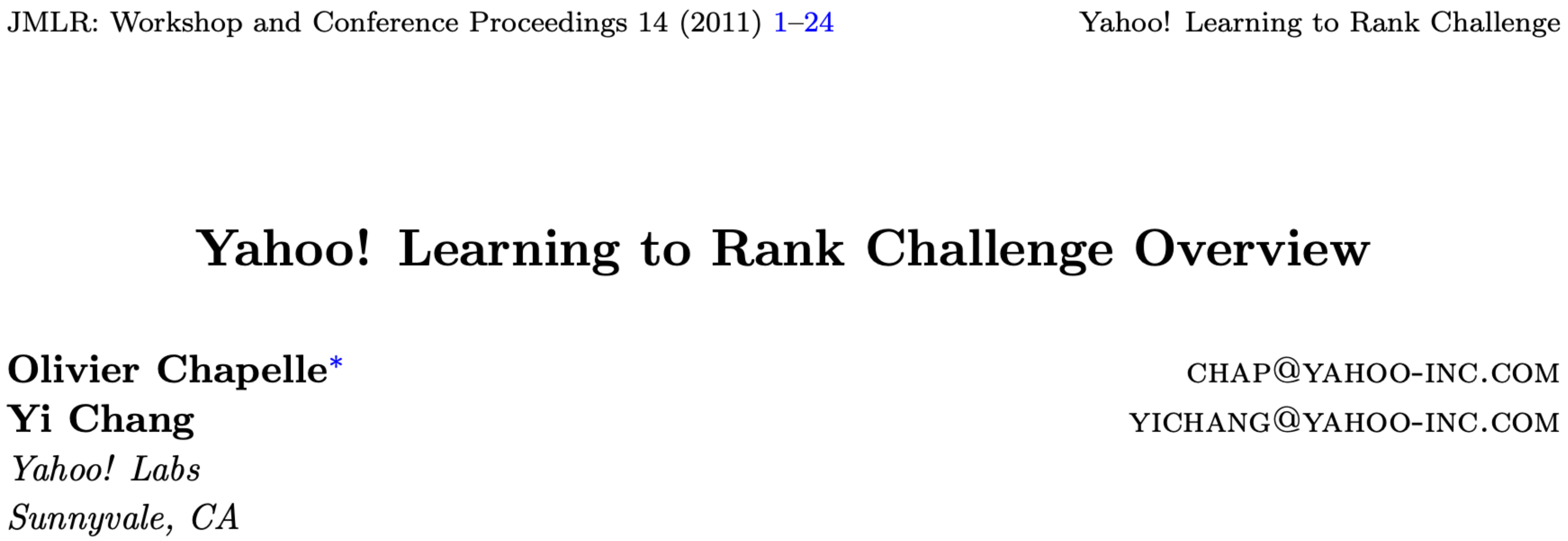
Abstract

We investigate using gradient descent methods for learning ranking functions; we pro-

that maps to the reals (having the model evaluate on pairs would be prohibitively slow for many applications). However (Herbrich et al., 2000) cast the rank-

RankNet [Burges et al., ICML 2005]

- Led to popular and successful variants:
  - LambdaRank
  - LambdaMART: top performer at the 2010 Yahoo Learning to Rank Challenge



JMLR: Workshop and Conference Proceedings 14 (2011) 1–24 Yahoo! Learning to Rank Challenge

Yahoo! Learning to Rank Challenge Overview

Olivier Chapelle* CHAP@YAHOO-INC.COM
Yi Chang YICHANG@YAHOO-INC.COM
Yahoo! Labs
Sunnyvale, CA

Existing Public Datasets

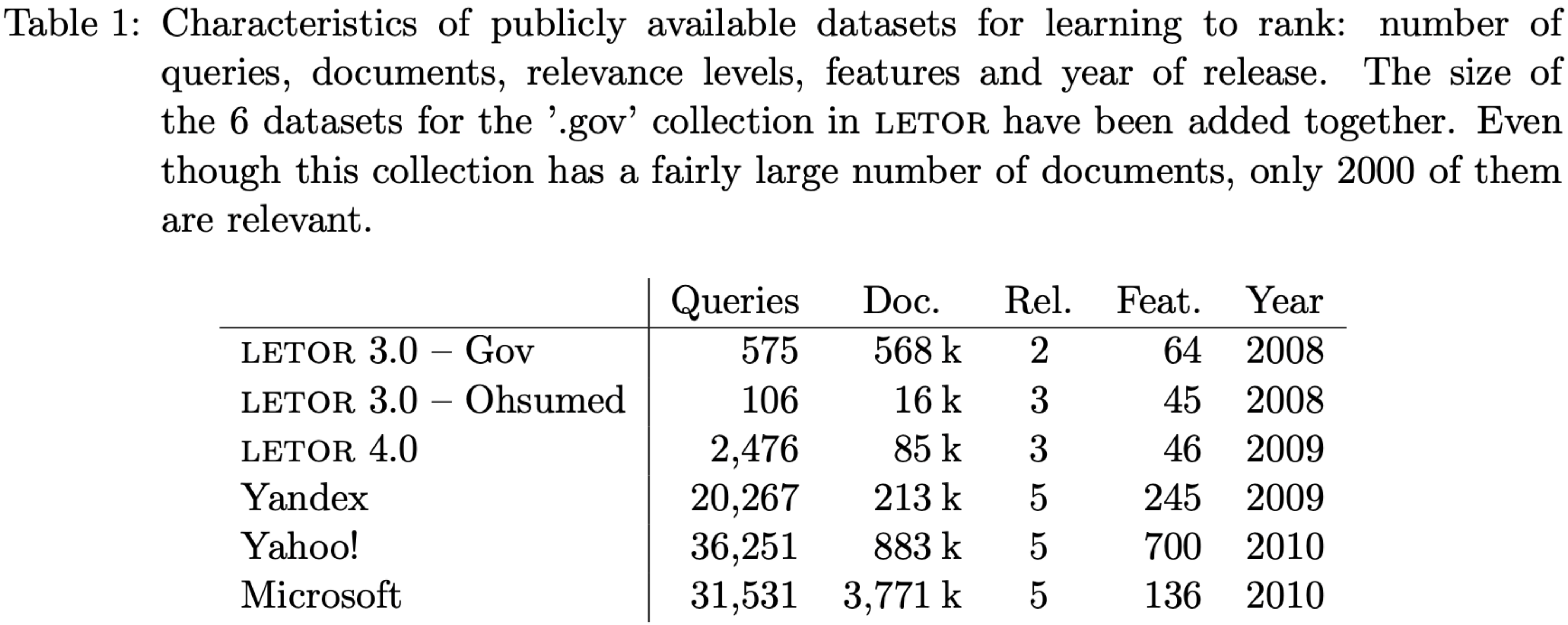
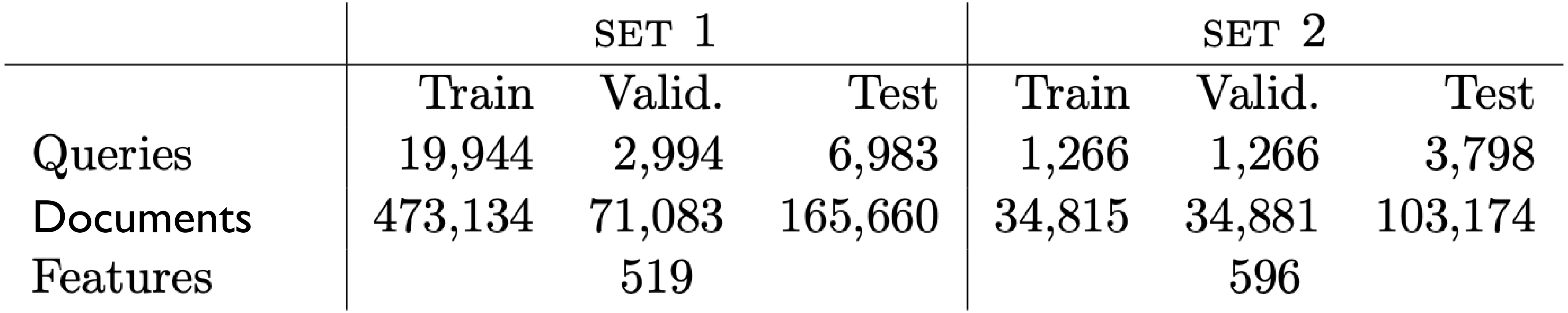


Table 1: Characteristics of publicly available datasets for learning to rank: number of queries, documents, relevance levels, features and year of release. The size of the 6 datasets for the '.gov' collection in LETOR have been added together. Even though this collection has a fairly large number of documents, only 2000 of them are relevant.

