Basic Elements of IR Systems

SI 650 / EECS 549
Information Retrieval

September 3, 2025

Some slides by Dr. Jurgens

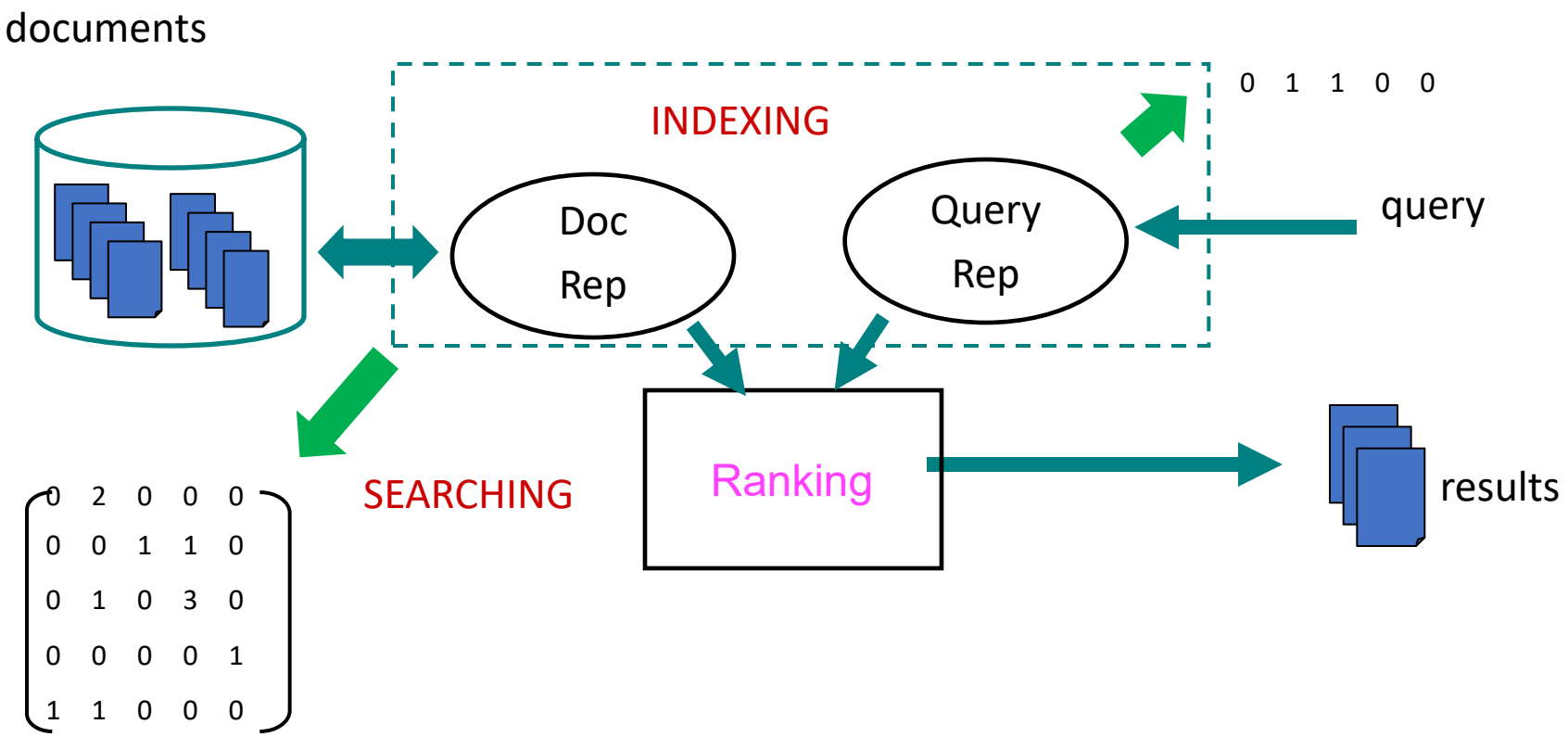
Announcements

- Homework 1 is released. Please attend discussion sections for a walk through.

Today's Goals

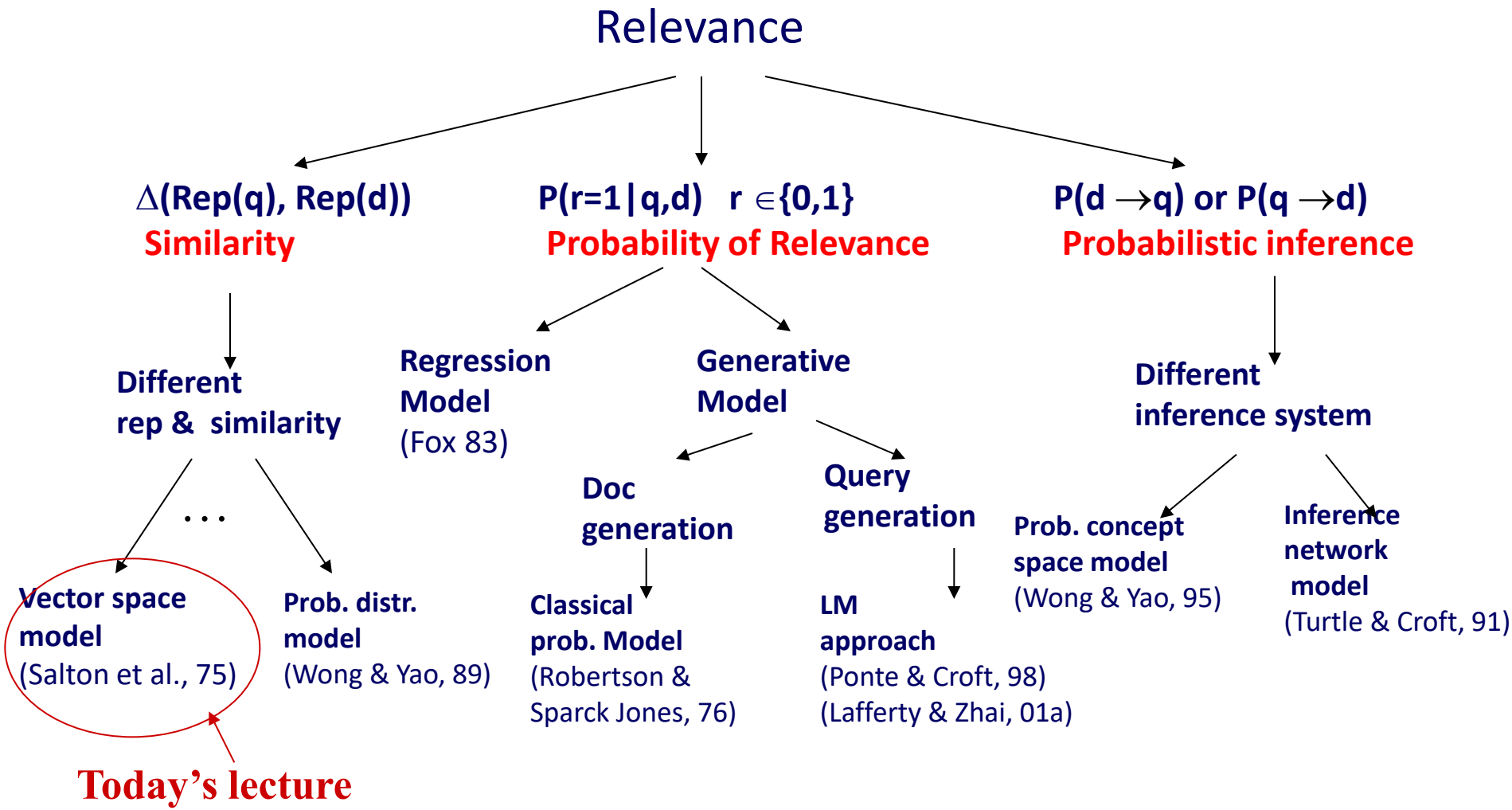
- Defining relevance
- Evaluating IR systems
- Indexing
- IR and fairness (optional)

Let's Start from Here…



How do we define/determine relevance?

The Notion of Relevance



Formal Formulation of Text Retrieval

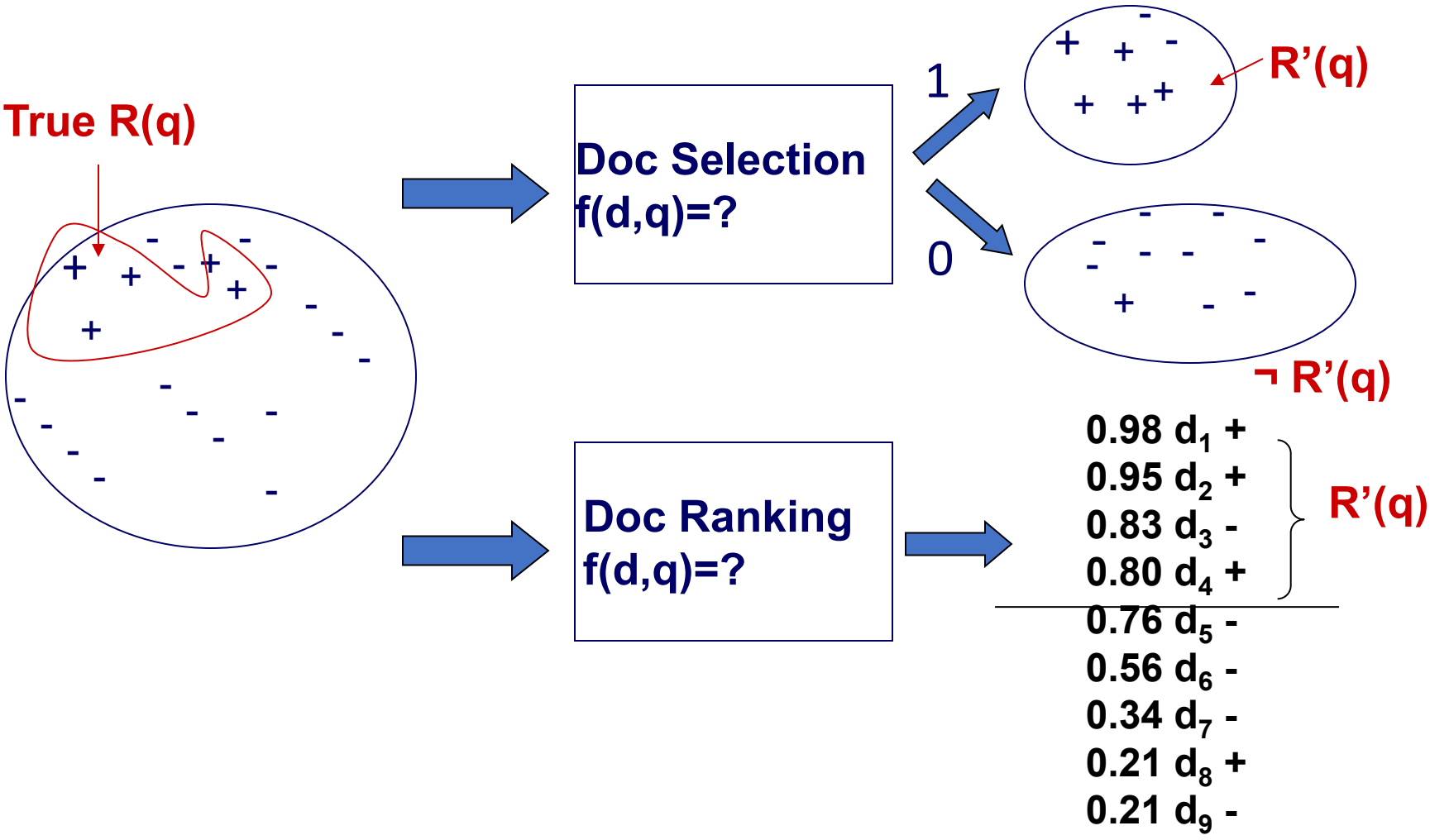
- Vocabulary V={w_1, w_2, …, w_N} of language
- Query q = q_1,…,$q_{m,}$ where $q_i \in V$
- Document d_i = d_{i1},…,$d_{iL,}$ where $d_{ij} \in V$
- Collection C= {d_1, …, d_k}
- Set of relevant documents R(q) $\subseteq$ C
  - Generally unknown and user-dependent
  - Query is a "hint" on which doc is in R(q)
- Task = compute R'(q), an "approximate R(q)"
  - Because we never know true R(q)…

R(q) is the ideal set of documents we should return for query q

Defining and computing R(q)

- Strategy 1: Document selection
  - $R(q)=\{d \in C | f(d,q)=1\}$, where $f(d,q) \in \{0,1\}$ is an indicator function or classifier
  - System must decide if a doc is relevant or not ("absolute relevance")
- Strategy 2: Document ranking
  - $R(q) = \{d \in C | f(d,q)>\theta\}$, where $f(d,q) \in \Re$ is a relevance measure function; θ is a cutoff
  - System must decide if one doc is more likely to be relevant than another ("relative relevance")

Document Selection vs. Ranking



Problems of Doc Selection

- The classifier is unlikely accurate
  - “Over-constrained” query (terms are too specific): no relevant documents found
  - “Under-constrained” query (terms are too general): over delivery
  - It is extremely hard to find the right position between these two extremes
- Even if it is accurate, all relevant documents are not equally relevant
- Relevance is a matter of degree!

