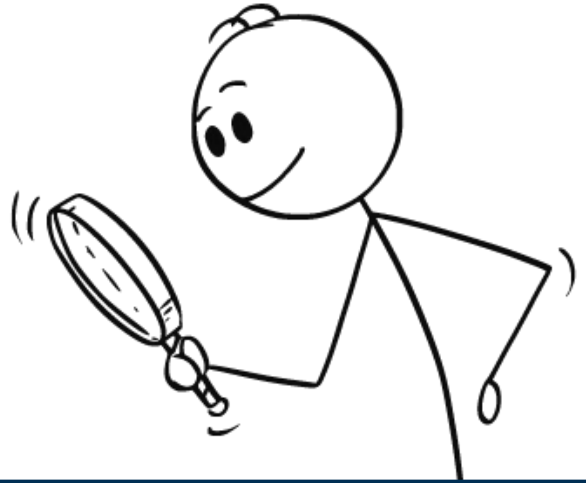


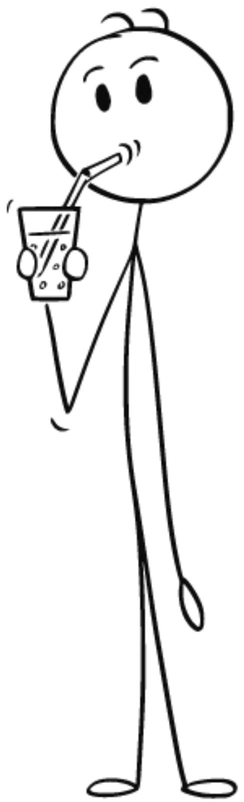
CSE 4/535
Information Retrieval

Sayantan Pal
PhD Student, Department of CSE
338Z Davis Hall

University at Buffalo
Department of CSE

Before we start

1. Project 1 released, due 27th September.
2. Join office hours if you have questions
3. Today's lecture - TF-IDF and VSM
4. Remind me 30 mins prior to class to solve your doubts related to project 1



Recap - Previous Class

1. Index Compression
    a. Dictionary Compression
    b. Postings Compression
        i. Gamma Codes
        ii. VB Codes

Term Weighting & Vector Space Models

Roadmap

- **Ranked retrieval** …. How is it different from Boolean retrieval?
- **Scoring documents** … Does it help?
- **Term frequency**
- **Collection statistics**
- **Weighting schemes**
- **Vector space scoring**

Ranked retrieval

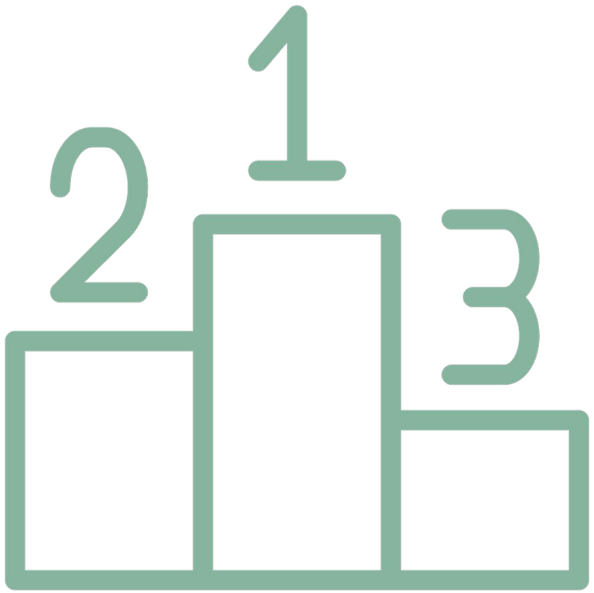
- Thus far, our queries have all been Boolean.
  - Documents either **match or don't**.
- Good for expert users with precise understanding of their needs and the collection.
  - Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users.
  - Most users incapable of writing Boolean queries (or they are, but they think it's too much work).
  - Most users don't want to wade through 1000s of results.
    - This is particularly true of web search.