	Queries	Doc.	Rel.	Feat.	Year
LETOR 3.0 – Gov	575	568 k	2	64	2008
LETOR 3.0 – Ohsumed	106	16 k	3	45	2008
LETOR 4.0	2,476	85 k	3	46	2009
Yandex	20,267	213 k	5	245	2009
Yahoo!	36,251	883 k	5	700	2010
Microsoft	31,531	3,771 k	5	136	2010

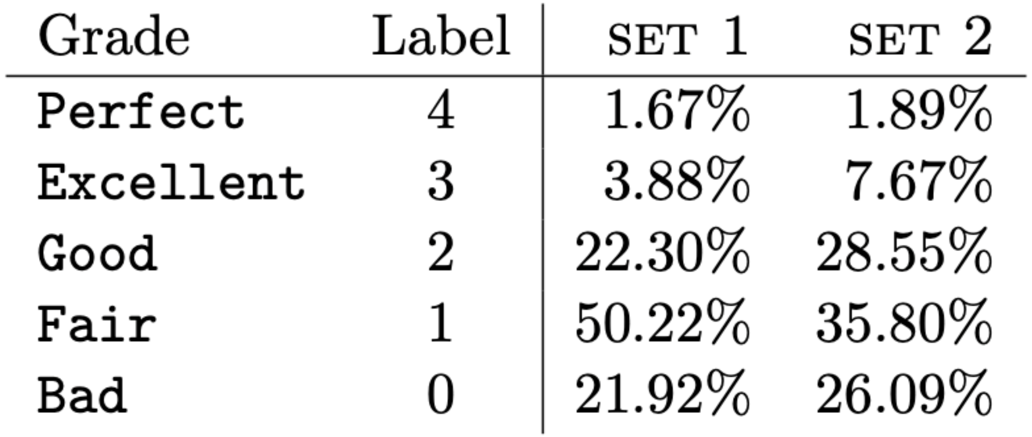
Datasets for the Challenge

Table 2: Statistics of the two datasets released for the challenge.



	SET 1			SET 2		
	Train	Valid.	Test	Train	Valid.	Test
Queries	19,944	2,994	6,983	1,266	1,266	3,798
Documents	473,134	71,083	165,660	34,815	34,881	103,174
Features		519			596	

Table 3: Distribution of relevance labels.



Grade	Label	SET 1	SET 2
Perfect	4	1.67%	1.89%
Excellent	3	3.88%	7.67%
Good	2	22.30%	28.55%
Fair	1	50.22%	35.80%
Bad	0	21.92%	26.09%

Features

- Web graph: in-links, out-links, PageRank, …
- Doc statistics: # of words in title, # of words in body, number of slashes in URL, …
- Doc classifier: spam, topic, language, …
- Query: # of terms, frequency of query and its terms, …
- Text match: BM25, counts, …
- Clicks: probability of a click, dwell time, …
- External references: tags
- Time: age of doc, age of in-links, …

Baselines

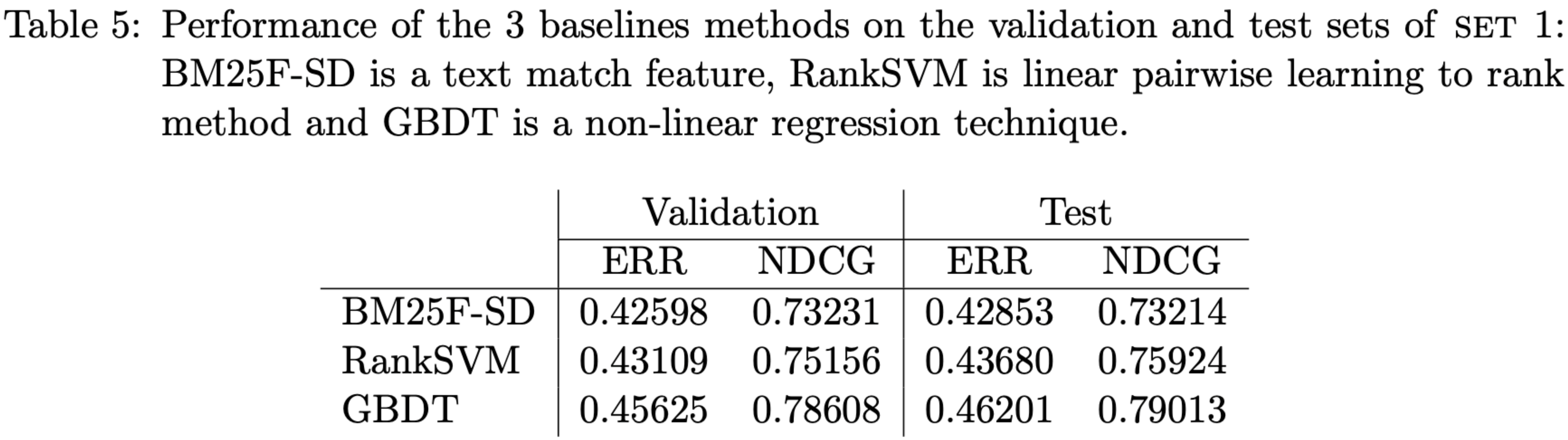
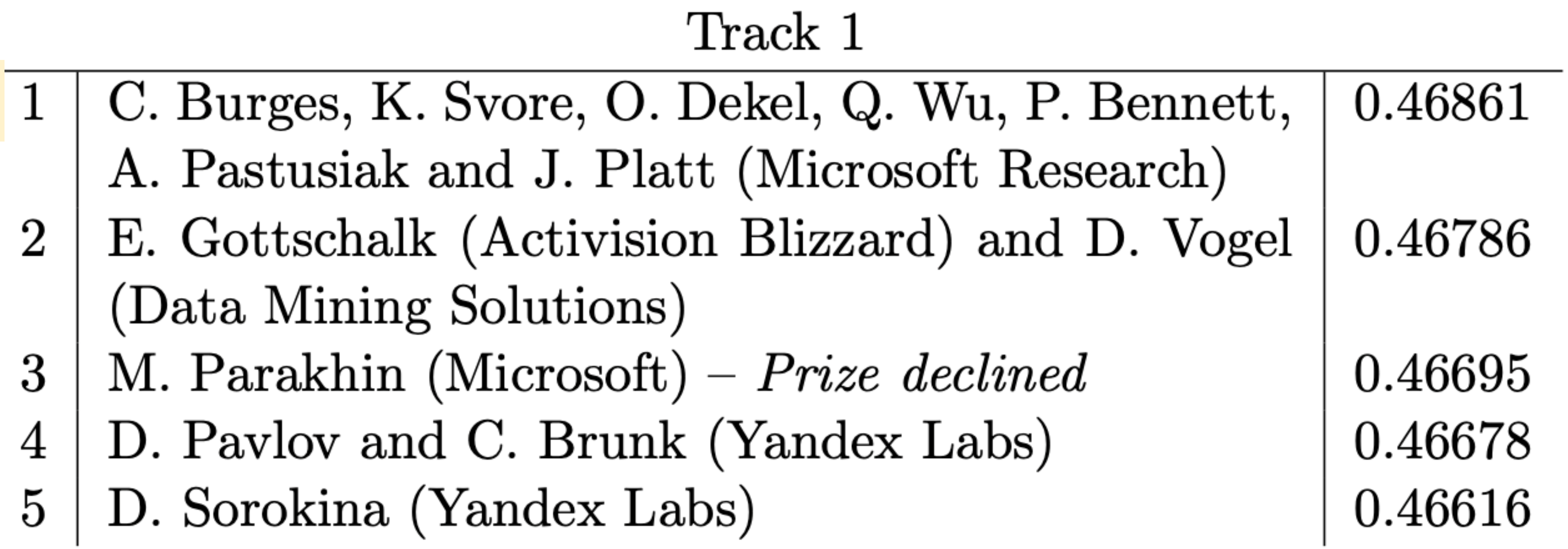


Table 5: Performance of the 3 baselines methods on the validation and test sets of SET 1: BM25F-SD is a text match feature, RankSVM is linear pairwise learning to rank method and GBDT is a non-linear regression technique.