Ranking is often preferred

- Relevance is a matter of degree
- A user can stop browsing anywhere, so the boundary is controlled by the user
  - High recall users would view more items
  - High precision users would view only a few
- Theoretical justification: Probability Ranking Principle [Robertson 77]

Probability Ranking Principle
[Robertson 77]

- As stated by Cooper

  "If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data."
- Robertson provides two formal justifications
- Assumptions: Independent relevance and sequential browsing (not necessarily all hold in reality)

According to the PRP, all we need is

"A relevance measure function f"

which satisfies

For all q, d_1, d_2,

$f(q,d_1) > f(q,d_2)$ iff $p(\text{Relevance}|q,d_1) > p(\text{Relevance}|q,d_2)$

Relevance = Similarity

- Assumptions
  - Query and document are represented similarly
  - A query can be regarded as a "document"
  - Relevance(d, q) $\propto$ similarity(d, q)
- R'(q) = {d$\in$C | f(d,q)>θ}, f(q,d) = Δ(Rep(q), Rep(d))
- Δ is a similarity function of the document and query representations
- Key issues
  - How to represent query/document?
  - How to define the similarity function Δ?

Revisit: Vector Space Model

- Represent a doc/query by a term vector
  - Term: basic concept, e.g., word or phrase
  - Each term corresponds to one dimension
  - N terms define a high-dimensional space
  - Element of vector corresponds to term weight
  - E.g., $d=(x_1,\ldots,x_N)$, x_i is the "importance" of term i
- Measure relevance by the similarity/distance (e.g., the cosine similarity) between the query vector and document vector in the vector space

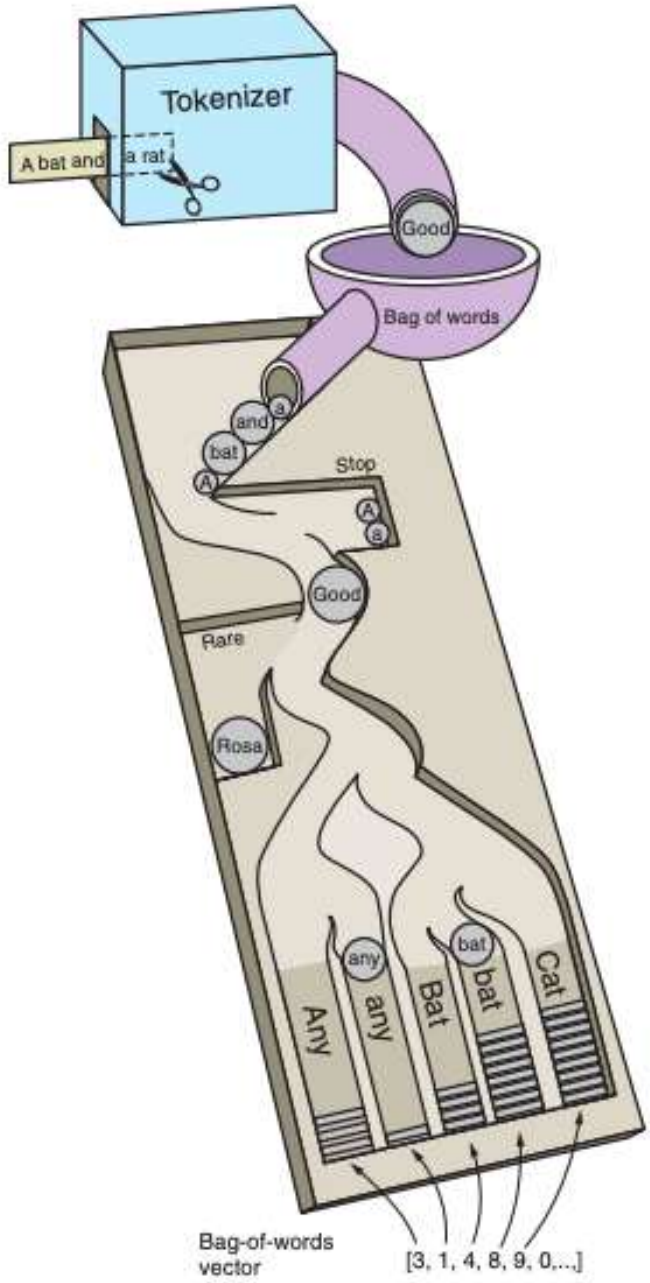


Figure 1.2 Token sorting tray

"Bag of words" means we don't care about word order—just how many times each word appears (simple/wrong but effective in practice)

What the Vector Space model *doesn't* say

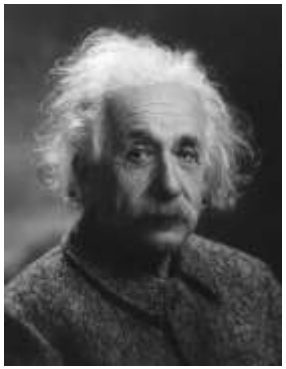
- How to define/select the "basic concept"
  - We talked about how to select index terms
  - Concepts are assumed to be orthogonal
- How to assign weights
  - Weight in query indicates importance of term
  - Weight in doc indicates how well the term characterizes the document
  - We talked about simple presence/absence
- How to define the similarity/distance measure

What's a good "basic concept"?

- Many possibilities: Words, stemmed words, phrases, "latent concept", …
- Orthogonal
  - Linearly independent
  - Non-overlapping in meaning
  - Related challenge: typos

Query = {laptop computer pc sale}
Document 1 = {computer sale}
Document 2 = {laptop computer pc}

Spelling Correction



albert einstein	4834
albert einstien	525
albert einstine	149
albert einsten	27
albert einsteins	25
albert einstain	11
albert einstin	10
albert eintein	9
albeart einstein	6
aolbert einstein	6
alber einstein	4
albert einseint	3

Counts of different (mis)spellings of Albert Einstein's name in a web query log.
Cucerzan and Brill, EMNLP 2004

Spelling Correction Task

- Input misspelled word: "alber einstien"
- Output strings that are:
  - Correct (in dictionary or frequently used); and
  - Close enough to the input

"Did you mean: *Albert Einstein*?"

Levenshtein Edit Distance

- The smallest number of single-character edits needed to transform one sequence to another.
- Single-character edits are defined as: 1) Insertions, 2) Deletions, or 3) Substitutions
- Examples:
  - “computing” → “computer”
    - at least 3 edits needed: 1 deletion and 2 substitutions
    - Edit distance = 3
  - “counter” → “computer”
    - at least 3 edits needed: 1 deletion and 2 insertions;
    - or 1 insertion and 2 substitutions
    - Edit distance = 3

How to Calculate Edit Distance?

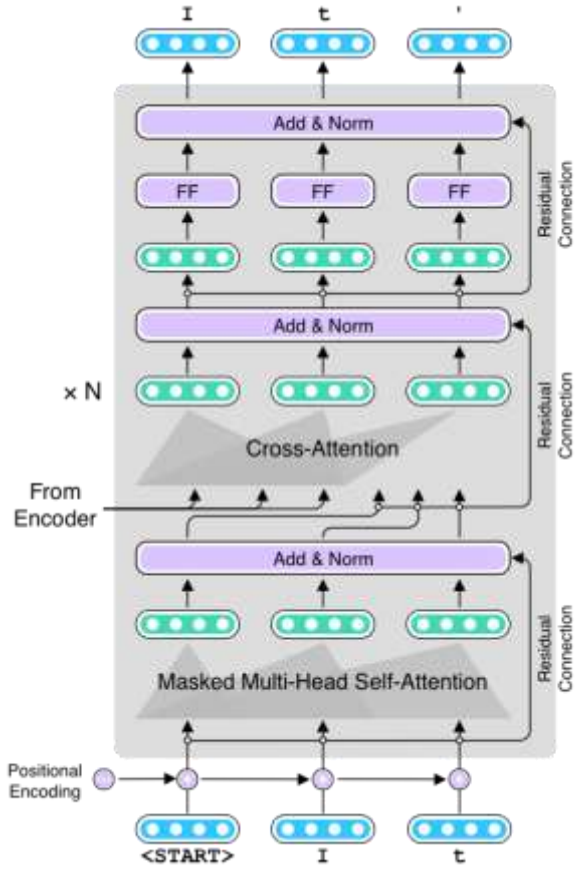
- Easy to find “one” transformation, but difficulty to find one with the “smallest” number of edits
  - e.g., “counter” → “computer” (5 edits)

- Solution: Dynamic Programming
  - Break down a complex problem into simpler sub-problems in a recursive manner
  - Store solutions for sub-problems
  - Use these solutions to find the solution of a complex problem

Weighted Edit Distance

- Used to emphasize the relative cost of different edit operations
- Typing/phonetic aware edit mistakes
  - “a” → “s” < “a” → “k”
  - “o” → “u” < “o” → “m”

Of course there is a deep learning based solution for this



- Possibly the *opposite* of dynamic programming in terms of efficiency
- Unreasonably effective

https://github.com/mhagiwara/xfspell

What's a good "basic concept"?

- Many possibilities: Words, stemmed words, phrases, "latent concept", …
- Orthogonal
  - Linearly independent
  - Non-overlapping in meaning
  - Related challenge: typos
- No ambiguity
- Weights can be assigned au
  hopefully accurately

Query = {Michael Jordan}
Document 1 = {Air Jordan Shoes}
Document 2 = {Michael Jordan Homepage}

Weighting a Vector Space

How to Assign Weights?

- Very very important!
- Why weighting
  - Query side: Not all terms are equally important
  - Doc side: Some terms carry more information about contents
- How?
  - Two basic heuristics
    - TF (Term Frequency) = Within-doc-frequency
    - IDF (Inverse Document Frequency)
  - TF normalization

What's a Reasonable Term Weight?

- How do we weight the multiple occurrences of a term in a document?
  - By Term Frequency (TF)

- How do we weight different terms differently?

Term Frequency (TF) Weighting

- Idea: A term is more important if it occurs more frequently in a document
- Formulas:
  - *c(t,d):* the frequency count of term *t* in doc *d*
  - Raw TF: TF(t, d) = c(t, d)
- We always need to normalize the raw TF
  - Log TF: TF(t, d) = log(c(t, d) + 1) （+1避免log0）
  - Maximum frequency normalization:

$$TF(t,d) = 0.5 + \frac{0.5 \times c(t,d)}{MaxFreq(d)}$$

Term Frequency (TF) Normalization

- TF Normalization can be even more complicated (next week)
- TF in Okapi/BM25:

$$TF(t,d) = \frac{k \times c(t,d)}{c(t,d) + k(1 - b + b \times length(d)/avg\_dl)}$$

Where length(d) is the length of d and avg_dl is the average length of documents in the collection

Why do we need to normalize TF?

- “Repeated occurrences” are less informative than the “first occurrence”

核心意思

•如果一个词在文档里出现 **很多次**，它的信息量（informativeness）并不会随着次数线性增加。

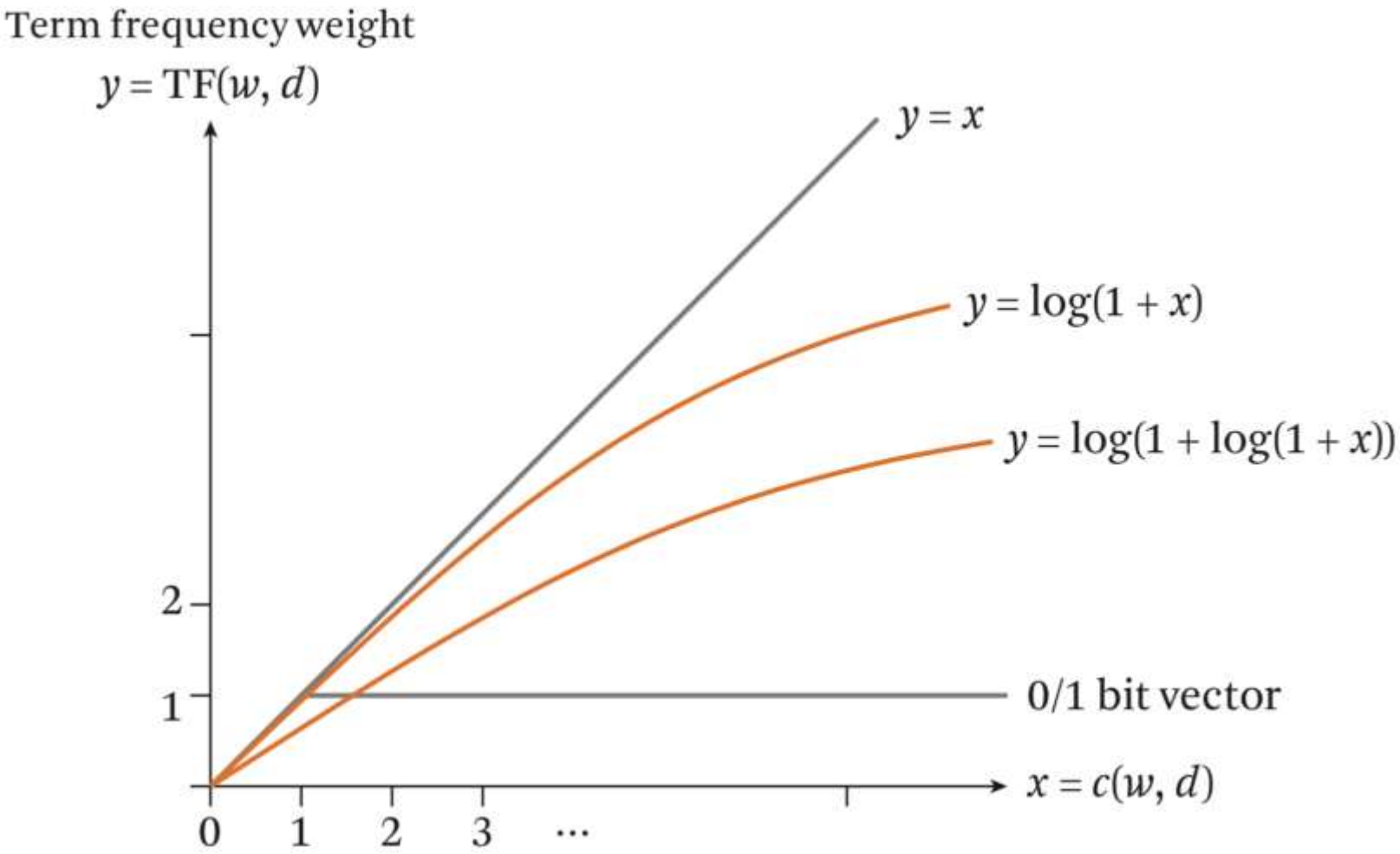
•第一次出现某个词，告诉我们“这篇文档和这个词相关”。

•但后面重复出现的次数，带来的新增信息量会越来越少。

•“归一化 TF” 的目的就是 **减弱高频词的权重膨胀**，因为 **重复出现的信息量小于首次出现**。

Why do we need to normalize TF?