Problem with Boolean search

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1: "standard user dlink 650" → 200,000 hits
- Query 2: "standard user dlink 650 no card found": 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits.
  - **AND gives too few; OR gives too many**

Ranked retrieval models

- Rather than a set of documents satisfying a query expression, in ranked retrieval, the system returns an ordering over the (top) documents in the collection for a query
- Free text queries: Rather than a query language of operators and expressions, the user's query is just one or more words in a human language
- In principle, there are two separate choices here, but in practice, ranked retrieval has normally been associated with free text queries and vice versa

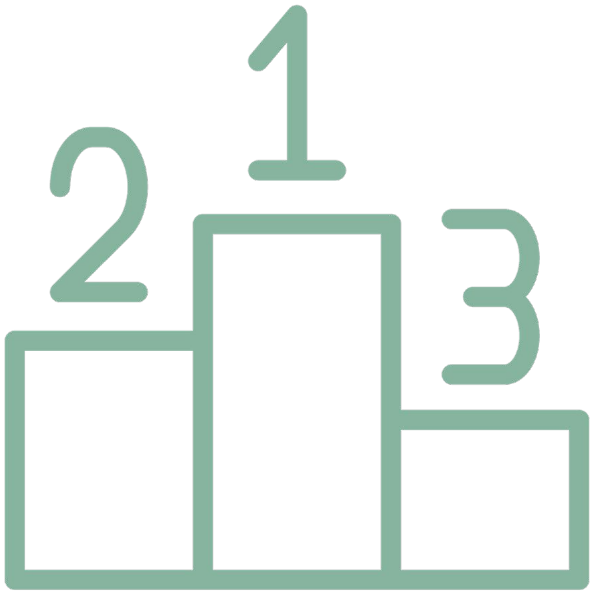


Feast or famine: not a problem in ranked retrieval

- When a system produces a ranked result set, large result sets are not an issue
  - Indeed, the size of the result set is not an issue
  - We just show the top k (≈ 10) results
  - We don’t overwhelm the user
- Premise: the ranking algorithm works

Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher
- How can we rank-order the documents in the collection with respect to a query?
- Assign a score – say in [0, 1] – to each document
- This score measures how well document and query "match".



Take 1: Jaccard coefficient

- A common measure of overlap of two sets A and B
  - jaccard(A,B) = $|A \cap B| / |A \cup B|$
  - jaccard(A,A) = 1
  - jaccard(A,B) = 0 if $A \cap B = 0$
- A and B don’t have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Scoring example

What is the query-document match score that the Jaccard coefficient computes for each of the two documents below?

Query: ides of march

Document 1: caesar died in march

Document 2: the long march

Issues with Jaccard for scoring

- It doesn’t consider term frequency (how many times a term occurs in a document)
- Rare terms in a collection are more informative than frequent terms. Jaccard doesn’t consider this information
- We need a more sophisticated way of normalizing for length

Query-document matching scores

- We need a way of assigning a score to a query/document pair
- Let’s start with a one-term query
- If the query term does not occur in the document: score should be 0
- The more frequent the query term in the document, the higher the score (should be)
- We will look at a number of alternatives for this.