	Validation		Test	
	ERR	NDCG	ERR	NDCG
BM25F-SD	0.42598	0.73231	0.42853	0.73214
RankSVM	0.43109	0.75156	0.43680	0.75924
GBDT	0.45625	0.78608	0.46201	0.79013

The Winners

RankNet



Track 1

1	C. Burges, K. Svore, O. Dekel, Q. Wu, P. Bennett, A. Pastusiak and J. Platt (Microsoft Research)	0.46861
2	E. Gottschalk (Activision Blizzard) and D. Vogel (Data Mining Solutions)	0.46786
3	M. Parakhin (Microsoft) – *Prize declined*	0.46695
4	D. Pavlov and C. Brunk (Yandex Labs)	0.46678
5	D. Sorokina (Yandex Labs)	0.46616

- What is RankNet? Next lecture!

RankNet: Background

- We have documents represented in a feature space: x is an input feature vector
- For document i the representation is x_i
- Our goal is to learn a function f that outputs a score (just a number)
  - $\mathrm{f}(x_i) = s_i$

- Suppose we have two documents i and j that we represent as: x_i and x_j
- We can score each document according to our function f
  - So, we have $\mathrm{f}(x_i) = s_i$ and $\mathrm{f}(x_j) = s_j$

- Let P_{ij} = Prob(document $i \succ$ document j)
  - That is, i is better than j for this query (the pairwise learning to rank setting)

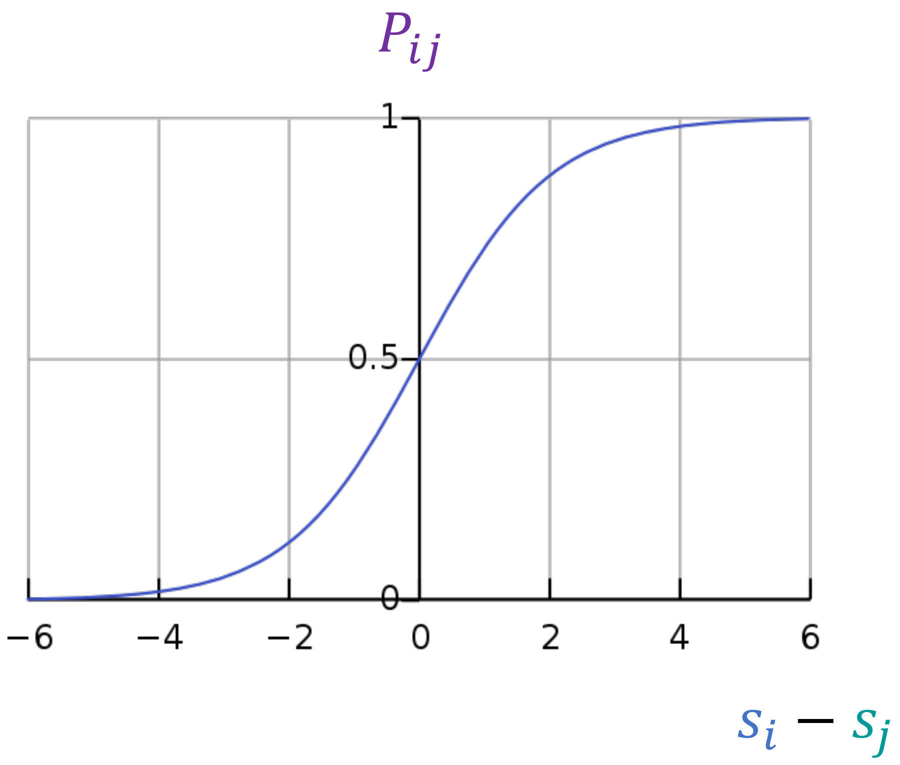
RankNet: Background

- $\mathrm{f}(x_i) = s_i$ and $\mathrm{f}(x_j) = s_j$
- Let P_{ij} = Prob(document $i \succ$ document j)

- We can use a Sigmoid function to define P_{ij} :
  - $P_{ij} = \frac{e^{s_i}}{e^{s_i} + e^{s_j}}$ *(a.k.a., Bradley–Terry Model)*

- What if $s_i \gg s_j$? $P_{ij} \rightarrow 1$
- What if $s_i \ll s_j$? $P_{ij} \rightarrow 0$
- What if $s_i = s_j$? $P_{ij} = 0.5$

- Why not directly define $P_{ij} = \begin{cases} 1, \text{if } s_i > s_j \\ 0.5, \text{if } s_i = s_j \\ 0, \text{if } s_i < s_j \end{cases}$?



Loss Function

- Loss function tells us what we pay when the function we are trying to learn makes mistakes.
- Here we use the cross-entropy loss function:

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - \left(1 - \hat{P}_{ij}\right)\log(1 - P_{ij})$$

- P_{ij} is the probability given by the model (i.e., $\frac{e^{s_i}}{e^{s_i}+e^{s_j}}$)
- $\hat{P}_{ij}$ is the actual (i.e., 1, 0.5, or 0, depending on if document $i \succ$ document j in our training data)
- $\log$: natural logarithm, to the base of e

Cross-Entropy Loss

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - \left(1 - \hat{P}_{ij}\right)\log(1 - P_{ij})$$

- Good for binary classification tasks (like in our case, is one document "better" than the other document?)

- Example:
  - Suppose the ground truth is document $i \succ$ document j
    - $\hat{P}_{ij} = 1$
  - Our model predicts document $i \succ$ document j with $P_{ij} = 0.99$
  - $\mathcal{L} \approx 0.01$

Cross-Entropy Loss

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - \left(1 - \hat{P}_{ij}\right)\log(1 - P_{ij})$$

- Good for binary classification tasks (like in our case, is one document "better" than the other document?)

- Example:
  - Suppose the ground truth is document $i \succ$ document j
    - $\hat{P}_{ij} = 1$
  - Our model predicts document $i \succ$ document j with $P_{ij} = 0.50$
  - $\mathcal{L} \approx 0.69$

Cross-Entropy Loss

$$\mathcal{L} = -\hat{P}_{ij}\log P_{ij} - \left(1 - \hat{P}_{ij}\right)\log(1 - P_{ij})$$

- Good for binary classification tasks (like in our case, is one document "better" than the other document?)

- Example:
  - Suppose the ground truth is document $i \succ$ document j
    - $\hat{P}_{ij} = 1$
  - Our model predicts document $i \succ$ document j with $P_{ij} = 0.01$
  - $\mathcal{L} \approx 4.61$

Loss Function

- $P_{ij} = \frac{e^{s_i}}{e^{s_i}+e^{s_j}}$
- $\mathcal{L} = -\hat{P}_{ij}\text{log}P_{ij} - \left(1 - \hat{P}_{ij}\right)\text{log}(1 - P_{ij})$
- After doing some math, you will have
- $\mathcal{L} = -\hat{P}_{ij}\left(s_i - s_j\right) + \log(1 + e^{s_i - s_j})$
- Given the cost function, our goal is to learn the best model parameters to minimize the cost. That is, we hope to pick some combination of features that do a good job of guessing when a document is preferred to another document in our training data.
- What are the model parameters in this case?
  - $\text{f}(x_i) = s_i$
  - E.g., $\text{f}(x_i) = w^T s_i$

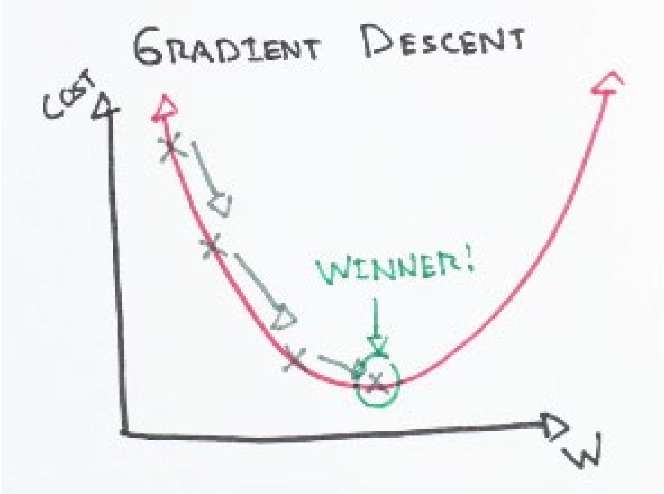
How to minimize the cost?

- Gradient descent!

- To minimize a function $\mathrm{f}(x)$:
    - Take the derivative of f: $\nabla\mathrm{f}$
    - Start at some point x_0 and evaluate $\nabla\mathrm{f}(x_0)$
    - Make a step in the reverse direction of the gradient: $x_1 = x_0 - \eta\nabla\mathrm{f}(x_0)$
        - $\eta > 0$ is the learning rate
    - Repeat until converged

How to minimize the cost?