TF Transformation: c(*w, d*) → TF(*w, d*)



Why do we need to normalize TF?

- “Repeated occurrences” are less informative than the “first occurrence”
- Document length variation

- Two views of document length
  - A doc is long because it uses more words
  - A doc is long because it has more contents
- Generally we want to penalize long documents, but avoid over-penalizing

What's a Reasonable Term Weight?

- How do we weight the multiple occurrences of a term in a document?
  - By Term Frequency (TF)

- How do we weight different terms differently?
  - By Inverted Document Frequency (IDF)

Inverted Document Frequency (IDF) Weighting （衡量某个词在整个语料库中的“普遍性/稀有性”。）

- Idea: A term is more discriminative if it occurs only a few documents
  - Why might this be true?
- Formula:

$$IDF(t) = 1 + log(\frac{n}{k})$$

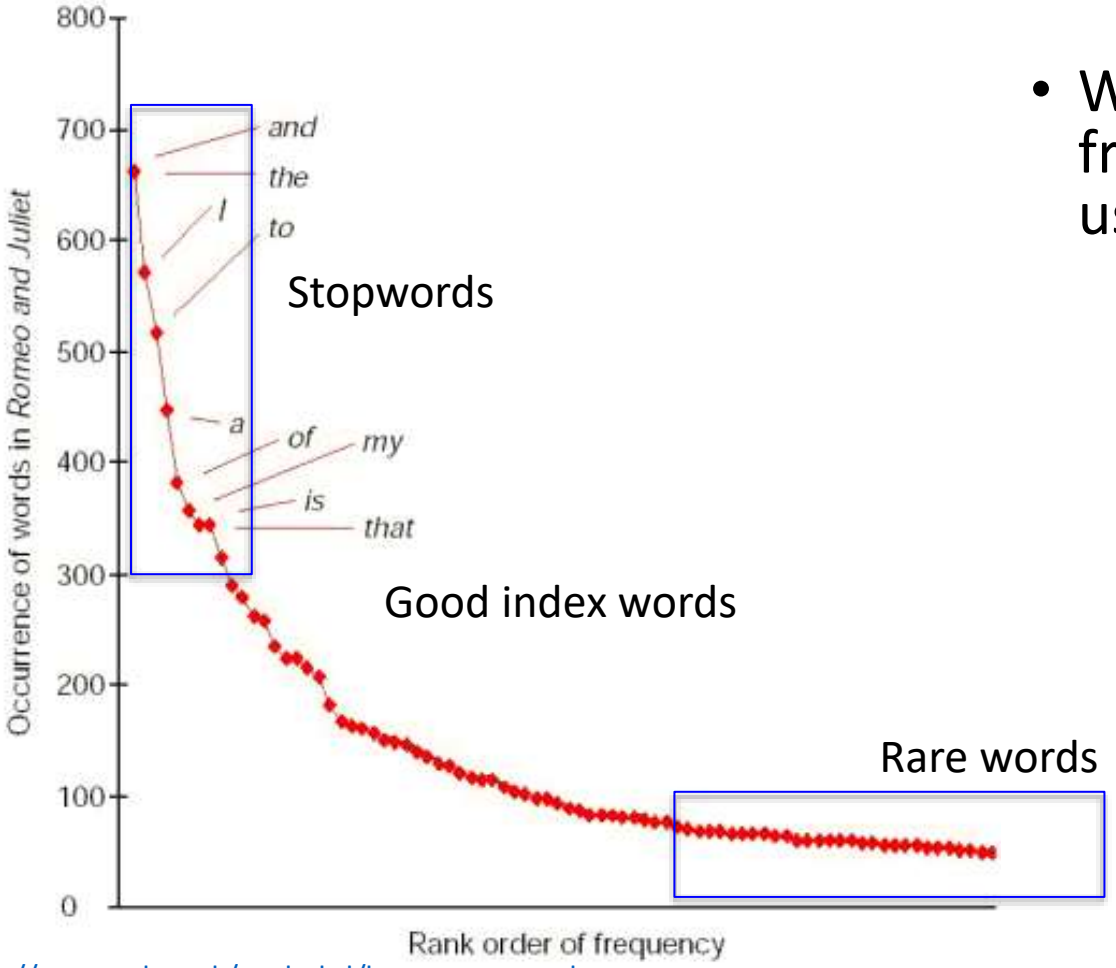
  - n — the total number of documents in the collection
  - k — the number of documents with term t (document frequency)

Note that IDF is document independent while TF is document dependent!

TF-IDF Weighting（特征权重计算方法，用来衡量某个词对于一篇文档和整个语料库的重要程度。）

- TF-IDF weighting:
  - weight(t, d)=TF(t, d)*IDF(t)
  - Term is common in doc → high tf → high weight
  - Term is rare in collection→ high idf→ high weight
- Imagine a word count profile, what kind of terms would have high weights?
- How is this related to Zipf's law?

How is TF-IDF related to Zipf's law?



- Words with middle frequency are most useful (Luhn)
  - Not too specific (low utility, but still useful!)
  - Not too general (lack of discrimination, stop words)

- http://www.ucl.ac.uk/~ucbplrd/language_page.htm

Alternative TF-IDF Weighting Schemes

- Many search engines allow for different weightings for queries vs. documents:
- A very standard weighting scheme is:
  - Document: logarithmic tf, no idf, and cosine normalization
  - Query: logarithmic tf, idf, no normalization

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

How to Measure Similarity with TF-IDF weighted vectors?

Now we know:

D = (w_{d1}, w_{d2}, …, w_{dN});

Q = (w_{q1}, w_{q2}, …, w_{qN});

w = tf * idf, or 0 if term is absent

Dot product similarity (in IR, we usually use dot product for efficiency)

$$sim(Q, D) = \sum_{j=1}^{N} w_{dj} \cdot w_{qj}$$

Cosine similarity (= normalized dot product)

$$sim(Q, D) = \frac{\sum_{j=1}^{N} w_{dj} \cdot w_{qj}}{\sqrt{\sum_{j=1}^{N} w_{dj}^2 \cdot \sum_{j=1}^{N} w_{qj}^2}}$$

Alternative Term Weighting: Keyword Discrimination

如果移除某个词后，文档之间的平均相似度 显著变化，说明这个词对区分文档很重要。这样的词就是一个 好 discriminator（区分性强的词），应该给予更高权重。

Keyword Discrimination Model

- The Vector representation of documents can be used as the source of another approach to term weighting
  - **Question**: what happens if we removed one of the words used as dimensions in the vector space?
  - If the average similarity among documents changes significantly, then the word was a good discriminator
  - If there is little change, the word is not as helpful and should be weighted less
- Note that the goal is to have a representation that makes it easier for a queries to discriminate among documents
- Average similarity can be measured after removing each word from the matrix
  - Any of the similarity measures can be used (we will look at a variety of other similarity measures later).

See additional slides if you would like to see details

Evaluating IR Systems

Evaluation Criteria

- Effectiveness/Accuracy
  - Precision, Recall
- Efficiency
  - Space and time complexity
- Usability
  - How useful for real user tasks?

Methodology: Cranfield Tradition

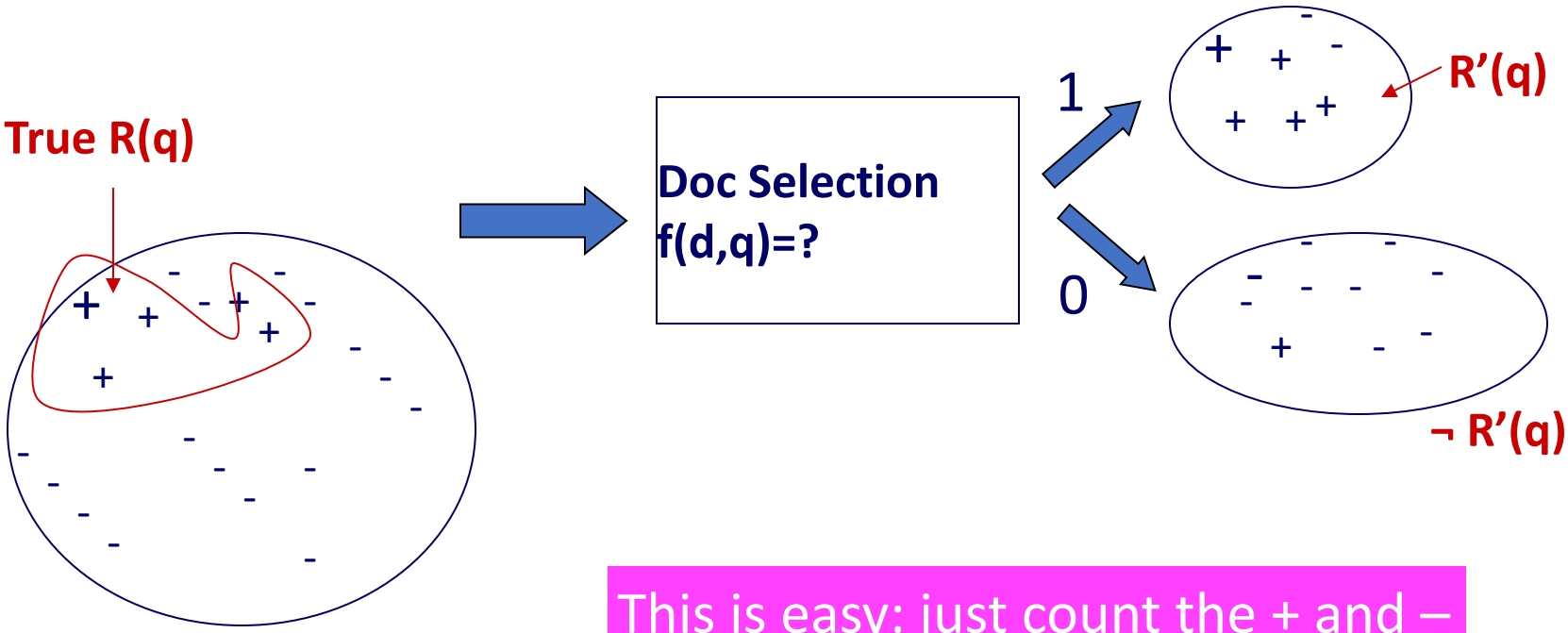
- Laboratory testing of system components
  - Precision, Recall
  - Comparative testing
- Test collections
  - Set of documents
  - Set of questions
  - Relevance judgments

Evaluation will depend on your definition of relevance.

Think back to earlier in this lecture… We talked about two ways to define relevance.

- Strategy 1: Document selection
  - System must decide if a doc is relevant or not (“absolute relevance”)
- Strategy 2: Document ranking
  - System must decide if one doc is more likely to be relevant than another (“relative relevance”)

1. Evaluating Document Selection



This is easy: just count the + and – in R'(q) and ¬ R'(q)