Recall: Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

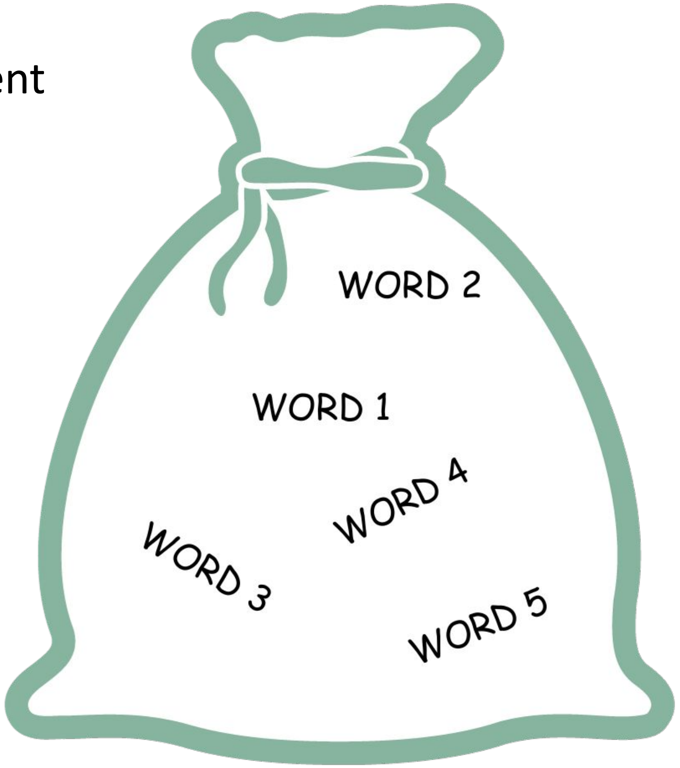
Term-document count matrices

- Consider the number of occurrences of a term in a document:

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words model

- Vector representation doesn't consider the ordering of words in a document
- John is quicker than Mary and Mary is quicker than John have the same vectors
- This is called the bag of words model.
- In a sense, this is a step back:
  a. The positional index was able to distinguish these two documents.



Term frequency - tf

1. The term frequency $tf_{t,d}$ of term t in document d is defined as the **number of times that t occurs in d**.
   a. Note: Frequency means count in IR
2. We want to use tf when computing query document match scores. But how?
3. Raw term frequency is not what we want:
   a. A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
   b. But not 10 times more relevant.
4. Relevance does not increase proportionally with term frequency.

Log-frequency weighting

- The log frequency weight of term t in d is

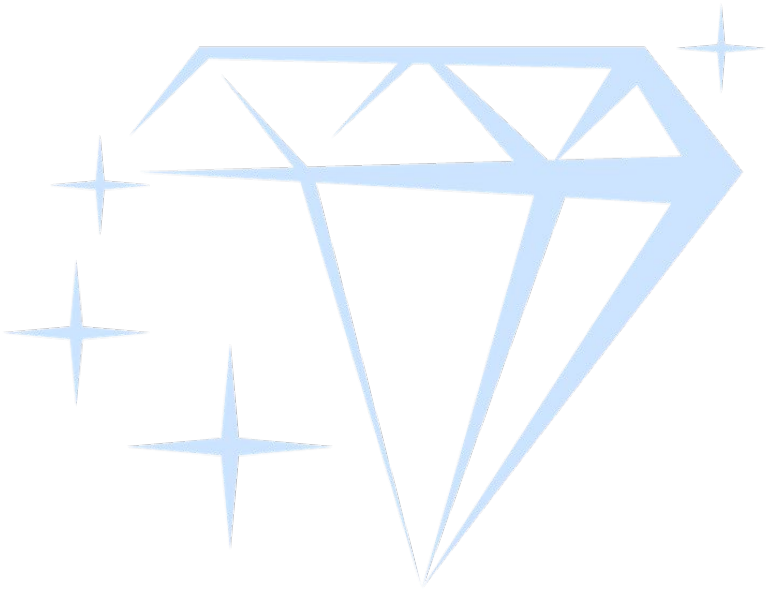
$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- Score for a document-query pair: sum over terms *t* in both *q* and *d*:
- score $= \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$

- The score is 0 if none of the query terms is present in the document.

Rare terms are more informative

1. **Rare terms are more informative than frequent terms**
    a. Recall stop words
2. Consider a term in the query that is rare in the collection (e.g., arachnocentric)
3. A document containing this term is very likely to be relevant to the query arachnocentric
4. → We want a high weight for rare terms like arachnocentric.



Collection vs. Document frequency

1. Collection frequency of t is the number of occurrences of t in the collection
2. Document frequency of t is the number of documents in which t occurs

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

Which word is better for search (gets higher weight)?