- Example
  - $\mathrm{f}(x) = (x-2)^2 + 1$
  - $\nabla\mathrm{f}(x) = 2(x-2)$
  - If $x_0 < 2$, then $\nabla\mathrm{f}(x_0) < 0$, so $x_1 = x_0 - \eta\nabla\mathrm{f}(x_0) > x_0$
    - If we still have $x_1 < 2$, then $x_2 = x_1 - \eta\nabla\mathrm{f}(x_1) > x_1$
  - If $x_0 > 2$, then $\nabla\mathrm{f}(x_0) > 0$, so $x_1 = x_0 - \eta\nabla\mathrm{f}(x_0) < x_0$
  - Finally, we always have x converge to 2, which minimizes $\mathrm{f}(x)$



How to minimize the cost?

- Example
  - $\mathrm{f}(x, y) = (x-2)^2 + (y-3)^2 + 1$
  - $\nabla \mathrm{f}(x, y) = \begin{bmatrix} \frac{\partial \mathrm{f}}{\partial x} \\ \frac{\partial \mathrm{f}}{\partial y} \end{bmatrix} = \begin{bmatrix} 2(x-2) \\ 2(y-3) \end{bmatrix}$
  - $\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} - \eta \begin{bmatrix} 2(x_0-2) \\ 2(y_0-3) \end{bmatrix}$
    - If $x_0 < 2$, then $x_1 > x_0$
    - If $x_0 > 2$, then $x_1 < x_0$
    - Regardless of how y changes, x keeps moving closer to 2
    - Similarly, regardless of how x changes, y keeps moving closer to 3
  - Finally, we always have $\begin{bmatrix} x \\ y \end{bmatrix}$ converge to $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$, which minimizes $\mathrm{f}(x, y)$

Going Back to RankNet

- $\mathcal{L} = -\hat{P}_{ij}\left(s_i - s_j\right) + \log(1 + e^{s_i - s_j})$
- $\frac{\partial \mathcal{L}}{\partial s_i} = -\hat{P}_{ij} + \frac{e^{s_i - s_j}}{1+e^{s_i - s_j}}$ Let's denote this as λ_{ij}
- $\frac{\partial \mathcal{L}}{\partial s_j} = \hat{P}_{ij} - \frac{e^{s_i - s_j}}{1+e^{s_i - s_j}} = -\lambda_{ij}$

- Then how can we update a model parameter w_k in $\mathrm{f}(x)$ (e.g., $\mathrm{f}(x_i) = w^T s_i$)?
- Gradient Descent:

$$w_k \leftarrow w_k - \eta \frac{\partial \mathcal{L}}{\partial w_k}$$

- But how to calculate $\frac{\partial \mathcal{L}}{\partial w_k}$?

Going Back to RankNet

- Then how can we update a model parameter w_k in $\mathrm{f}(x)$ (e.g., $\mathrm{f}(x_i) = w^T s_i$)?
- Gradient Descent:

$$w_k \leftarrow w_k - \eta \frac{\partial \mathcal{L}}{\partial w_k}$$

- But how to calculate $\frac{\partial \mathcal{L}}{\partial w_k}$?
  - (Multi-variable) chain rule:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \frac{\partial \mathcal{L}}{\partial s_i}\frac{\mathrm{d}s_i}{\mathrm{d}w_k} + \frac{\partial \mathcal{L}}{\partial s_j}\frac{\mathrm{d}s_j}{\mathrm{d}w_k} = \lambda_{ij}\frac{\mathrm{d}s_i}{\mathrm{d}w_k} + (-\lambda_{ij})\frac{\mathrm{d}s_j}{\mathrm{d}w_k} = \lambda_{ij}\left(\frac{\mathrm{d}s_i}{\mathrm{d}w_k} - \frac{\mathrm{d}s_j}{\mathrm{d}w_k}\right)$$

Interesting Observation

$$\frac{\partial \mathcal{L}}{\partial w_k} = \lambda_{ij}\left(\frac{\mathrm{d}s_i}{\mathrm{d}w_k} - \frac{\mathrm{d}s_j}{\mathrm{d}w_k}\right)$$

- This is the gradient for only one training sample (document $i \succ$ document j)
- What if you have lots of training samples?
  - Add these gradients together!
  - If document i appears in the following training samples:
    - $(i \succ j_1), (i \succ j_2), \ldots, (i \succ j_{10})$
    - $(j_{11} \succ i), (j_{12} \succ i), \ldots, (j_{18} \succ i)$
  - What would be the term related to document i in the total gradient?

$$(\lambda_{ij_1} + \lambda_{ij_2} + \cdots + \lambda_{ij_{10}} - \lambda_{j_{11}i} - \lambda_{j_{12}i} - \cdots - \lambda_{j_{18}i})\frac{\mathrm{d}s_i}{\mathrm{d}w_k}$$

Interesting Observation

- What would be the term related to document i in the total gradient?

$$(\lambda_{ij_1} + \lambda_{ij_2} + \cdots + \lambda_{ij_{10}} - \lambda_{j_{11}i} - \lambda_{j_{12}i} - \cdots - \lambda_{j_{18}i}) \frac{\mathrm{d}s_i}{\mathrm{d}w_k}$$

- λ_{ij} describes the desired change of scores for the pair of training sample (document $i \succ$ document j).

- The sum of all λ_{ij}'s and λ_{ji}'s related to document i describes the desired direction of i's movement within the entire ranking list.

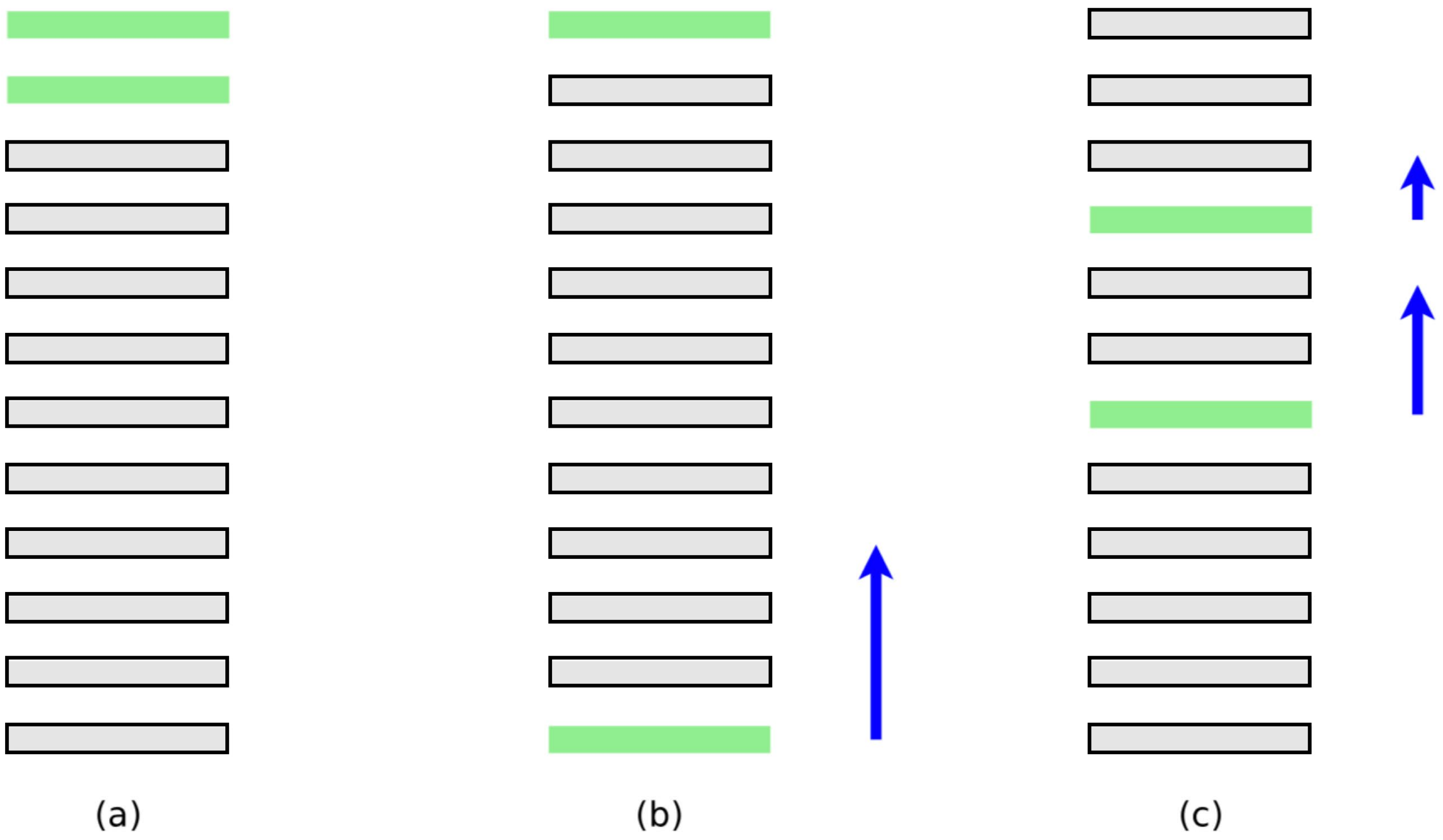
$$\lambda_i = \sum_{j:(i \succ j)} \lambda_{ij} - \sum_{j:(j \succ i)} \lambda_{ji}$$

Example

Each blue arrow represents the λ_i for the corresponding document i.