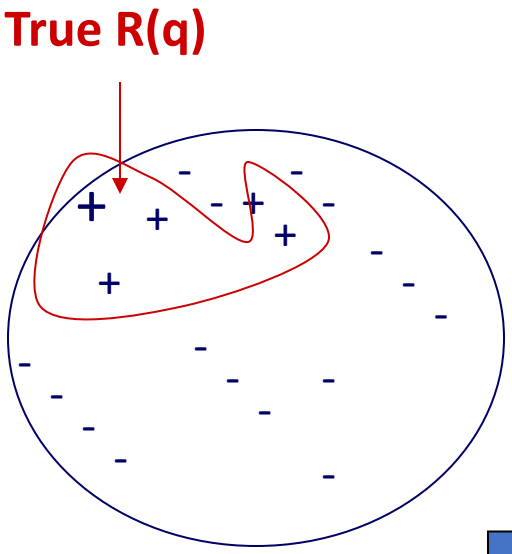
The Contingency Table

Doc \ Action	Retrieved R'(q)	Not Retrieved ¬ R'(q)
Relevant +	Relevant Retrieved	Relevant Rejected
Not relevant -	Irrelevant Retrieved	Irrelevant Rejected

$$\textbf{Precision} = \frac{\textbf{Relevant Retrieved}}{\textbf{Retrieved}}$$

$$\textbf{Recall} = \frac{\textbf{Relevant Retrieved}}{\textbf{Relevant}}$$

2. Evaluation of Document Ranking



Ranking is much harder to evaluate…

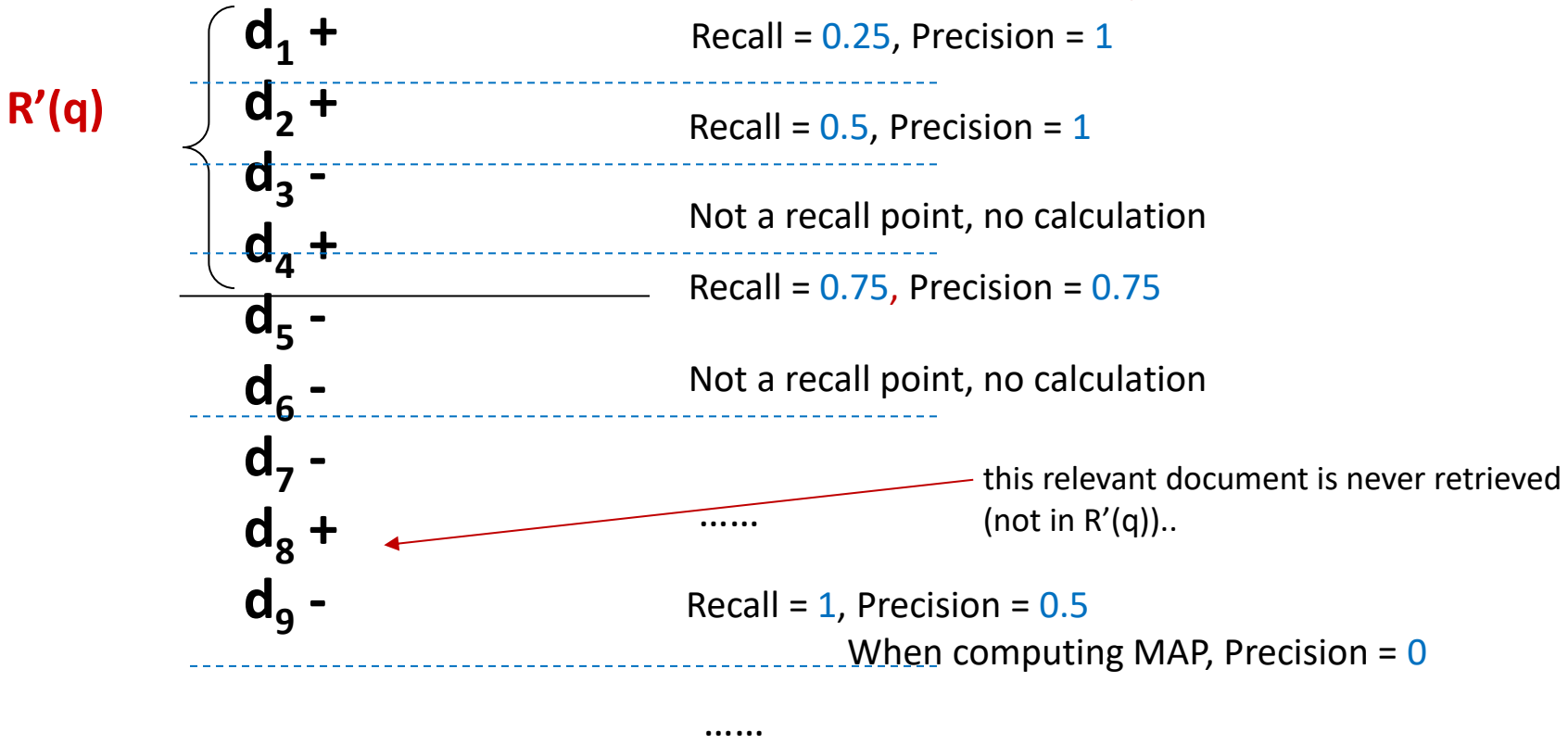
Doc Ranking
f(d,q)=?

0.98 d_1 +
0.95 d_2 +
0.83 d_3 -
0.80 d_4 +
0.76 d_5 -
0.56 d_6 -
0.34 d_7 -
0.21 d_8 +
0.21 d_9 -

R'(q)

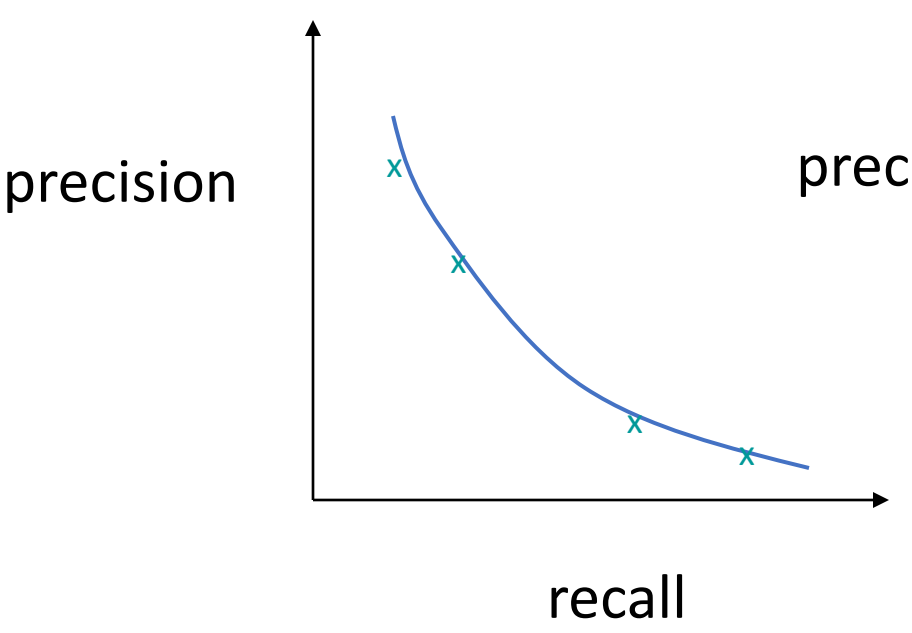
How to measure the goodness of a ranking?

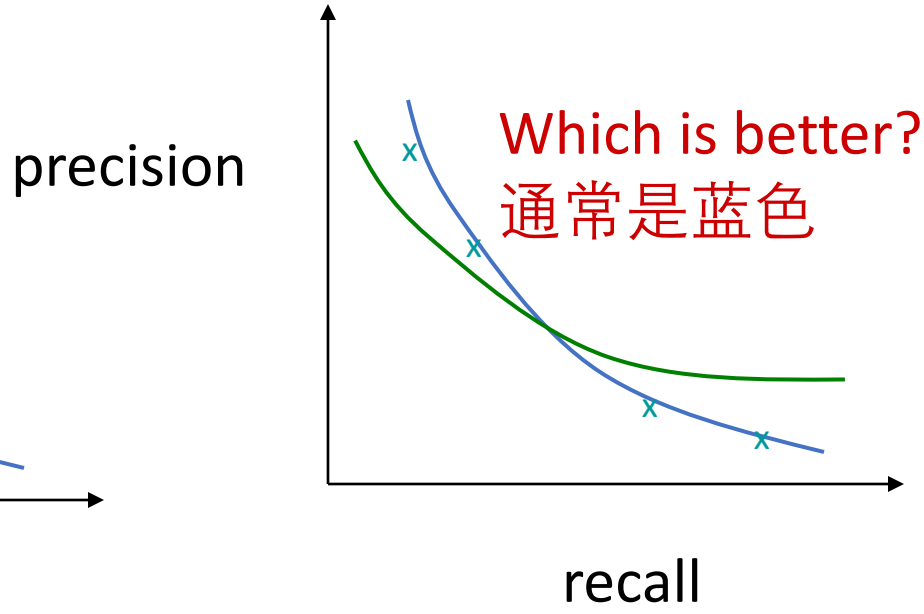
- Compute the precision at every recall point (the position where each relevant document is retrieved)



How to measure the goodness of a ranking?

- Compute the precision at every recall point
- Plot a precision-recall (PR) curve

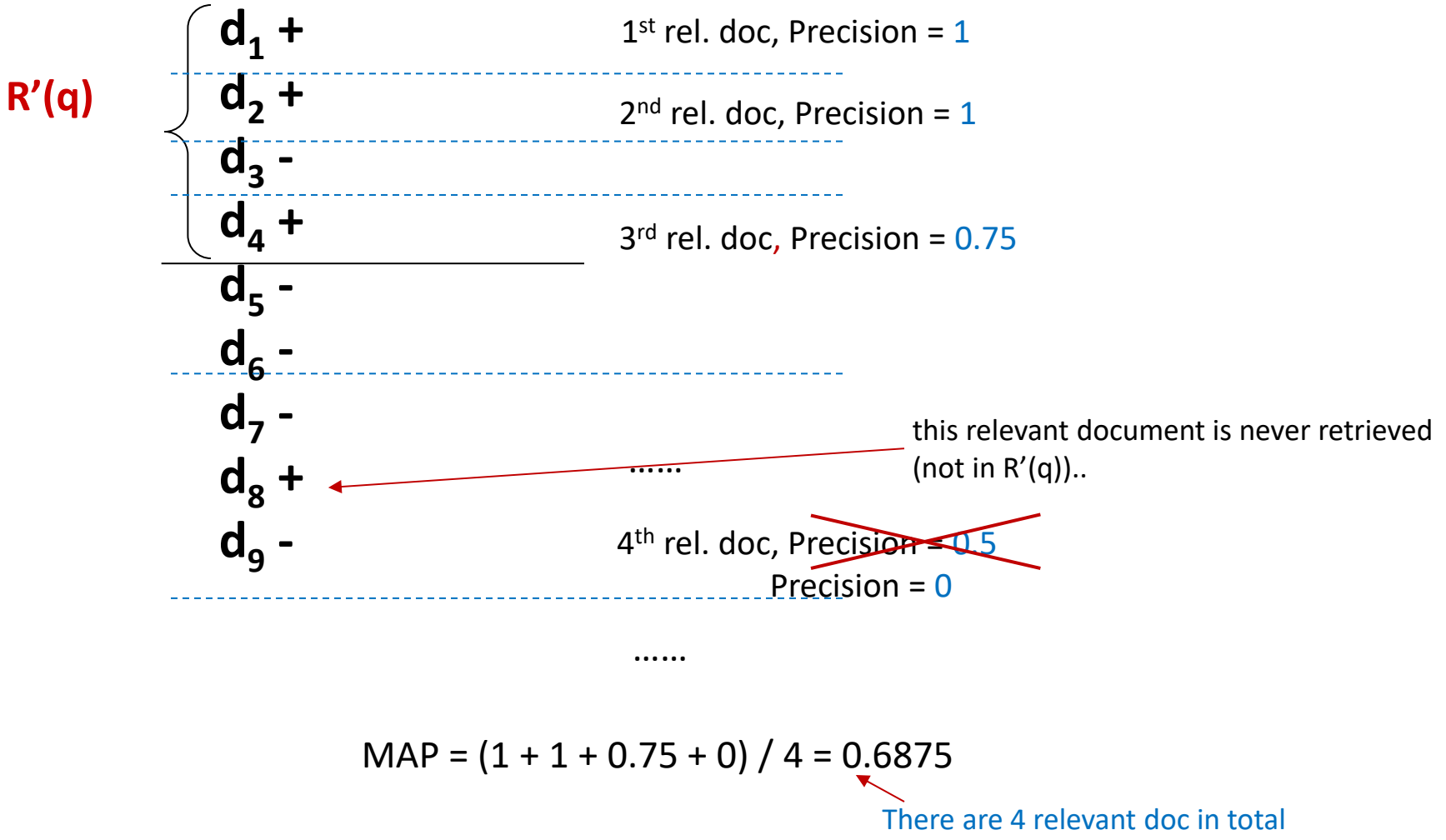




Summarize a Ranking: MAP

- Given that n docs are retrieved
  - Compute the precision (at rank) where each (new) relevant document is retrieved => p(1),…,p(k), if we have k rel. docs
  - E.g., if the first rel. doc is at the 2nd rank, then p(1)=1/2.
  - If a relevant document never gets retrieved, we assume the precision corresponding to that rel. doc to be zero
- Compute the average precision over all the relevant documents
  - Average precision = (p(1)+…p(k))/k
- This gives us an average precision, which captures both precision and recall and is sensitive to the rank of each relevant document
- Mean Average Precisions (MAP)
  - MAP = arithmetic mean average precision over a set of queries
  - gMAP = geometric mean average precision over a set of queries (more affected by difficult queries)

Computing MAP



Summarize a Ranking: DCG

- What if relevance judgments are in a scale of [1,r]? r>2
- Cumulative Gain (CG) at rank n
  - Let the ratings of the n documents be r_1, r_2, …r_n (in ranked order)
  - CG = r_1+r_2+…r_n
- Discounted Cumulative Gain (DCG) at rank n(更关心前面的文档)
  - DCG = $r_1 + r_2/\log_2 2 + r_3/\log_2 3 + \ldots r_n/\log_2 n$
  - We may use any base for the logarithm, e.g., base=b
  - For rank positions above b, do not discount

Summarize a Ranking: NDCG

- Normalized Cumulative Gain (NDCG) at rank n
  - Normalize DCG at rank n by the DCG value at rank n of the ideal ranking
  - The ideal ranking would first return the documents with the highest relevance level, then the next highest relevance level, etc
  - Compute the precision (at rank) where each (new) relevant document is retrieved => p(1),…,p(k), if we have k rel. docs
- NDCG is now quite popular in evaluating Web search

Other Measures

- Precision at k documents (e.g., prec@10doc):
  - more meaningful than MAP (why?)
  - also called breakeven precision when k is the same as the number of relevant documents
- F-Measure (F1): harmonic mean of precision and recall

$$F_\beta = \frac{1}{\frac{\beta^2}{\beta^2+1}P + \frac{1}{\beta^2+1}R} = \frac{(\beta^2+1)P*R}{\beta^2 P + R}$$