Collection vs. Document frequency

1. Collection frequency of t is the number of occurrences of t in the collection
2. Document frequency of t is the number of documents in which t occurs

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

Which word is better for search (gets higher weight)?

Answer: Insurance

Inverse Document Frequency (idf) weight

- df_t is the document frequency of t: the number of documents that contain t
  - df_t is an inverse measure of the informativeness of t
  - $\mathrm{df}_t \leq N$
- We define the idf (inverse document frequency) of t by

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

Example -> N = 1M Docs

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

There is one idf value for each term t in a collection.

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

Effect of idf on ranking

- Does idf have an effect on ranking for one-term queries, like
  - iPhone

Effect of idf on ranking

- Does idf have an effect on ranking for one-term queries, like
    - iPhone
- idf has no effect on ranking one term queries
- idf affects the ranking of documents for queries with at least two terms
- For the query capricious person, idf weighting makes occurrences of capricious count for much more in the final document ranking than occurrences of person.

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
  - Note: the "-" in tf-idf is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Score for a document given a query

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

- There are many variants
  - How “tf” is computed (with/without logs)
  - Whether the terms in the query are also weighted
  - …

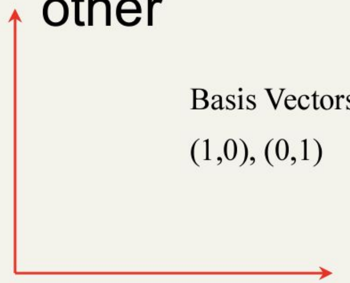
Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

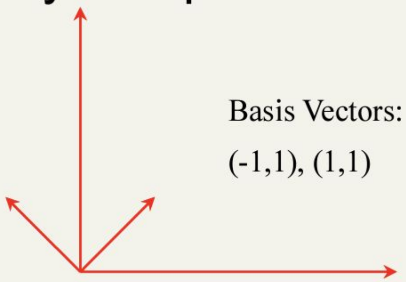
Vector Space Model

Vector Space

- A vector space is defined by a set of linearly independent Basis Vectors
  - typically, correspond to the dimension of a vector space
  - describe all vectors in the vector space
  - should be orthogonal or linearly independent to each other



Basis Vectors:
(1,0), (0,1)



Basis Vectors:
(-1,1), (1,1)

- What will be the basis vectors for information retrieval?

Vector Space Model

- Everything is represented as a vector in a high-dimensional space - documents, queries

$$D_j = (a_{j1}, a_{j2}, \ldots, a_{jn}) \qquad Q_j = (q_{j1}, q_{j2}, \ldots, q_{jn})$$

- Vocabulary : *n* distinct terms - (t_1, t_2, …, t_n)

Term	t_1	t_2	t_3	t_4	, …,	t_n
D_1	1	1	0	0	, …,	0
D_2	1	0	0	1	, …,	1
D_3	0	0	1	0	, …,	0
:	:	:	:	:	, …,	:

- How to compute similarity of document and query vectors? Linear Algebra

Documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
- These are very sparse vectors -most entries are zero.