Green: Relevant to the query

Gray: Irrelevant to the query



Perfect

10 pairwise errors

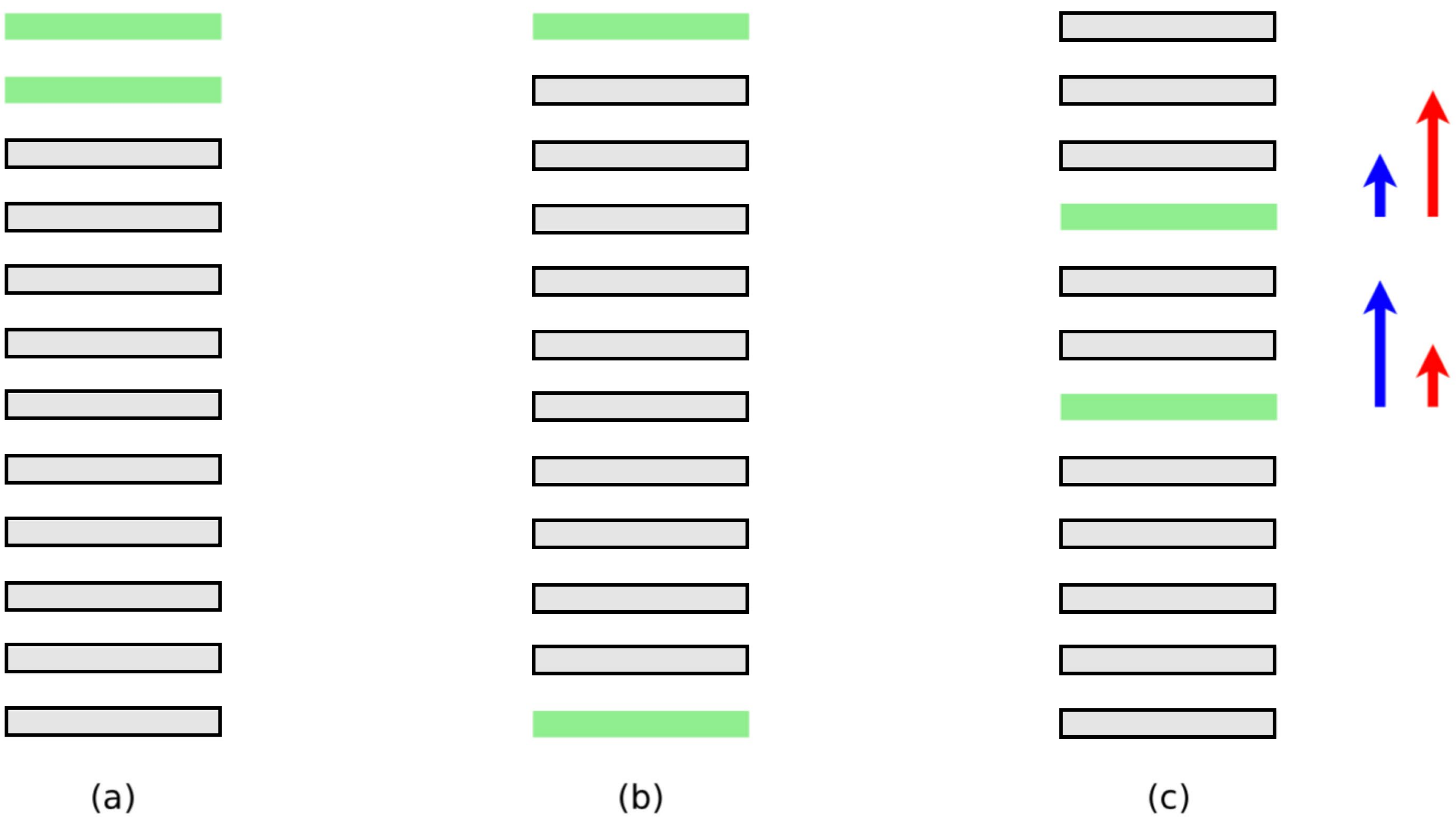
8 pairwise errors

Example

However, based on what we know about the importance of ranking in search engines, the red arrow seems more reasonable. (Why?)

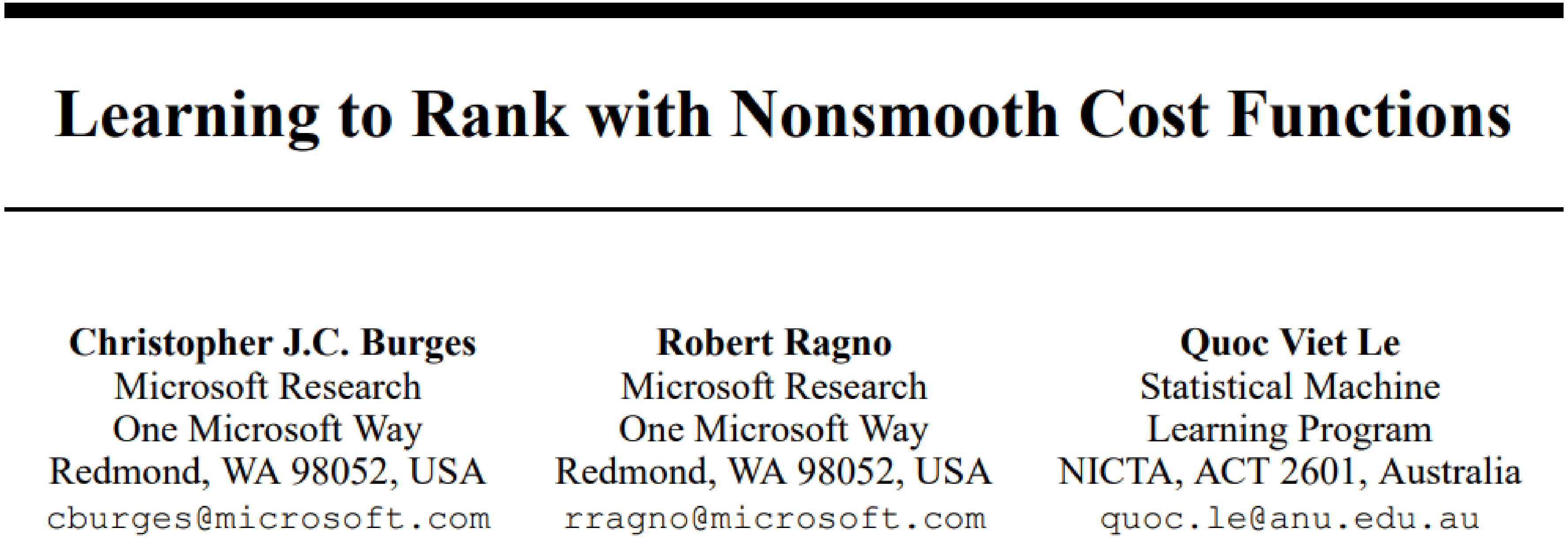
Green: Relevant to the query

Gray: Irrelevant to the query



RankNet → LambdaRank [Burges et al., NIPS 2006]

- Instead of working directly with pairwise ranking errors, scale them by their impact on nDCG.
- Idea: Multiply λ_{ij} by the difference of an IR measure (e.g., nDCG) when document i and document j are swapped