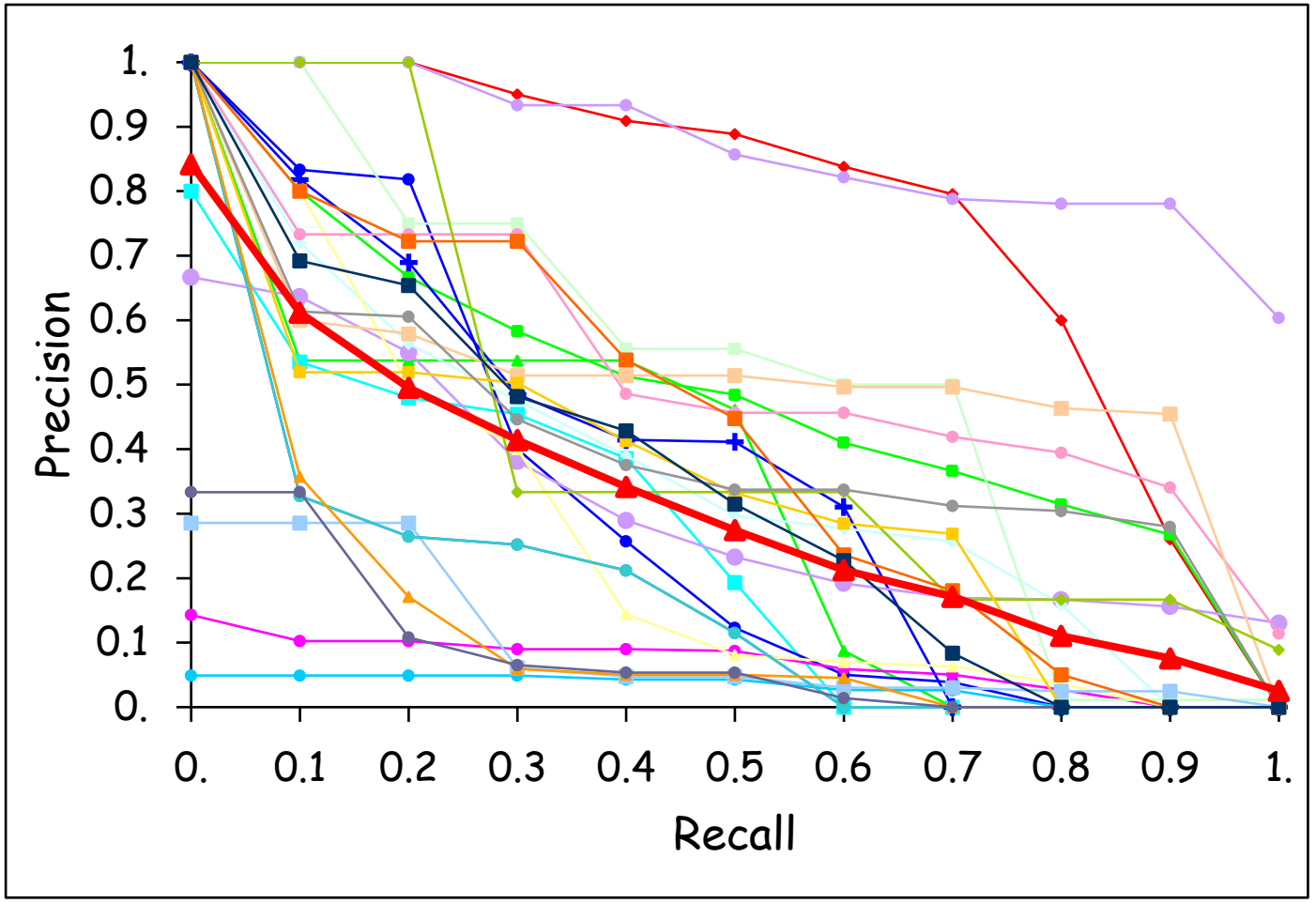
$$F_1 = \frac{2PR}{P+R}$$

P: precision

R: recall

β: parameter

What Query Averaging Hides



Slide from Doug Oard's presentation, originally from Ellen Voorhees' presentation

How to Obtain Relevance Judgements?

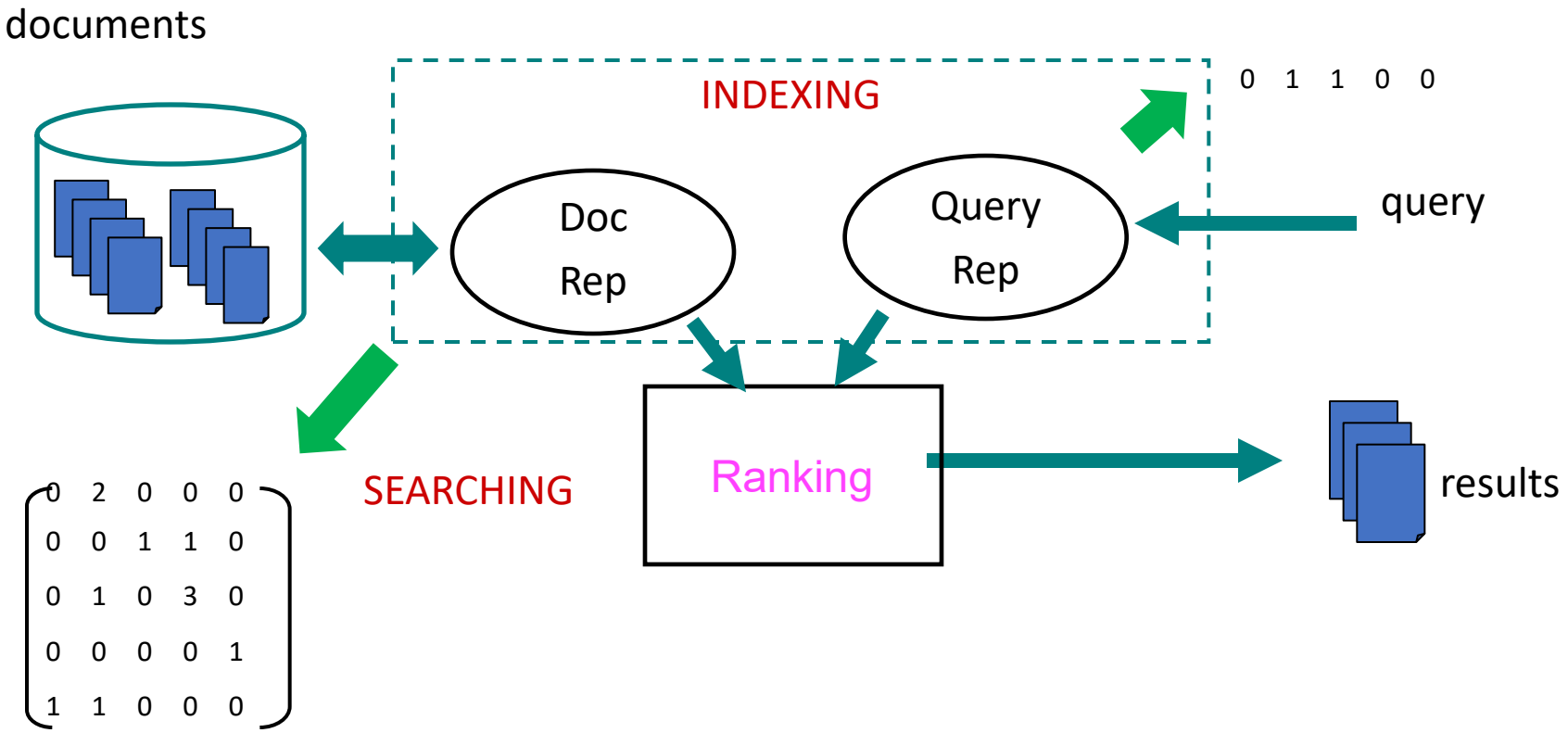
- Human annotation
  - Task: classification vs. rating vs. ranking
  - Multiple raters
  - Inter-rater agreements (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/)
    - Cohen's kappa (two raters)
    - Fleiss' kappa (multiple raters)
- Crowdsourcing
  - interfaces, incentives, many raters, rating aggregation, reliability of raters, …

Pooling

- A standard design of TREC
- Pooling the results of participating systems
  - E.g., 20 candidate systems, each submitting top 10 results per query
  - Ended up with 10 - 200 docs after pooling
  - Sent to human annotators
  - Documents not in the pool are considered irrelevant.
- These judgments are then used to evaluate “old” and “new” systems (favors participating sys.).

Indexing

Let's Start from Here…



Reminder: Text Representation/Indexing

- Making it easier to match a query with a document
- Query and document should be represented using the same units/terms
- Controlled vocabulary vs. full text indexing
- Full-text indexing is more practically useful and has proven to be as effective as manual indexing with controlled vocabulary

Handling large collections

- Life is good when every document is mapped into a vector of words, but …
- Consider *N* = 1 million documents, each with about 1000 words.
- Avg. 6 bytes/word including spaces/punctuation
  - 6GB of data in the documents.
- Say there are *M* = 500K *distinct* terms among these.

Storage issue

- 500K x 1M matrix has half-a-trillion elements.
  - 4 bytes for an integer
  - 500K x 1M x 4 = 2T (your laptop would fail)
  - 500K x 100B x 4 = $2*10^5$ T (challenging even for google)
- But it has no more than one billion positive numbers.
  - matrix is extremely sparse.
  - 1000 x 1M x 4 = 4G
- What's a better representation?

Another Motivation…

- Let's look at the simplest case (boolean retrieval)
- Query = "school" + "information"
  - The only thing we need to do is to return all the documents containing both the term "school" and the term "information"
- Simple strategy:
  - Scan each document and output anyone who contains both words.
  - Can we afford to scan all the documents for each query?
- Any faster way to do it?

Indexing

- Indexing = Convert documents to data structures that enable fast search
- Inverted index is the dominating indexing method (used by all search engines)
- Other indices (e.g., document index) may be needed for feedback

Inverted index（用于存储从 词 → 文档列表 的映射）

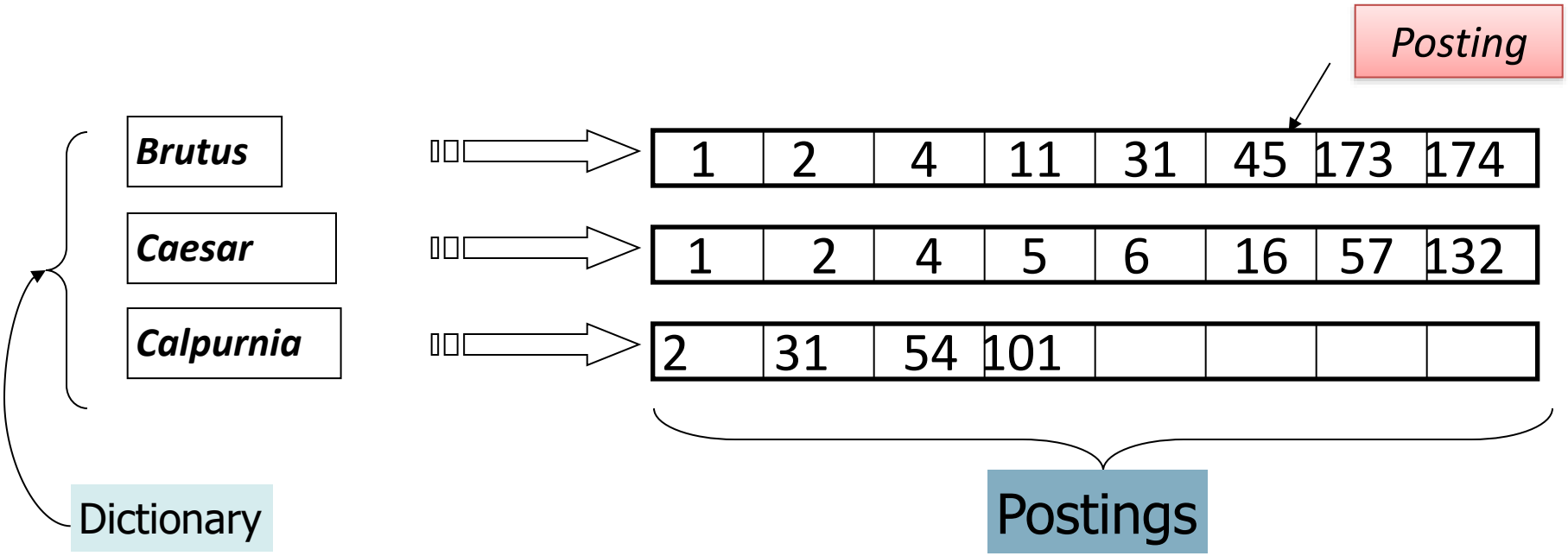
- Instead of an incidence vector, use a posting table
  - CLEVELAND: D1, D2, D6
  - OHIO: D1, D5, D6, D7
- Use linked lists to be able to insert new document postings in order and to remove existing postings.
- More efficient than scanning docs (why?)

Inverted index

- Fast access to all docs containing a given term (along with frequency and position information)
- For each term, we get a list of tuples
  - (docID, freq, pos).
- Given a query, we can fetch the lists for all query terms and work on the involved documents.
  - Boolean query: set operation
  - Natural language query: term weight summing
- Keep everything sorted! This gives you a logarithmic improvement in access.

