

# Information Retrieval

## Lecture 3: tfidf

[Reference] CS276: Information Retrieval and Web Search

## This lecture

- Parametric and field searches
  - Zones in documents
- Scoring documents: zone weighting
  - Index support for scoring
- Term weighting

## 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
- 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.

Fields

Values

## Parametric search example

CarFinder.com  
Over one million fictional vehicles to choose from!

Choose your search criteria from the drop down menus:

Make:  Model:  Category:  Year:   
City:  Color:  Price:

Number of results to display:

Notice that the output is a (large) table. Various parameters in the table (column headings) may be clicked on to effect a sort.

Make	Model	Year	City	Mileage	Price	Category	Description	Color
BMW	5-Series	1995	San Francisco	16100	11100	Luxury	Never driven in winter conditions. Body work makes it look like new. Keyless entry and security features. This is a bargain.	Silver
BMW	5-Series	1995	San Francisco	16600	11100	Luxury	Great first car for your teen-aged kid. Solid, dependable, affordable with 0% down and lower financing.	Blue
BMW	5-Series	1995	San Francisco	16800	11200	Luxury	Upgraded sound system really rocks. Customized interior features wood grain dash and beige leather seats. Power locks, windows, steering. Price firm.	White
BMW	5-Series	1995	San Francisco	16100	11300	Luxury	Safe choice for a young family: ABS, driver and passenger air bags. Roomy interior with power everything. Low mileage driving kids back and forth to soccer.	Maroon
BMW	5-Series	1995	San Francisco	16300	11400	Luxury	This baby's got it all: power steering, cruise, power locks, power windows, remote entry, leather interior, security alarm, ABS! Full-spectrum. Priced to sell.	Brown

## Parametric search example (con.)

**CarFinder.com**  
Over one million fictional vehicles to choose from!

Choose your search criteria from the drop down menus:

We can add text search.

Number of results to display: 50

Make:  Model:  Category:  Year:

City:  Color:  Price:  Description:

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 this price for real? You get 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. Gargled all winter and pandered the rest of the year. This is a steal!	Orange
BMW	5-Series	1997	San Francisco	14900	13200	Luxury	Fully loaded, automatic transmission. Power everything. Air 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	Sunroof, air, CD player, driver side air bag, 10% deposit required. Owner financing available. Best deal	Red

## Parametric / field search

- In these examples, we select field values
  - Values can be hierarchical, e.g.,
  - Geography**: Continent → Country → State → City
- A paradigm for navigating through the document collection, e.g.,
  - "Aerospace companies in Brazil" can be arrived at first by selecting **Geography** then **Line of Business**, or vice versa
  - Filter docs in contention and run text searches scoped to subset

## Index support for parametric search

- Must be able to support queries of the form
  - Find pdf documents that contain "stanford university"
  - A field selection (on doc format) and a phrase query
- Field selection
  - use inverted index of field values → docids
    - Organized by field name
    - Use compression etc. as before

## Parametric index support

- Optional
  - provide richer search on field values – e.g., wildcards
    - Find books whose Author field contains *s\*trup*
- Range search
  - find docs authored between September and December
    - Inverted index doesn't work (as well)
    - Use techniques from database range search
    - See for instance [www.bluerwhite.org/btree/](http://www.bluerwhite.org/btree/) for a summary of B-trees
- Use query optimization heuristics as before

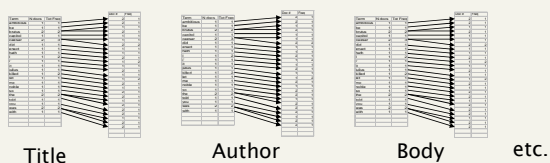
## Field retrieval

- In some cases, must retrieve field values
  - E.g., ISBN numbers of books by **s\*trup**
- Maintain “forward” index
  - For each doc, those field values that are “retrievable”
  - Indexing control **file specifies**
    - which fields are retrievable (and can be updated)
  - Storing primary data here, not just an index

## 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**
- Not queries like “all papers whose authors cite themselves”

## Zone indexes – simple view



## So we have a database now?

- **Not really**
- Databases do lots of things we don't need
  - Transactions
  - Recovery
    - our index is not the system of record; if it breaks, simply reconstruct from the original source
  - Indeed, **only indexes**
    - **we never have to store text in a search engine**
- **Focusing on optimized indexes** for text-oriented queries, **not** an **SQL engine**

## Scoring and Ranking

### Scoring

- Queries have all been Boolean: **Docs either match or not**
- **Good:**
  - Expert users with precise understanding of their needs and the corpus
  - Applications can consume 1000's of results
- **Not good:**
  - users with poor Boolean formulation of their needs
- **Most** users don't want to wade through 1000's of results
  - use of **web search engines**

- *Wish: to return in order the documents **most likely to be useful** to the searcher*
- **How can rank order** the docs
  - in the corpus with respect to a query?
- Assign a **score**
  - say in [0,1]
  - for each doc on each query
  - under web search

### Linear zone combinations (线性组合)

- First generation of scoring methods
  - a **linear combination of Booleans**
  - E.g.,

$$\text{Score} = 0.6 * \langle \text{sorting in Title} \rangle + 0.3 * \langle \text{sorting in Abstract} \rangle + 0.05 * \langle \text{sorting in Body} \rangle + 0.05 * \langle \text{sorting in Boldface} \rangle$$

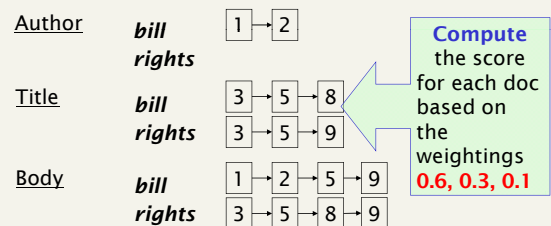
- Each expression takes on a value in {0,1}.
  - such as **<sorting in Title>**
- Then the overall score is in [0,1].