Learning to Rank with Nonsmooth Cost Functions

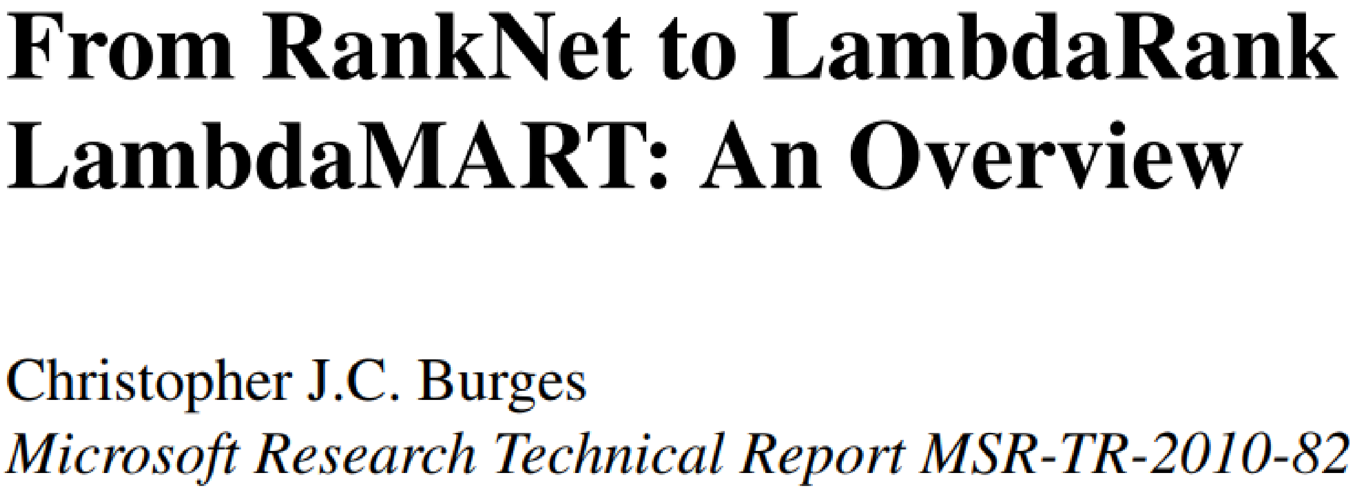
Christopher J.C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
cburges@microsoft.com

Robert Ragno
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
rragno@microsoft.com

Quoc Viet Le
Statistical Machine
Learning Program
NICTA, ACT 2601, Australia
quoc.le@anu.edu.au

LambdaRank → LambdaMART [Burges, 2010]

- A boosted tree version of LambdaRank
- MART = multiple additive regression trees



From RankNet to LambdaRank to LambdaMART: An Overview

Christopher J.C. Burges

Microsoft Research Technical Report MSR-TR-2010-82

Course Project Information Session

Group Project

- Teams of 3 or 4 (any deviation from this size requires prior approval from the instructor)
  - Option 1:A prototype (search engine or recommender system)
  - Option 2:A research project (e.g., reasoning-search interleaved LLMs)
  - Option 3:A survey (e.g., recent studies on search-enhanced LLMs)
- Topic-wise: your choice, as long as it is related to the themes of this course!
- Proposal is due after you will have seen basics of both search engines and recommender systems
  - Think of this as a launching pad for your project and not a constraint

Themes of this Course

- ✅ Phase 1: Search Engines
  - Boolean and ranked retrieval, link analysis, evaluation, learning to rank (ML + ranking), …
- Phase 2: Recommender Systems
  - content-based recommendation, collaborative filtering, matrix factorization, recommendation with implicit feedback, …

Proposal Due

- Phase 3: From Foundations to Modern Methods
  - word embedding learning, Transformer, "small" language models, … (for search and recommendation)
- Phase 4: Large Language Models (!!)

Proposal (2% of your Overall Grade)

- March 7 (Saturday) by 11:59pm CT
- Post to Canvas
  - Only one team member needs to submit the PDF
  - TA will create the teams on Canvas after grading all proposals, since some groups may deviate from the required size without notifying us in advance and we may need to adjust their teams a bit
- ACL 2023 template: https://2023.aclweb.org/calls/style_and_formatting/
  - Keep it within 2 pages maximum
- In the proposal PDF, include all team member names and NetID
- List each of the following questions and provide a brief answer (pick the set of questions appropriate for your project: prototype, research, or survey)

Proposal: Prototype

- What is exactly the function of your tool? That is, what will it do?
- Why would we need such a tool and who would you expect to use it and benefit from it?
- Does this kind of tool already exist? If similar tools exist, how is your tool different from them? Would people care about the difference? How hard is it to build such a tool? What is the challenge?
- How do you plan to build it? You should mention the data you will use and the core algorithm that you will implement.
- What existing resources can you use?
- How will you demonstrate the usefulness of your tool?
- A rough timeline to show when you expect to finish what. List a couple of milestones.

Additional Requirements for a Prototype Project

- Requirement 1:You must use *non-trivial* data
  - Examples of *non-trivial* data
    - Sample using an API
    - Download an existing collection
    - Write a crawler
  - Examples of *trivial* data
    - You spend 10 minutes writing 3 queries and 10 candidate sentences

- Requirement 2:You should spend 3-4 minutes giving a live demo of the prototype in your final project presentation.
  - Example (in terms of form, not content): https://www.youtube.com/watch?v=a1IQZz0cUUc

Proposal: Research

- What is your research question? Clearly define the research problem/question.
- Why is this an interesting question to ask and why would we care about the answer to this question or a solution to the problem.
- Has any existing research work tried to answer the same or a similar question, and if so, what is still unknown?
- How do you plan to work out the answer to the question. (At the proposal stage, you are only expected to have a sketch of your methods.)
- How would you evaluate your solution. That is, how do you plan to demonstrate that your solution/answer is good or is reasonable.
- A rough timeline to show when you expect to finish what. List a couple of milestones.

Additional Requirements for a Research Project

- Requirement 1:You should identify a recent high-quality research paper (e.g., published at SIGIR, WWW, KDD, WSDM, RecSys, ACL, EMNLP, NAACL, etc.), better with an existing codebase
  - Technically, focus on one clean insight to improve the model proposed in the paper
  - Empirically, you do NOT have to show an improvement (remember, this is a short-fuse class project), but you need to identify key findings
- Requirement 2: Code must live on GitHub
  - Public code repository; or private + share with TA and me
  - Write a detailed README
  - Example (in terms of form, not content): https://github.com/yuzhimanhua/MATCH

Proposal: Survey

- What is the scope of your survey? Clearly define the area, topic, or set of methods you plan to cover.
- Why is this an important area to survey? Who would benefit from such a survey?
- Has there already been a survey or overview written on this topic? If so, what gaps or limitations exist that your survey could address?
- How will you structure your survey? For example, by taxonomy, chronology, methodology, application, or another organizing principle.
- What do you hope the reader will learn from your survey? What kinds of insights or take-aways will you emphasize?
- A rough timeline to show when you expect to finish what. List a couple of milestones.