Inverted index - example

- For each term *t*, we must store a list of all documents that contain *t*.
  - Identify each by a docID, a document serial number



- From Chris Manning's slides

Inverted index - example

Doc 1

This is a sample document with one sample sentence

Doc 2

This is another sample document

Dictionary

Term	# docs	Total freq
This	2	2
is	2	2
sample	2	3
another	1	1
…	…	…

Postings

Doc id	Freq	Pos
1	1	1
2	1	1
1	1	2
2	1	2
1	2	4, 8
2	1	4
2	1	3
…	…	…
…	…	…

- From ChengXiang Zhai's slides

Boolean operations on inverted indexes

- Conjunction (AND) – iterative merge of the two postings: O(x+y) （x, y是两个posting的长度）
- Disjunction (OR) – very similar
- Negation (NOT) – can we still do it in O(x+y)?
  - Example: MICHIGAN AND NOT OHIO
  - Example: MICHIGAN OR NOT OHIO
- Recursive operations
- Optimization: start with the smallest sets

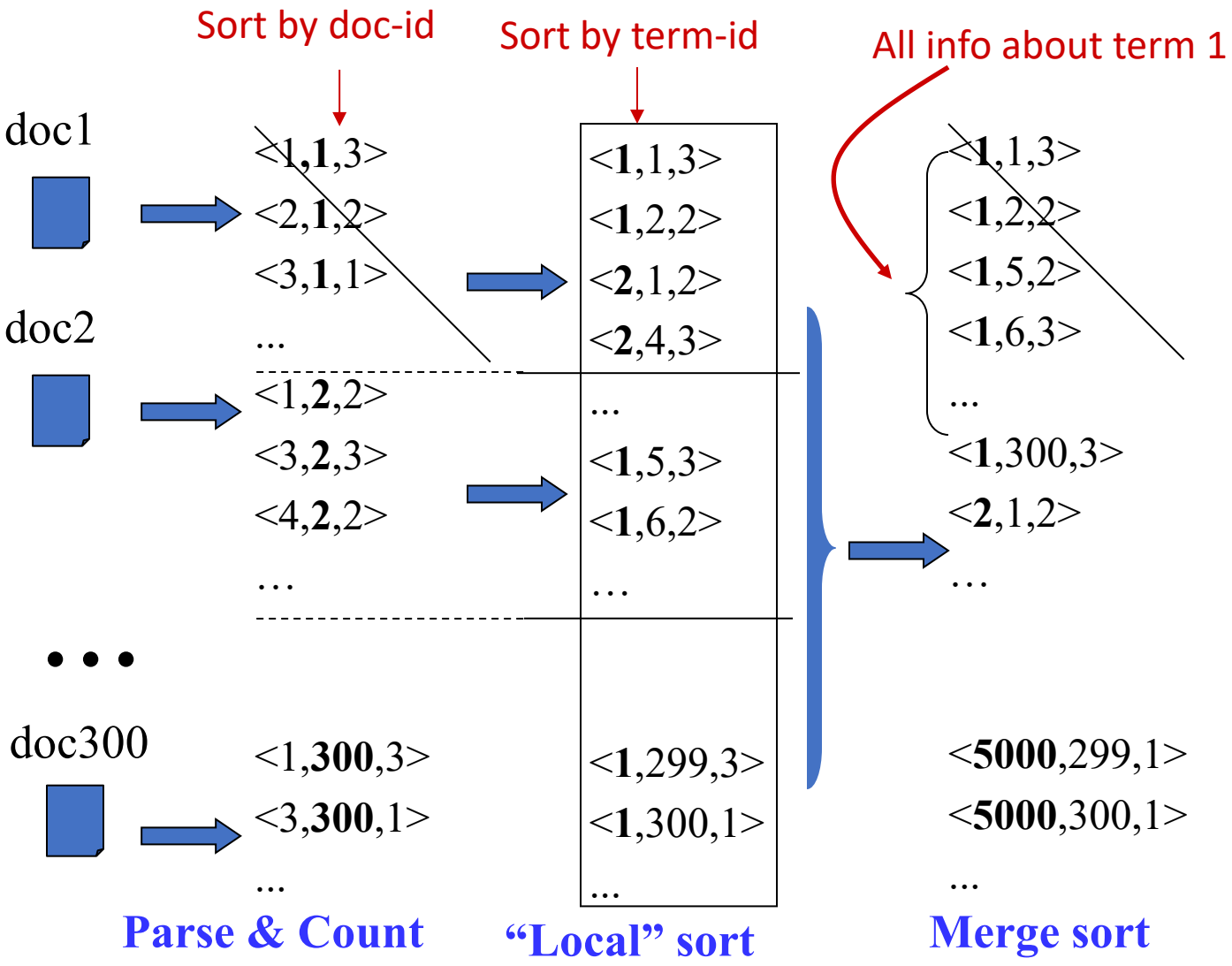
Data structures for inverted index

- Dictionary: modest size
  - Needs fast random access
  - Preferred to be in memory
  - Hash table, B-tree, trie, …
- Postings: huge
  - Sequential access is expected
  - Can stay on disk
  - May contain docID, term freq., term pos, etc
  - Compression is desirable

Constructing inverted index

- The main difficulty is to build a huge index with limited memory
- Memory-based methods: not usable for large collections
- Sort-based methods:
  - Step 1: collect local (termID, docID, freq) tuples
  - Step 2: sort local tuples (to make “runs”)
  - Step 3: pair-wise merge runs
  - Step 4: Output inverted file

Sort-based inversion



Term Lexicon:

the 1
cold 2
days 3
a 4
…

DocID Lexicon:

doc1 1
doc2 2
doc3 3
…

Searching

- Given a query, how to score documents efficiently?
- Boolean query
  - Fetch the inverted list for all query terms
  - Perform set operations to get the subset of docs that satisfy the Boolean condition
  - E.g., Q1="info" AND "security" , Q2="info" OR "security"
    - info: d1, d2, d3, d4
    - security: d2, d4, d6
    - Results: {d2,d4} (Q1) {d1,d2,d3,d4,d6} (Q2)

Ranking Documents

- Assumption:

  score(d,q)=f [g(w(d,q,t_1), …, w(d,q,t_n)), w(d), w(q)],

  - t_i :matched terms; w: weights; g: some accumulation
- Maintain a score accumulator for each doc to compute function g
- For each query term t_i
  - Fetch the inverted list {(d_1,f_1),…,(d_n,f_n)}
  - For each entry (d_j,f_j), Compute $w(d_j,q,t_i)$, and Update score accumulator for doc d_i
- Adjust the score to compute f, and sort

Ranking Documents: Example

$S(d,q)=g(t_1)+\ldots+g(t_n)$ [sum of freq of matched terms]
Query = "info security"
➔ S(d, q) = g("info") + g("security")

Info: (d1, 3), (d2, 4), (d3, 1), (d4, 5)
Security: (d2, 3), (d4,1), (d5, 3)

Accumulators:		d1	d2	d3	d4	d5
		0	0	0	0	0
info	(d1,3) =>	**3**	0	0	0	0
	(d2,4) =>	3	**4**	0	0	0
	(d3,1) =>	3	4	**1**	0	0
	(d4,5) =>	3	4	1	**5**	0
security	(d2,3) =>	3	**7**	1	5	0
	(d4,1) =>	3	7	1	**6**	0
	(d5,3) =>	3	7	1	6	**3**

Ranking Documents: TF-IDF

$S(d,q)=g(t_1)+\ldots+g(t_n)$ [TF*IDF]

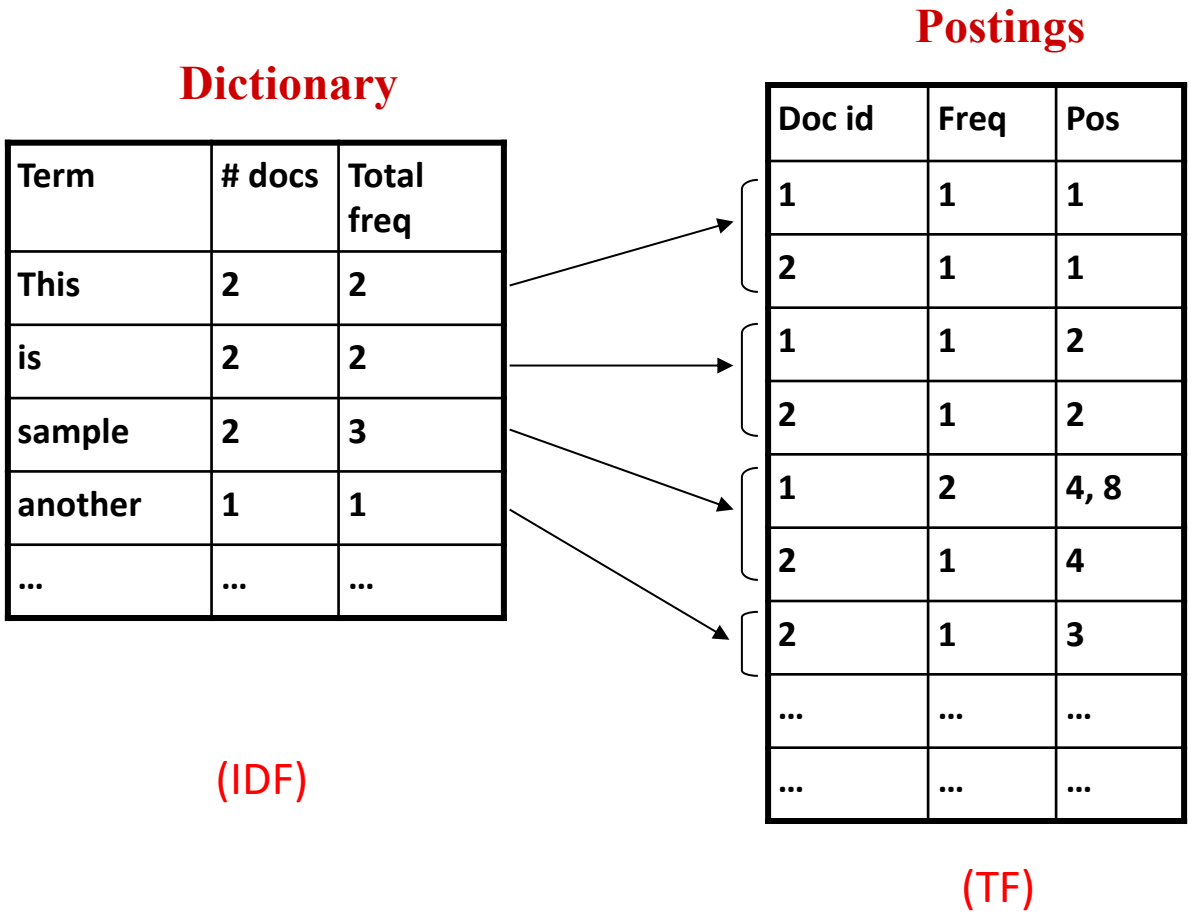
TF*IDF ~ freq of term t in d / number of docs with t

We will talk about the details next week.

Query = “info security”
➔ S(d, q) = g(“info”) + g(“security”)

TF-IDF and Inverted Index

- What is need to calculate TF-IDF?
- Can you find such information easily in an inverted index?

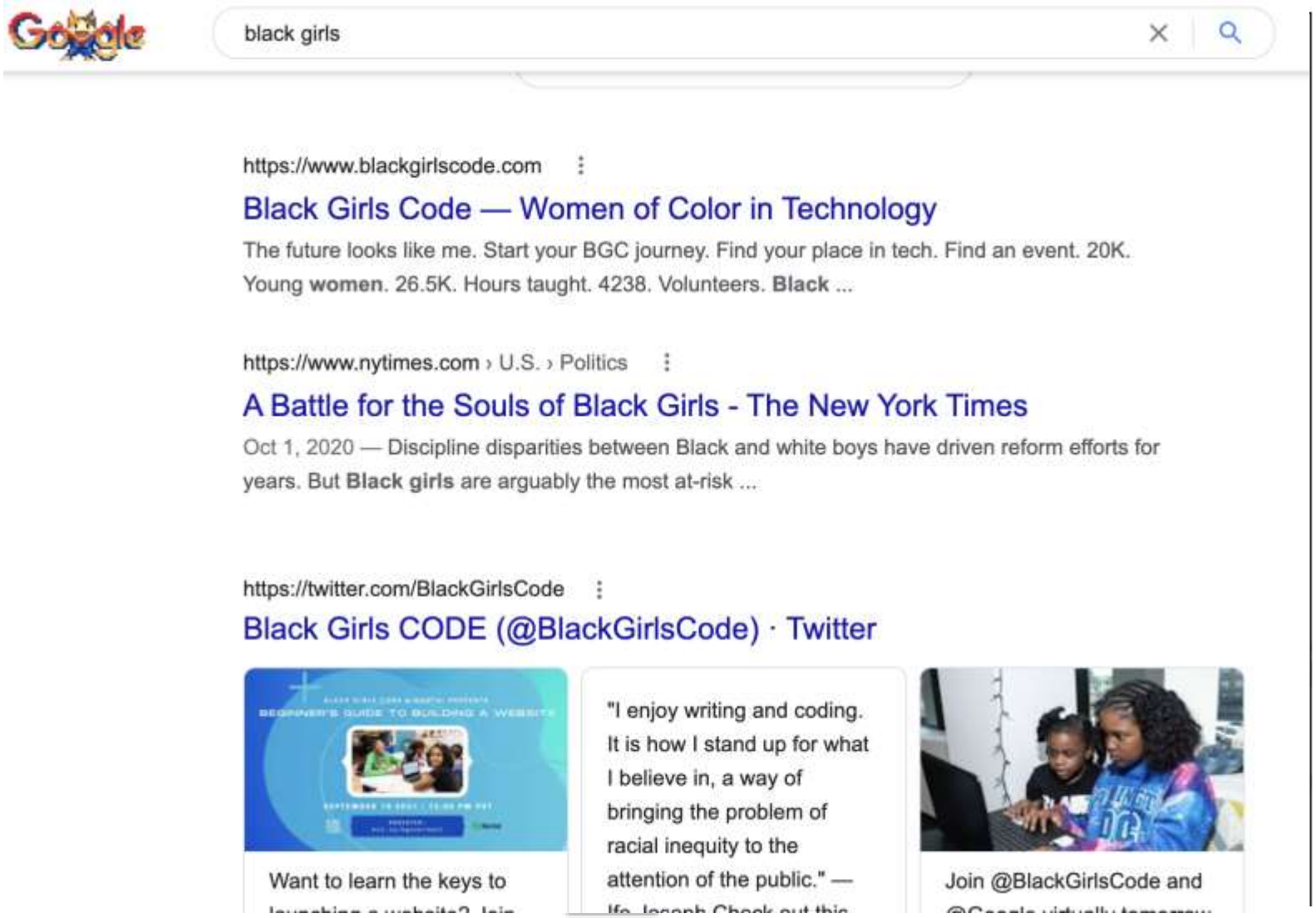


Further Improving Efficiency

- Keep only the most promising accumulators
- Sort the inverted list in decreasing order of weights and fetch only N entries with the highest weights
- Pre-compute as much as possible
- Scaling up to the Web-scale (more about this later)