Queries as vectors

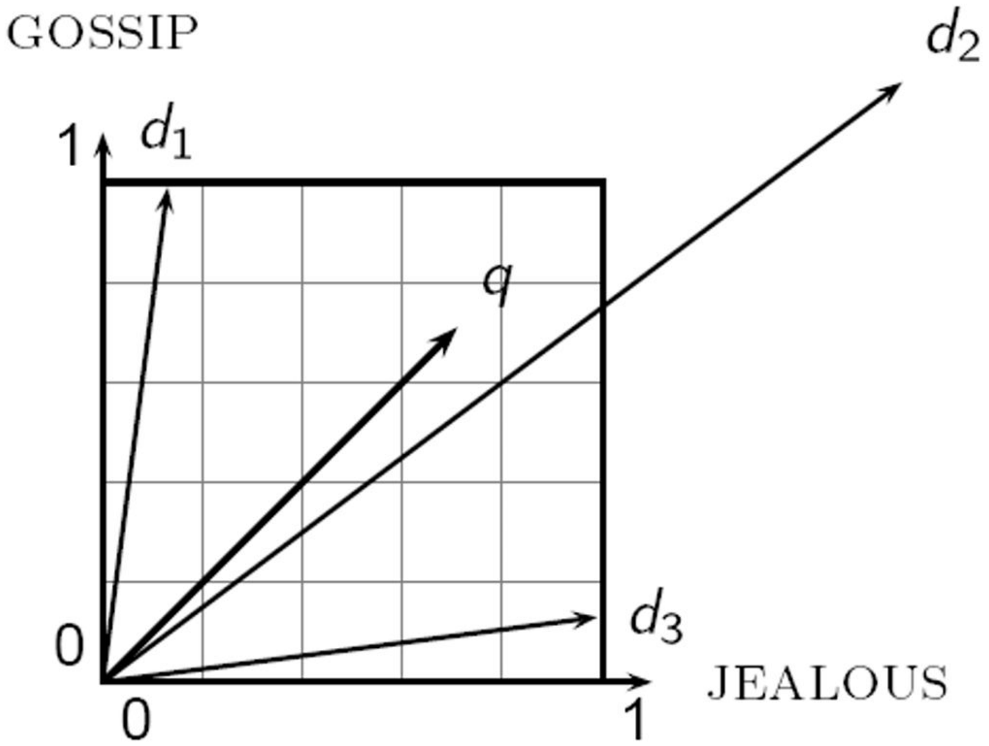
- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
- proximity = similarity of vectors
- proximity ≈ inverse of distance

Formalizing vector space proximity

- First cut: distance between two points is the distance between the endpoints of the two vectors
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea

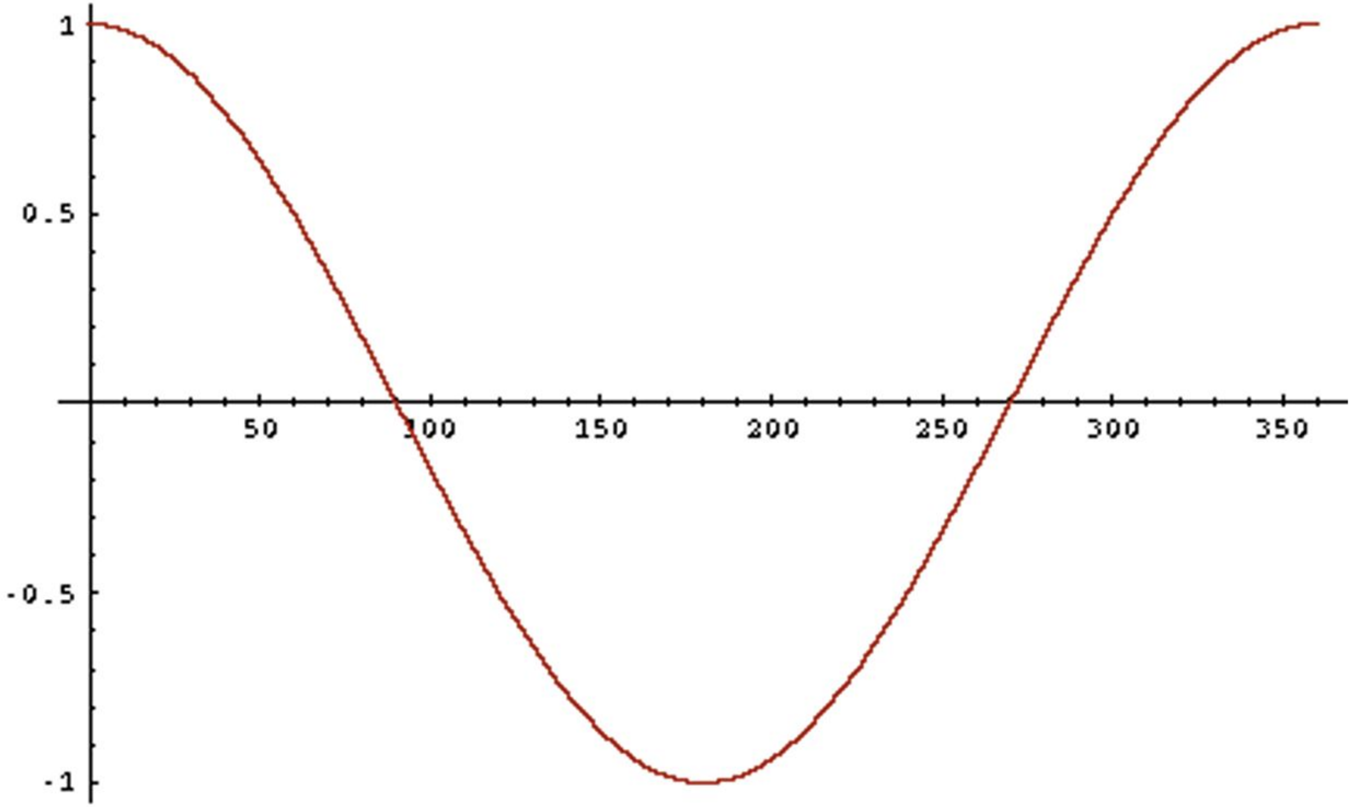
The Euclidean distance between $\vec{q}$ and $\vec{d_2}$ is large even though the distribution of terms in the query $\vec{q}$ and the distribution of terms in the document $\vec{d_2}$ are very similar.