Additional Requirements for a Survey Project

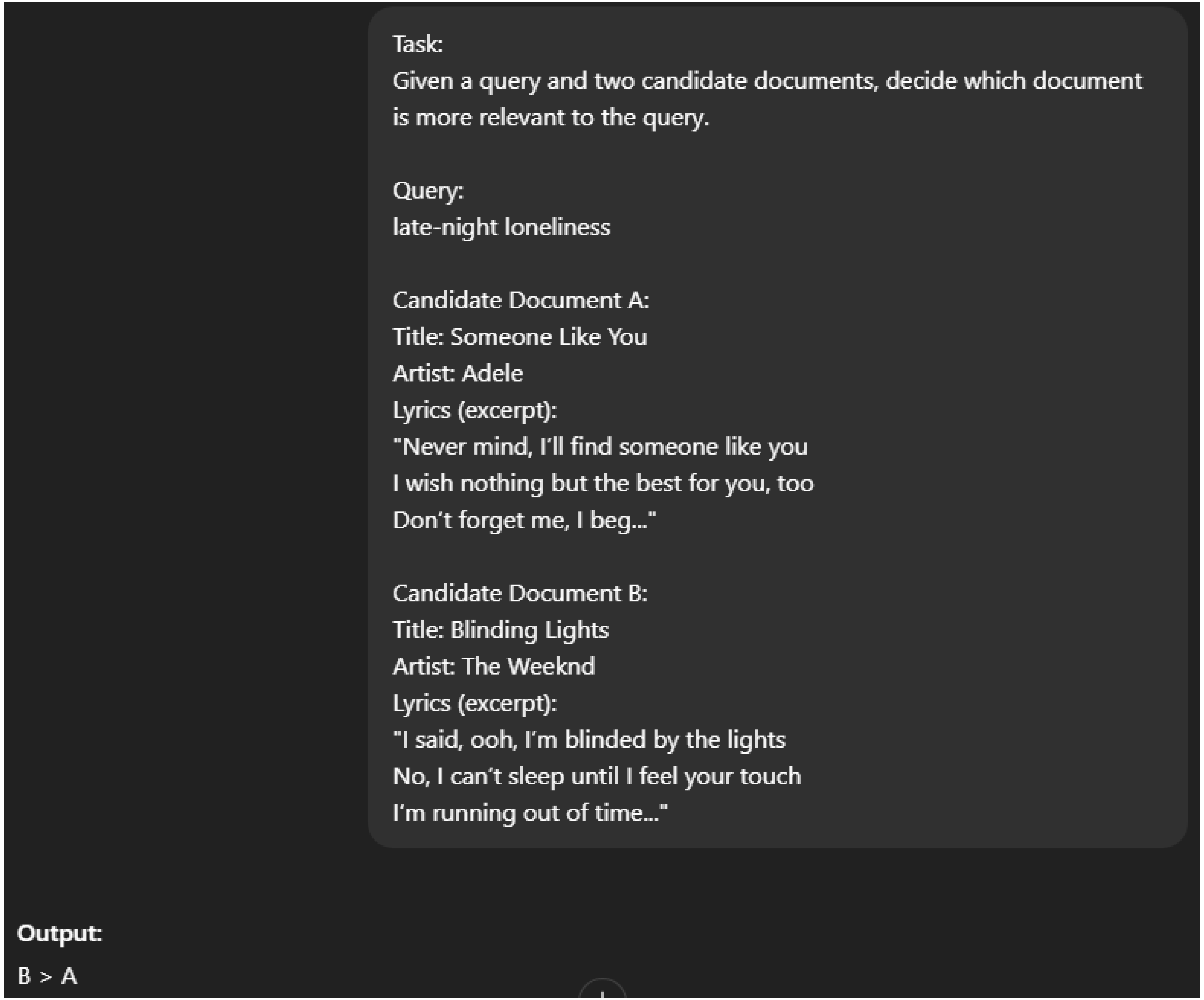
- Requirement 1: At least 20 papers, with no fewer than half published within the past three years.
  - Unpublished preprints (e.g., on arXiv) can also be included.

- Requirement 2: The works must be organized in a coherent and systematic order, rather than simply listed/appended.
  - A taxonomy of surveyed works should be provided in your final presentation/report
  - Example (in terms of form, not content): Figure 3 in https://arxiv.org/pdf/2502.17504

- Requirement 3: A corresponding GitHub repository (usually named as "*Awesome-XXX*") is required (a README file suffices).
  - Public; or private + share with TA and me
  - Example (in terms of form, not content or number of papers surveyed): https://github.com/yuzhimanhua/Awesome-Scientific-Language-Models

Suggestions

- Suggestion 1: Clearly distinguish your project from existing efforts
  - You are welcome to use any existing resources that you like
  - We evaluate what you add, rather than what you use
- Suggestion 2: Have a sharp focus
  - Do something well, rather than a million things poorly
- Suggestion 3: If you do not have great data, consider ChatGPT as a "data augmenter".



Task:
Given a query and two candidate documents, decide which document is more relevant to the query.

Query:
late-night loneliness

Candidate Document A:
Title: Someone Like You
Artist: Adele
Lyrics (excerpt):
"Never mind, I'll find someone like you
I wish nothing but the best for you, too
Don't forget me, I beg..."

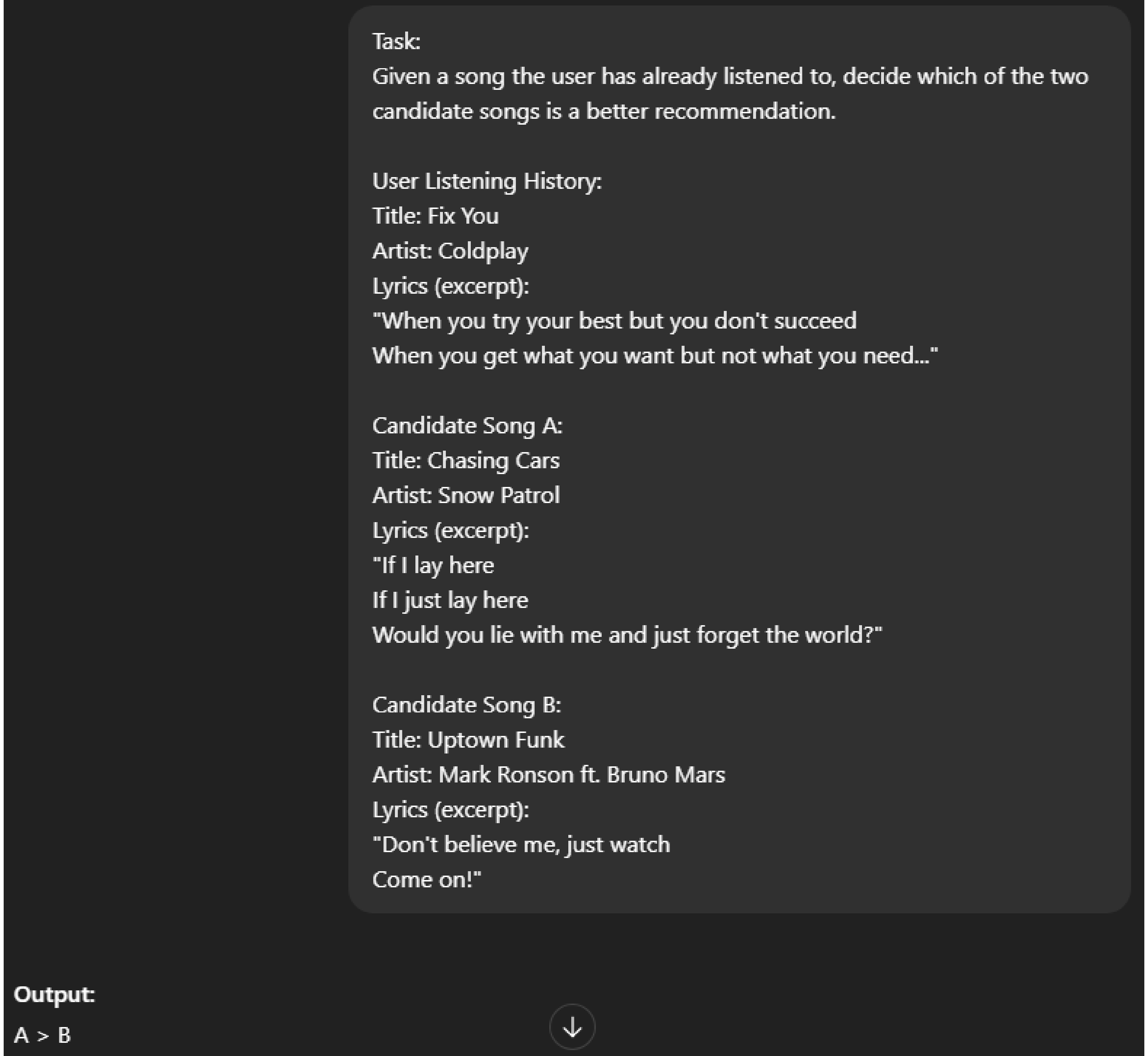
Candidate Document B:
Title: Blinding Lights
Artist: The Weeknd
Lyrics (excerpt):
"I said, ooh, I'm blinded by the lights
No, I can't sleep until I feel your touch
I'm running out of time..."