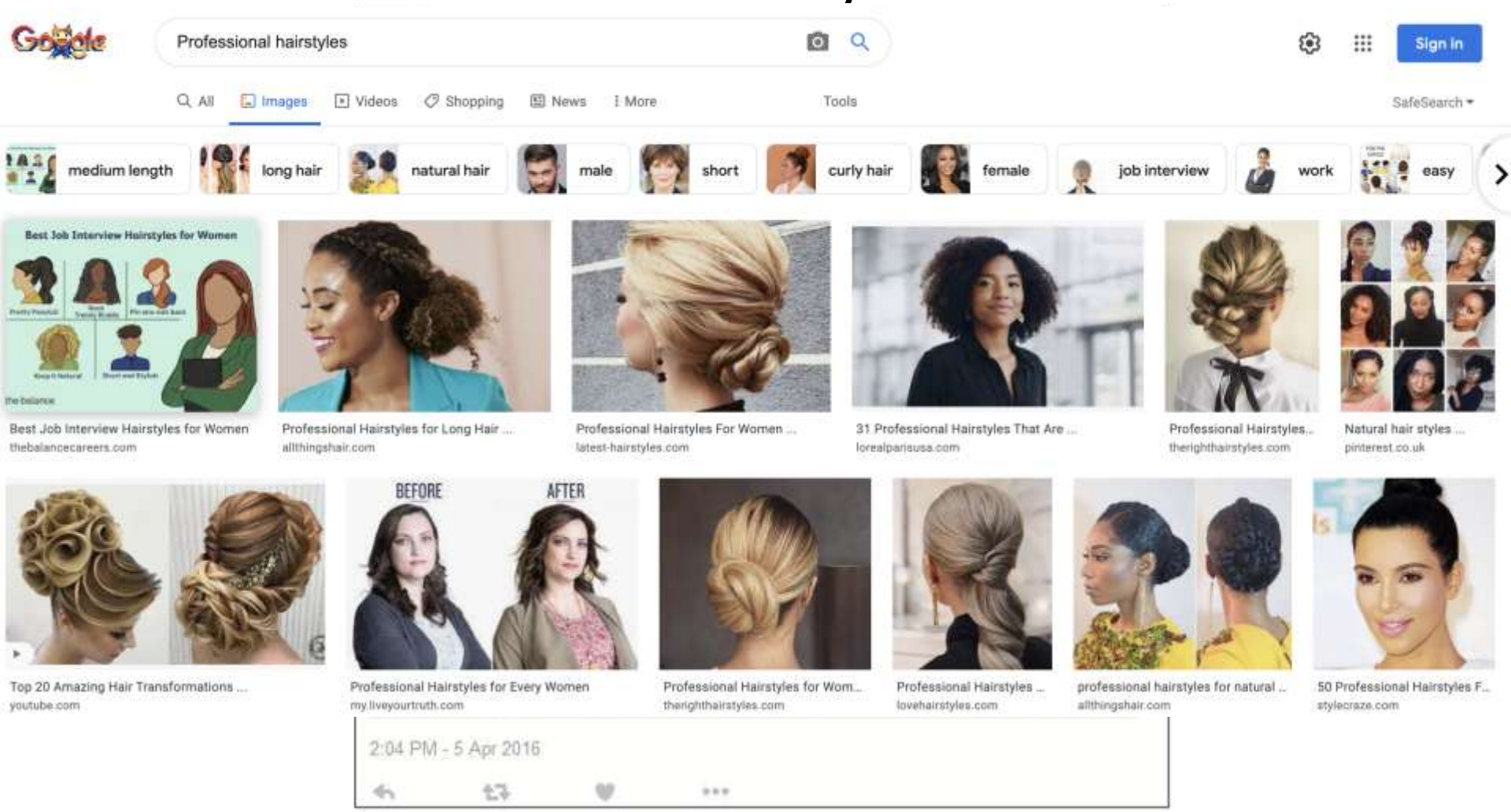
IR and Fairness

Searching for “black girls”



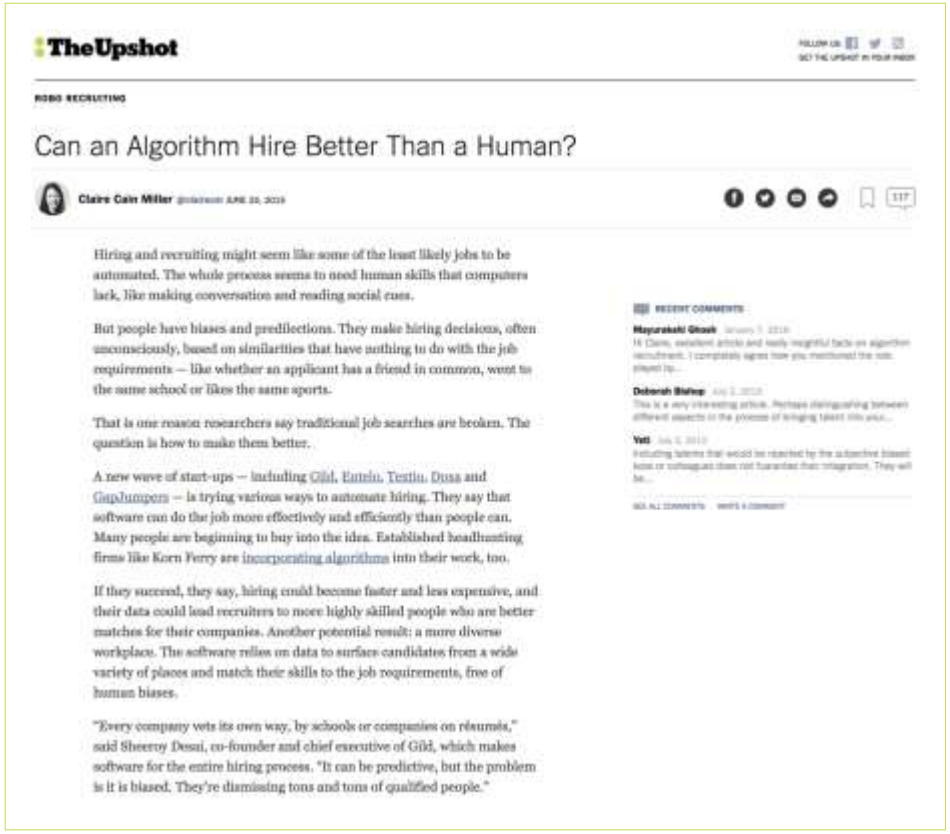
Safiya Umoja Noble. Algorithms of Oppression : How Search Engines Reinforce Racism, New York University Press, 2018.

“Professional hairstyles”



Safiya Umoja Noble. Algorithms of Oppression : How Search Engines Reinforce Racism, New York University Press, 2018.

IR bias can have huge implications



The Upshot

ROBO RECRUITING

Can an Algorithm Hire Better Than a Human?

Claire Cain Miller

Hiring and recruiting might seem like some of the least likely jobs to be automated. The whole process seems to need human skills that computers lack, like making conversation and reading social cues.

But people have biases and predilections. They make hiring decisions, often unconsciously, based on similarities that have nothing to do with the job requirements — like whether an applicant has a friend in common, went to the same school or likes the same sports.

That is one reason researchers say traditional job searches are broken. The question is how to make them better.

A new wave of start-ups — including Gild, Entelo, Textio, Doxa and GapJumpers — is trying various ways to automate hiring. They say that software can do the job more effectively and efficiently than people can. Many people are beginning to buy into the idea. Established headhunting firms like Korn Ferry are incorporating algorithms into their work, too.

If they succeed, they say, hiring could become faster and less expensive, and their data could lead recruiters to more highly skilled people who are better matches for their companies. Another potential result: a more diverse workplace. The software relies on data to surface candidates from a wide variety of places and match their skills to the job requirements, free of human biases.

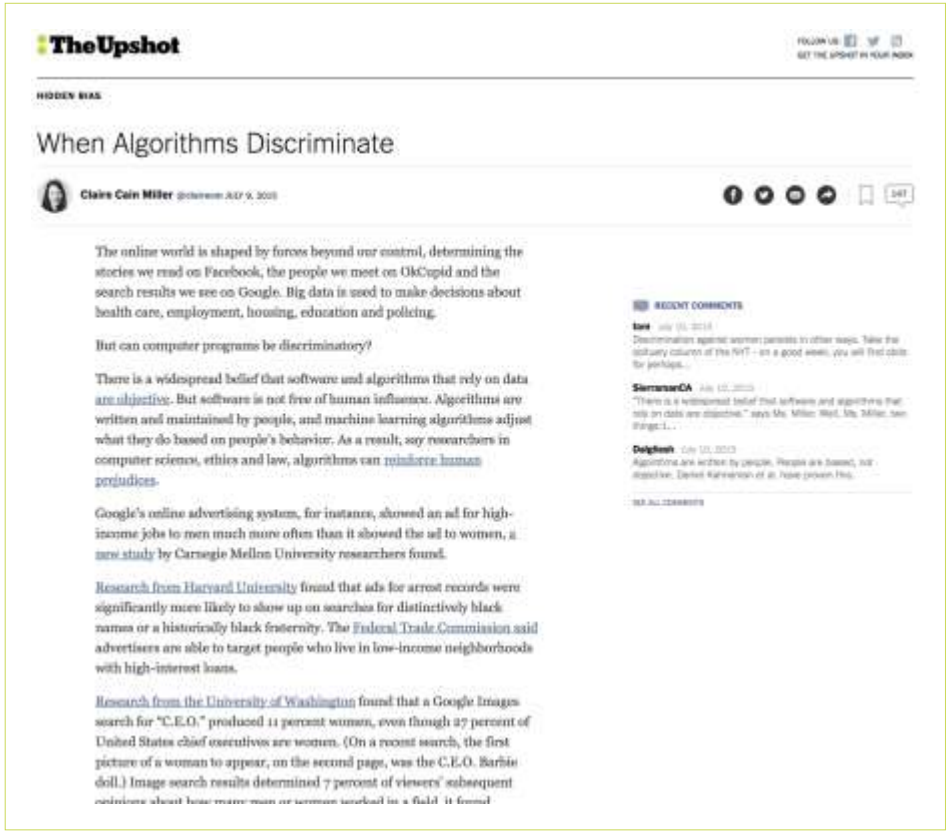
"Every company vets its own way, by schools or companies on résumés," said Sheeroy Desai, co-founder and chief executive of Gild, which makes software for the entire hiring process. "It can be predictive, but the problem is it is biased. They're dismissing tons and tons of qualified people."

"[H]iring could become faster and less expensive, and […] lead recruiters to more highly skilled people who are better matches for their companies. Another potential result: a more diverse workplace. The software relies on data to surface candidates from a wide variety of places and match their skills to the job requirements, **free of human biases**."

Miller (2015)

Slide courtesy of Sarah Bird

IR bias can have huge implications



The Upshot

HIDDEN BIAS

When Algorithms Discriminate

Claire Cain Miller

The online world is shaped by forces beyond our control, determining the stories we read on Facebook, the people we meet on OkCupid and the search results we see on Google. Big data is used to make decisions about health care, employment, housing, education and policing.

But can computer programs be discriminatory?

There is a widespread belief that software and algorithms that rely on data are objective. But software is not free of human influence. Algorithms are written and maintained by people, and machine learning algorithms adjust what they do based on people's behavior. As a result, say researchers in computer science, ethics and law, algorithms can reinforce human prejudices.

Google's online advertising system, for instance, showed an ad for high-income jobs to men much more often than it showed the ad to women, a new study by Carnegie Mellon University researchers found.

Research from Harvard University found that ads for arrest records were significantly more likely to show up on searches for distinctively black names or a historically black fraternity. The Federal Trade Commission said advertisers are able to target people who live in low-income neighborhoods with high-interest loans.

Research from the University of Washington found that a Google Images search for "C.E.O." produced 11 percent women, even though 27 percent of United States chief executives are women. (On a recent search, the first picture of a woman to appear, on the second page, was the C.E.O. Barbie doll.) Image search results determined 7 percent of viewers' subsequent

RECENT COMMENTS

Iara

SierraneCA

Dolphish

"But software is not free of human influence. Algorithms are written and maintained by people, and machine learning algorithms adjust what they do based on people's behavior. As a result [...] **algorithms can reinforce human prejudices**."