Use angle instead of distance

- Thought experiment: take a document d and append it to itself. Call this document d'.
- “Semantically” d and d' have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.
- Key idea: Rank documents according to angle with query.

From angles to cosines



Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- Dividing a vector by its L_2 norm makes it a unit (length) vector (on surface of unit hypersphere)
- Effect on the two documents d and d′ (d appended to itself) from earlier slide: they have identical vectors after length-normalization.
  - Long and short documents now have comparable weights

cosine(query,document)

$$\cos(\vec{q},\vec{d}) = \frac{\overbrace{\vec{q}\bullet\vec{d}}^{\text{Dot product}}}{|\vec{q}||\vec{d}|} = \underbrace{\frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|}}_{\text{Unit vectors}} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

q_i is the weight of term i in the query
d_i is the weight of term i in the document

$\cos(\vec{q},\vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Cosine similarity amongst 3 documents

How similar are the novels
SaS: *Sense and Sensibility*
PaP: *Pride and Prejudice*, and
WH: *Wuthering Heights*?

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies (counts)

Note: To simplify this example, we don't do idf weighting.

3 documents example contd.

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

dot(SaS,PaP) ≈ 12.1
dot(SaS,WH) ≈ 13.4
dot(PaP,WH) ≈ 10.1

After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

cos(SaS,PaP) ≈ 0.94
cos(SaS,WH) ≈ 0.79
cos(PaP,WH) ≈ 0.69

Summary –vector space ranking

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top K(e.g., K= 10) to the user

Note - Cosine Similarity

- Cosine similarity measure emphasizes on relationship between document and query
- In large documents only parts of it will be relevant to a query - most of it will be non-relevant.
- In some sense, Cosine similarity seems to prefer ???? documents than ???? ones.

Note - Cosine Similarity

- Cosine similarity measure emphasizes on relationship between document and query
- In large documents only parts of it will be relevant to a query - most of it will be non-relevant.
- In some sense, Cosine similarity seems to prefer **shorter** documents than **longer** ones.