For this example the scores can only take on a **finite set of values** – what are they?

- the expressions between **< .... >**
  - could be **any Boolean query**
- Who generates the Score expression?
  - with weights **such as 0.6** etc.?
  - In uncommon cases: the **user**, in the **UI**
  - Most commonly: a **query parser**
    - that takes the user's Boolean query and runs it on the indexes for each zone

## Exercise

- On the query: **bill OR rights**
  - suppose : retrieve the following docs from the various zone indexes (索引区).

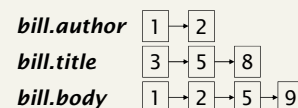


## General idea

- Given a **weight vector**
  - whose components **sum up to 1**.
  - There is a weight for **each zone/field**.
- Given a Boolean query
  - assign a score to each doc by **adding up the weighted contributions** of the zones/fields.
- Typically
  - users want to see the **K highest-scoring** docs.

## Index support for zone combinations

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



compress zone names like author/title/body.

## Zone combinations index

- The scheme is still wasteful:
  - each term is potentially replicated for each zone
- Slightly better scheme
  - encode the zone in the postings

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

As before, the zone names get compressed.

- At query time
  - accumulate (积累) contributions to the total score of a document from the various postings.

## Score accumulation

1	0.7
2	0.7
3	0.4
5	0.4

*bill* 1.author, 1.body → 2.author, 2.body → 3.title  
*rights* 3.title, 3.body → 5.title, 5.body →

- Seeks the postings for the query *bill* OR *rights*
- Accumulate scores for each doc
  - in a linear merge as before
- Note:
  - get both *bill* and *rights* in the Title field of doc 3
  - but score it no higher.
- Should give more weight to more hits?

```
ZoneScore(q1,q2)
    float scores[N]={0}
    constant gzone[I]={Wzone1, Wzone2,...,WzoneI}
    p1= postings(q1)
    p2= postings(q2)
    while p1!=NULL and p2!=NULL
        do if docId(p1) == docId(p2)
            then scores[docId(p1)] = WeightedZone(p1,p2,g)
                p1=next(p1)
                p2=next(p2)
            else if docId(p1) < docId(p2)
                then p1=next(p1)
            else p2=next(p2)
    return scores
```

Accumulate scores

## Where do these weights come from?

- Machine learned relevance
- Given
  - A test corpus
  - A suite of test queries
  - A set of relevance judgments
- Learn a set of weights
  - such that relevance judgments matched
- Be formulated as ordinal regression (有序回归)

## Full text queries

- Just scored the Boolean query:  
**bill OR rights**
- Most users more likely to type  
**bill rights** or **bill of rights**
  - How do interpret(理解) these *full text* queries?
  - No** Boolean connectives (布尔连接词)
  - Of several query terms some may be **missing** in a doc
  - Only some query terms may **occur in the title**, etc.

- To use zone combinations for free text queries
- Need
  - A way of **assigning a score**
    - to a pair **<free text query, zone>**
  - Zero query terms in the zone
    - should **mean a zero score**
  - More query terms in the zone
    - should **mean a higher score**
  - Scores **don't** have to be Boolean
- Will look at some alternatives

## Incidence matrices (关联矩阵)

/ Correlation matrices

- Recall:**
  - Document (or a zone in it) is **binary vector**  $X$  in  $\{0,1\}^v$
  - Query is a vector
- Score:** Overlap (重叠) measure

$$|X \cap Y|$$

	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

## Example

- On the query: **ides of march** (三月十五)
  - Shakespeare's(莎士比亚的) *Julius Caesar*(尤利乌斯 凯撒) has a score of **3**
- All other Shakespeare plays have a score of **2**
  - because they contain **march** or 1
- Thus: in a rank order (排名序列)
  - Julius Caesar* would come **out tops**

## Overlap matching

- What's wrong with the overlap measure?
- It doesn't consider:
  - **Term frequency** in document
  - **Term scarcity** in collection (document mention frequency)
    - *of* is more common than *ides* or *march*
  - **Length** of documents
    - (And queries: score not normalized)

## Overlap matching

- One can normalize in various ways:

- Jaccard(杰卡德) coefficient(系数)

$$|X \cap Y| / |X \cup Y|$$

- Cosine(余弦) measure

$$|X \cap Y| / \sqrt{|X| \times |Y|}$$

机器学习中的相似性度量方法:

1. 欧氏距离
2. 曼哈顿距离
3. 切比雪夫距离
4. 闵可夫斯基距离
5. 标准化欧氏距离
6. 马氏距离
7. 夹角余弦
8. 汉明距离
9. 杰卡德距离&杰卡德相似系数
10. 相关系数&相关距离
11. 信息熵

## Scoring: density-based(基于密度)

- Thus far
  - **position** and **overlap** of **terms** in a doc
    - title, author etc.
- Obvious next idea
  - if a document talks about a **topic** *more*,
  - then it is a **better match**
- This applies even when we only have a **single query term**.
- Document **relevant** if it has a **lot of** the **terms**
- This leads to the idea of **term weighting**.