Output:

B > A

LLM as a “Ground-Truth” Generator (Search)

LLM as a “Ground-Truth” Generator (Recommendation)



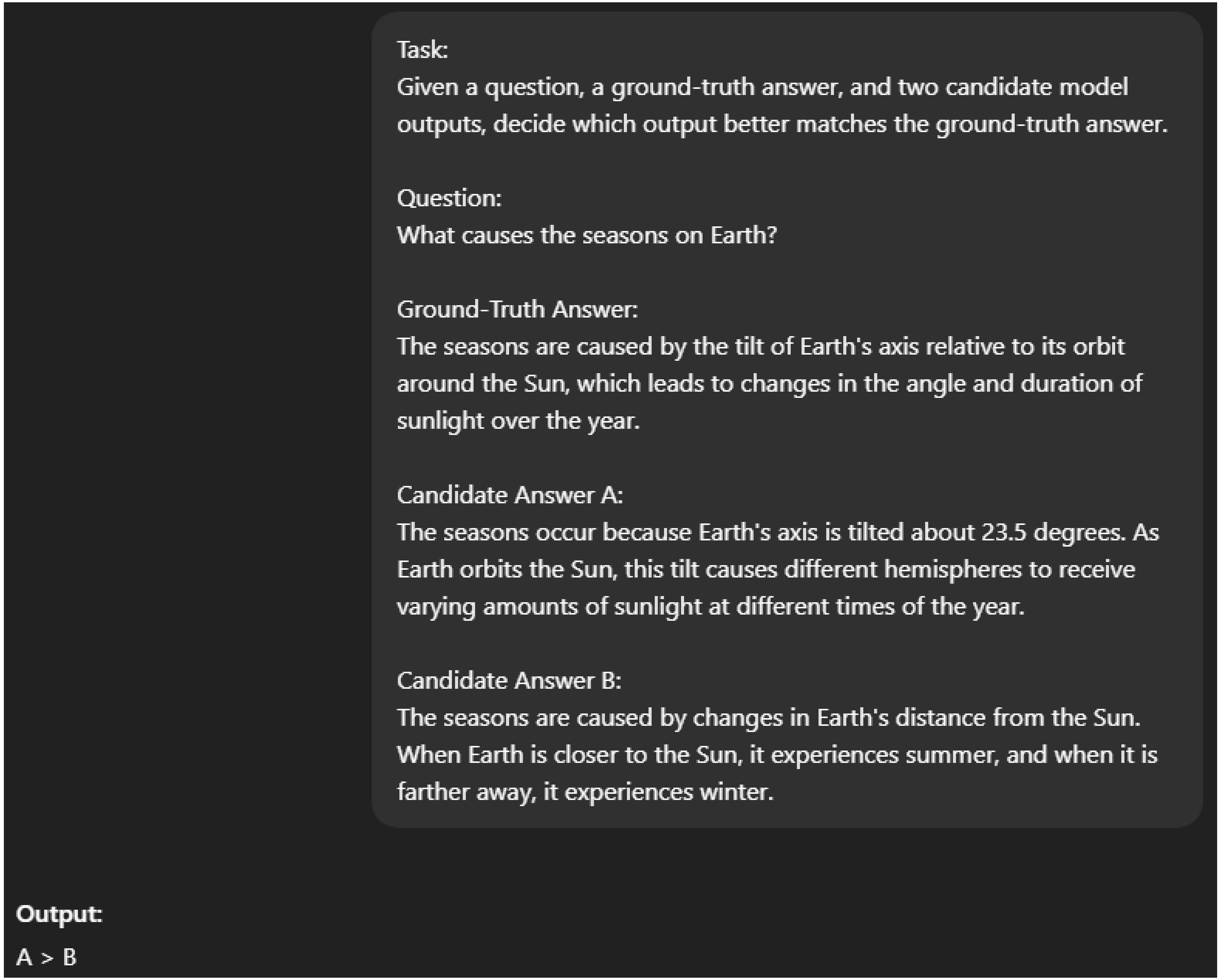
Task:
Given a song the user has already listened to, decide which of the two candidate songs is a better recommendation.

User Listening History:
Title: Fix You
Artist: Coldplay
Lyrics (excerpt):
"When you try your best but you don't succeed
When you get what you want but not what you need..."

Candidate Song A:
Title: Chasing Cars
Artist: Snow Patrol
Lyrics (excerpt):
"If I lay here
If I just lay here
Would you lie with me and just forget the world?"

Candidate Song B:
Title: Uptown Funk
Artist: Mark Ronson ft. Bruno Mars
Lyrics (excerpt):
"Don't believe me, just watch
Come on!"

Output:
A > B



Task:
Given a question, a ground-truth answer, and two candidate model outputs, decide which output better matches the ground-truth answer.

Question:
What causes the seasons on Earth?

Ground-Truth Answer:
The seasons are caused by the tilt of Earth's axis relative to its orbit around the Sun, which leads to changes in the angle and duration of sunlight over the year.

Candidate Answer A:
The seasons occur because Earth's axis is tilted about 23.5 degrees. As Earth orbits the Sun, this tilt causes different hemispheres to receive varying amounts of sunlight at different times of the year.

Candidate Answer B:
The seasons are caused by changes in Earth's distance from the Sun. When Earth is closer to the Sun, it experiences summer, and when it is farther away, it experiences winter.

Output:

A > B

LLM as a Judge (RAG)

Presentation (9% of your Overall Grade)

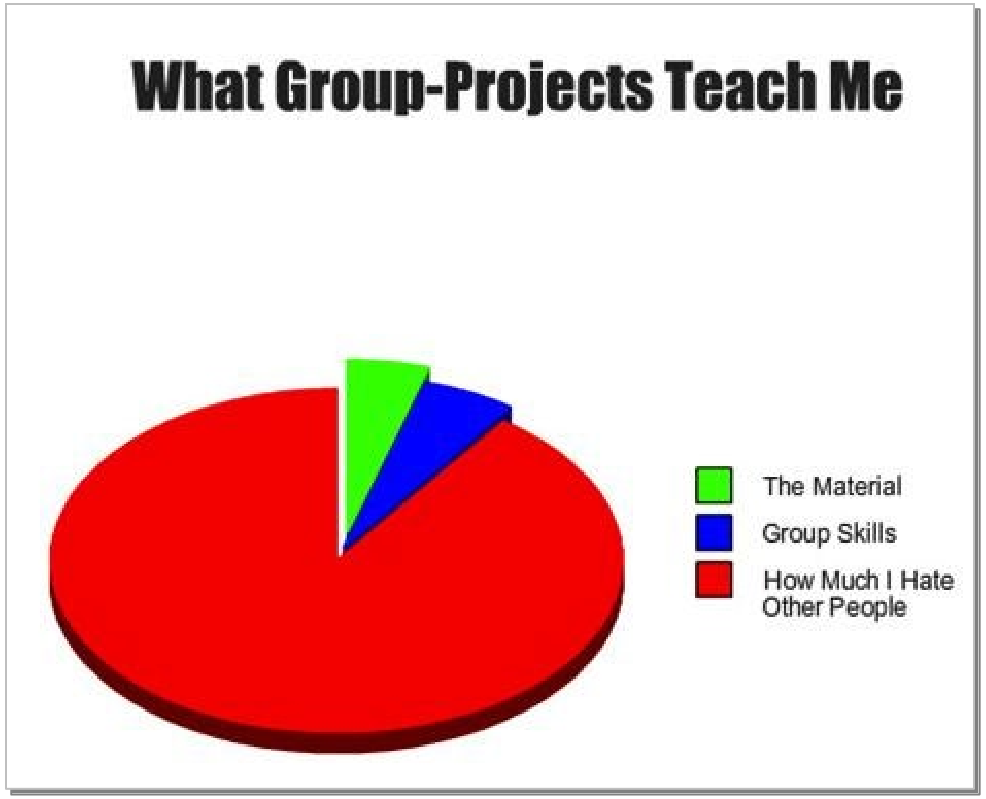
- April 20, April 22, April 24, and April 27 classes
- Zoom: https://tamu.zoom.us/j/6411788612
- The duration depends on the number of groups, with an estimated 8-10 minutes per group.
    - The 3-4 minute live demo by prototype groups is included within the estimated 8-10 minutes per group.
- Not everyone in the group needs to participate in the presentation.
- Any slide template is okay.

Final Report (9% of your Overall Grade)

- May 2 (Saturday) by 11:59pm CT
- Post to Canvas
- ACL 2023 template: https://2023.aclweb.org/calls/style_and_formatting/
  - Prototype: 3-5 pages
    - Example (form, not content): https://aclanthology.org/2020.acl-demos.5.pdf
  - Research: 6-8 pages
    - Example (form, not content): https://aclanthology.org/2024.findings-acl.11.pdf
  - Survey: 6-8 pages
    - Example (form, not content): https://aclanthology.org/2024.emnlp-main.498.pdf
- At that point, we will have already assigned you to teams on Canvas. Each team only needs to submit one PDF

Teamwork

- You will receive an overall grade based on the performance of your entire team.
- We will also assess your individual contribution (through peer feedback, contributions to the code repository, etc.).
- Typically, the individual rating can increase or depress your project grade by only some small delta.



Thank You!

Course Website: https://yuzhang-teaching.github.io/CSCE670-S26.html