Parametric search

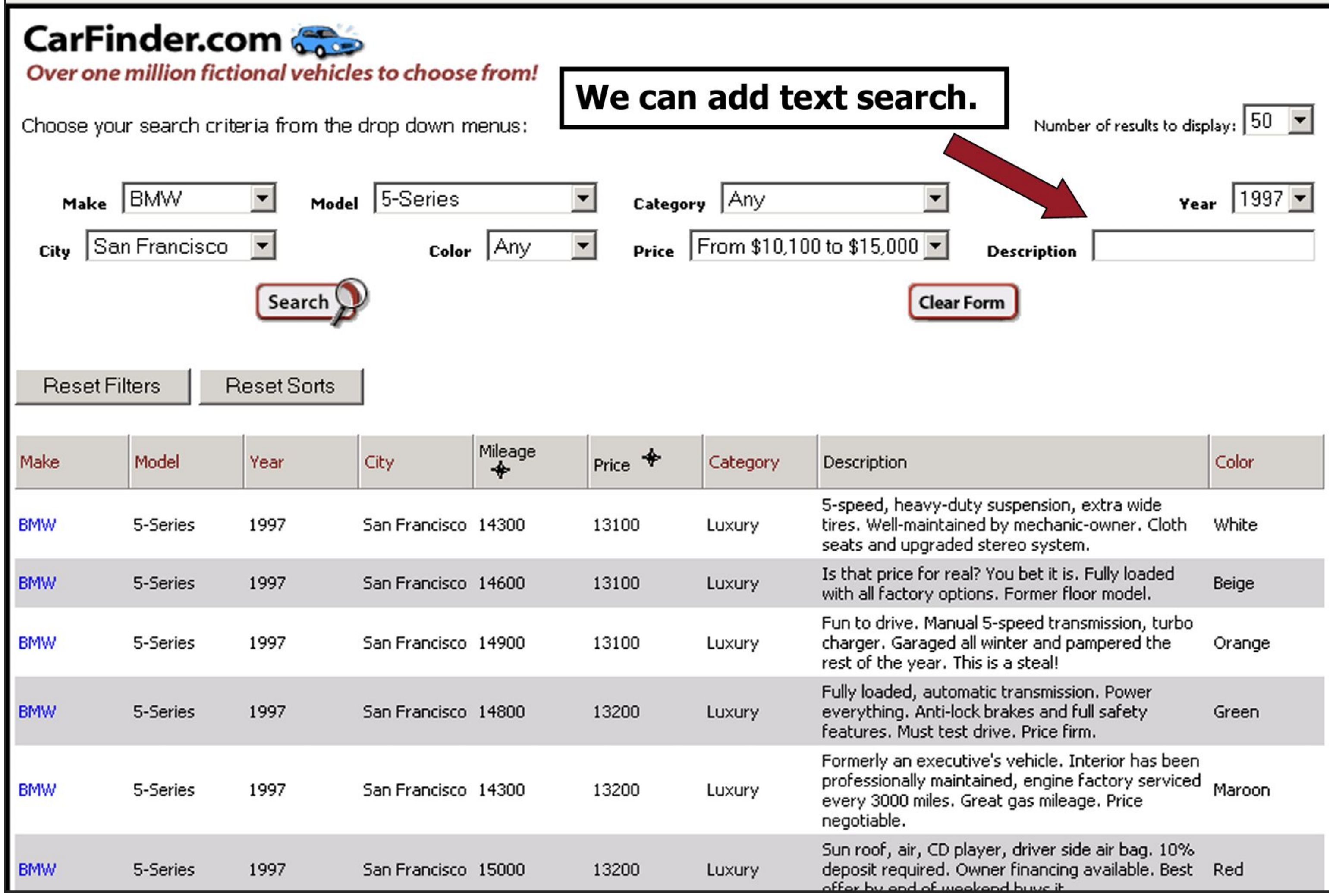
- Most documents have, in addition to text, some “meta-data” in fields e.g.,
  - Language = French
  - Format = pdf
  - Subject = Physics etc.
  - Date = Feb 2000

Field

Value

- A parametric search interface allows the user to combine a full-text query with selections on these field values e.g.,
  - language, date range, etc.

Parametric search example



CarFinder.com

Over one million fictional vehicles to choose from!

We can add text search.