Miller (2015)

Slide courtesy of Sarah Bird

Types of Harm

- **Harms of allocation**: withhold opportunity or resources (**（分配上的危害）定义**：当算法或系统在分配资源或机会时存在偏见，某些群体就可能被**排除**、**忽视**或**不公平对待**。)

- **Harms of representation**: reinforce subordination along the lines of identity, stereotypes(（**再现上的危害）定义**：当算法或系统在内容呈现上延续甚至强化了既有的**身份歧视、刻板印象或社会边缘化**。）

Slide courtesy of Sarah Bird

Legally Recognized Protected Classes

- Race (Civil Rights Act of 1964);
- Color (Civil Rights Act of 1964);
- Sex (Equal Pay Act of 1963; Civil Rights Act of 1964);
- Religion (Civil Rights Act of 1964);
- National origin (Civil Rights Act of 1964);
- Citizenship (Immigration Reform and Control Act);
- Age (Age Discrimination in Employment Act of 1967);
- Pregnancy (Pregnancy Discrimination Act);
- Familial status (Civil Rights Act of 1968);
- Disability status (Rehabilitation Act of 1973; Americans with Disabilities Act of 1990);
- Veteran status (Vietnam Era Veterans' Readjustment Assistance Act of 1974; Uniformed Services Employment and Reemployment Rights Act);
- Genetic information (Genetic Information Nondiscrimination Act)

Slide courtesy of Sarah Bird

Other non-protected categories to consider

- **Societal Categories**: i.e., political ideology, language, income, location, topical interests, (sub)culture, physical traits, etc.

- **Intersectional Subpopulations**: i.e., women from tech

- **Application-specific subpopulations**: i.e., device type

Slide courtesy of Sarah Bird

Search and IR have multiple motivations for improving fairness

- Better product and Serving Broader Population
- Responsibility and Social Impact
- Legal and Policy
- Competitive Advantage and Brand

Slide courtesy of Sarah Bird

Bias, Discrimination & Machine Learning

- **Isn't bias a technical concept?**
  - Selection, sampling, reporting bias, Bias of an estimator, Inductive bias

- **Isn't discrimination the very point of machine learning?**
  - Unjustified basis for differentiation

Slide courtesy of Sarah Bird

Good IR/ML Practices Go a Long Way

01

Lots of low hanging fruit in terms of improving fairness simply by using IR and machine learning best practices

- Representative data
- Introspection tools
- Visualization tools
- Testing

02

Fairness improvements often lead to overall improvements

- It's a common misconception that it's always a tradeoff

Slide courtesy of Sarah Bird

We'll need to analyze an IR system in multiple ways!

- Consider the complete system end-to-end including people, technology and processes
- Break your system into components to analyze their particular impact
- Make sure the data you are using to train and evaluate your system is not only a weak correlate of the outcome measure you care about [Zanger-Tischler et al., 2024]
- …
- Many others we will cover later this semester. And you will get to practice this hands on with Homework 5!

Risk Scores, Label Bias, and Everything but the Kitchen Sink.
Zanger-Tischler et al., 2024

What You Should Know

- How to calculate TF-IDF and why it's needed
- What is an inverted index
- Why does an inverted index help make search fast
- How to construct a large inverted index
- How to support phrase/position search
- IR systems have fairness issues

Extra Slides (Not covered in lecture) Alternative Term Weighting: Keyword Discrimination

Keyword Discrimination Model

- The Vector representation of documents can be used as the source of another approach to term weighting
  - **Question**: what happens if we removed one of the words used as dimensions in the vector space?
  - If the average similarity among documents changes significantly, then the word was a good discriminator
  - If there is little change, the word is not as helpful and should be weighted less
- Note that the goal is to have a representation that makes it easier for queries to discriminate among documents
- Average similarity can be measured after removing each word from the matrix
  - Any of the similarity measures can be used (we will look at a variety of other similarity measures later).

Keyword Discrimination

- Measuring average similarity (assume there are N documents)

  sim(D_1,D_2) = similarity score for pair of documents D_1 and D_2

$$\overline{sim} = \frac{1}{N^2}\sum_{i,j} sim(D_i, D_j)$$

$$\overline{sim_k} = \overline{sim} \text{ when } term_k \text{ removed}$$

Computationally Expensive

- Better way to calculate AVG-SIM
  - Calculate centroid D^* (avg. document vector = Sum vectors / N)
  - Then:

$$\overline{sim} = \frac{1}{N}\sum_{i} sim(D_i, D^*)$$

Keyword Discrimination

- Discrimination value (discriminant) and term weights

$$disc_k = \overline{sim_k} - \overline{sim}$$

$disc_k > 0 \rightarrow term_k$ is a good discriminant
$disc_k < 0 \rightarrow term_k$ is a poor discriminant
$disc_k = 0 \rightarrow term_k$ is indifferent

- Computing Term Weights
  - New weight for a term k in a document i is the original term frequency of k in i times the discriminant value:

$$w_{ik} = tf_{ik} \times disc_k$$

Keyword Discrimination - Example

docs	t1	t2	t3
D1	10	1	0
D2	9	2	10
D3	8	1	1
D4	8	1	50
D5	19	2	15
D6	9	2	0
D*	10.50	1.50	12.67

Doc-Sim to Centroid	
sim(D1,D*)	0.641
sim(D2,D*)	0.998
sim(D3,D*)	0.731
sim(D4,D*)	0.859
sim(D5,D*)	0.978
sim(D6,D*)	0.640
AVG-SIM	0.808

$$\overline{sim} = \frac{1}{N}\sum_{i} sim(D_i, D^*)$$

Using Normalized Cosine

$\overline{sim_k} = \overline{sim}$ when $term_k$ removed

sim1(D1,D*)	0.118	sim2(D1,D*)	0.638	sim3(D1,D*)	0.999
sim1(D2,D*)	0.997	sim2(D2,D*)	0.999	sim3(D2,D*)	0.997
sim1(D3,D*)	0.785	sim2(D3,D*)	0.729	sim3(D3,D*)	1.000
sim1(D4,D*)	0.995	sim2(D4,D*)	0.861	sim3(D4,D*)	1.000
sim1(D5,D*)	1.000	sim2(D5,D*)	0.978	sim3(D5,D*)	0.999
sim1(D6,D*)	0.118	sim2(D6,D*)	0.638	sim3(D6,D*)	0.997
SIM1	0.669	SIM2	0.807	SIM3	0.999

Note: D* for each of the SIM_k is now computed with only two terms

Keyword Discrimination - Example

$$disc_k = \overline{sim_k} - \overline{sim}$$

Term	$disc_k$
t1	-0.139
t2	-0.001
t3	0.191

This shows that ***t1*** tends to be a poor discriminator, while ***t3*** is a good discriminator. The new term weight will now reflect the discrimination value for these terms. Note that further normalization can be done to make all term weights positive.

	t1	t2	t3
D1	-1.392	-0.001	0.000
D2	-1.253	-0.001	1.908
D3	-1.114	-0.001	0.191
D4	-1.114	-0.001	9.538
D5	-2.645	-0.001	2.861
D6	-1.253	-0.001	0.000

New Weights for Terms t1, t2, and t3

$$w_{ik} = tf_{ik} \times disc_k$$

More on Bias/Fairness

Discrimination is not a general concept

It is domain specific

Concerned with important opportunities that affect people's life chances

It is feature specific

Concerned with socially salient qualities that have served as the basis for unjustified and systematically adverse treatment in the past

[Barocas & Hardt 2017]

Slide courtesy of Sarah Bird

Discrimination Law and Legal Terms

Treatment

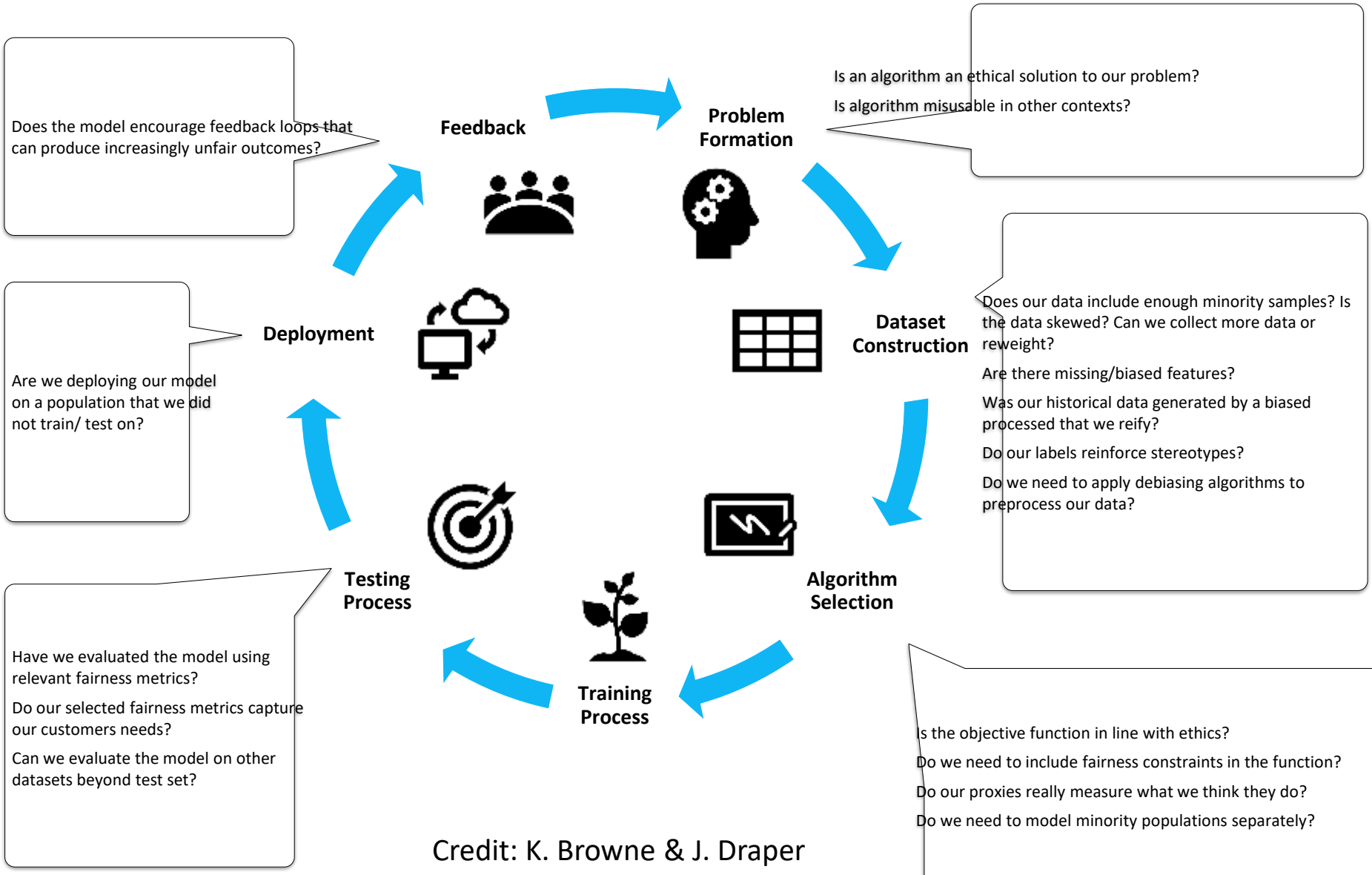
Disparate Treatment, Equality of Opportunity, Procedural Fairness

Outcome

Disparate Impact, Distributive justice, Minimized inequality of outcome

Slide courtesy of Sarah Bird

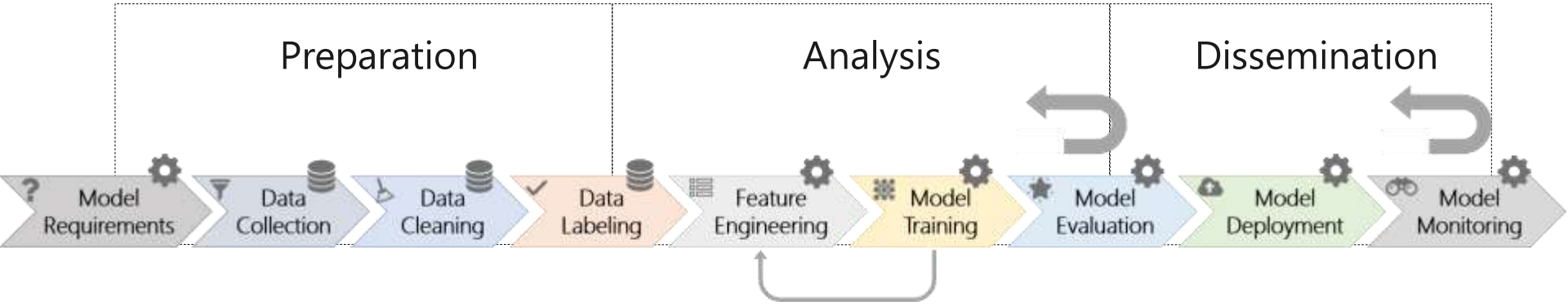
Engineering for equity during all phases of IR design



Credit: K. Browne & J. Draper

Slide courtesy of Sarah Bird

Measurement Fairness Tooling



Data Integrity	Label Quality	Model Integrity	Model Effect	Monitoring
• Representativeness • Quality • Intersectional properties	• Lablers • Coverage	• Representative Power of Features • Correlations • Interpretability	• Measure different fairness metrics • Calibration • Demographic Parody • False Positive Rate • False Negative Rate	• Track key metrics

Slide courtesy of Sarah Bird