Choose your search criteria from the drop down menus:

Number of results to display: 50

Make: BMW — Model: 5-Series — Category: Any — Year: 1997

City: San Francisco — Color: Any — Price: From $10,100 to $15,000 — Description:

Search — Clear Form

Reset Filters — Reset Sorts

Make	Model	Year	City	Mileage	Price	Category	Description	Color
BMW	5-Series	1997	San Francisco	14300	13100	Luxury	5-speed, heavy-duty suspension, extra wide tires. Well-maintained by mechanic-owner. Cloth seats and upgraded stereo system.	White
BMW	5-Series	1997	San Francisco	14600	13100	Luxury	Is that price for real? You bet it is. Fully loaded with all factory options. Former floor model.	Beige
BMW	5-Series	1997	San Francisco	14900	13100	Luxury	Fun to drive. Manual 5-speed transmission, turbo charger. Garaged all winter and pampered the rest of the year. This is a steal!	Orange
BMW	5-Series	1997	San Francisco	14800	13200	Luxury	Fully loaded, automatic transmission. Power everything. Anti-lock brakes and full safety features. Must test drive. Price firm.	Green
BMW	5-Series	1997	San Francisco	14300	13200	Luxury	Formerly an executive's vehicle. Interior has been professionally maintained, engine factory serviced every 3000 miles. Great gas mileage. Price negotiable.	Maroon
BMW	5-Series	1997	San Francisco	15000	13200	Luxury	Sun roof, air, CD player, driver side air bag. 10% deposit required. Owner financing available. Best	Red

Zones

- A zone is an identified region within a doc
- E.g., Title, Abstract, Bibliography
- Generally culled from marked-up input or document metadata (e.g., powerpoint)
- Contents of a zone are free text
- Not a “finite” vocabulary
- Indexes for each zone -allow queries like
- sorting in Title AND smith in Bibliography AND recur* in Body

Boosting

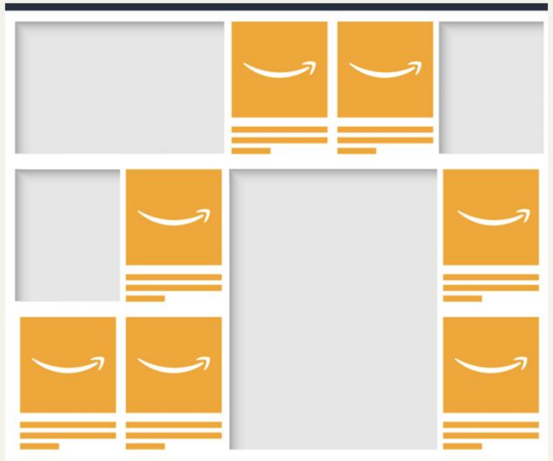
- Supported by Solr
- What to boost Query terms
  - E.g. terms appearing in title more important those those in body of document
  - Named entities
- Documents
  - E.g. more recent documents

Amazon Product Search (Sept 2019)

◆ WSJ NEWS EXCLUSIVE

Amazon Changed Search Algorithm in Ways That Boost Its Own Products

The e-commerce giant overcame internal dissent from engineers and lawyers, people familiar with the move say



Index support for zone combinations

- In the simplest version we have a separate inverted index for each zone
- Variant: have a single index with a separate dictionary entry for each term and zone
- E.g.,

bill.author 1 → 2

bill.title 3 → 5 → 8

bill.body 1 → 2 → 5 → 9

Of course, compress zone names like author/title/body.

Zone combinations index

- The above scheme is still wasteful: each term is potentially replicated for each zone
- In a slightly better scheme, we encode the zone in the postings:

bill | 1.author, 1.body | → | 2.author, 2.body | → | 3.title |

As before, the zone names get compressed.

References

1. Slides provided by Sougata Saha (Instructor, Fall 2022 - CSE 4/535)
2. Materials provided by Dr. Rohini K Srihari
3. https://nlp.stanford.edu/IR-book/information-retrieval-book.html