Scoring

Fields and Zones

- Is a document simply a sequence of words?
  - Many structural components, e.g., authors, title, date of publication, …
- Meta data: the data about documents
- Fields: document features where the possible values are finite
  - Examples: dates, ISBN
- Zones: document features whose content can be arbitrary free text
  - Examples: title, abstracts

Parametric Search

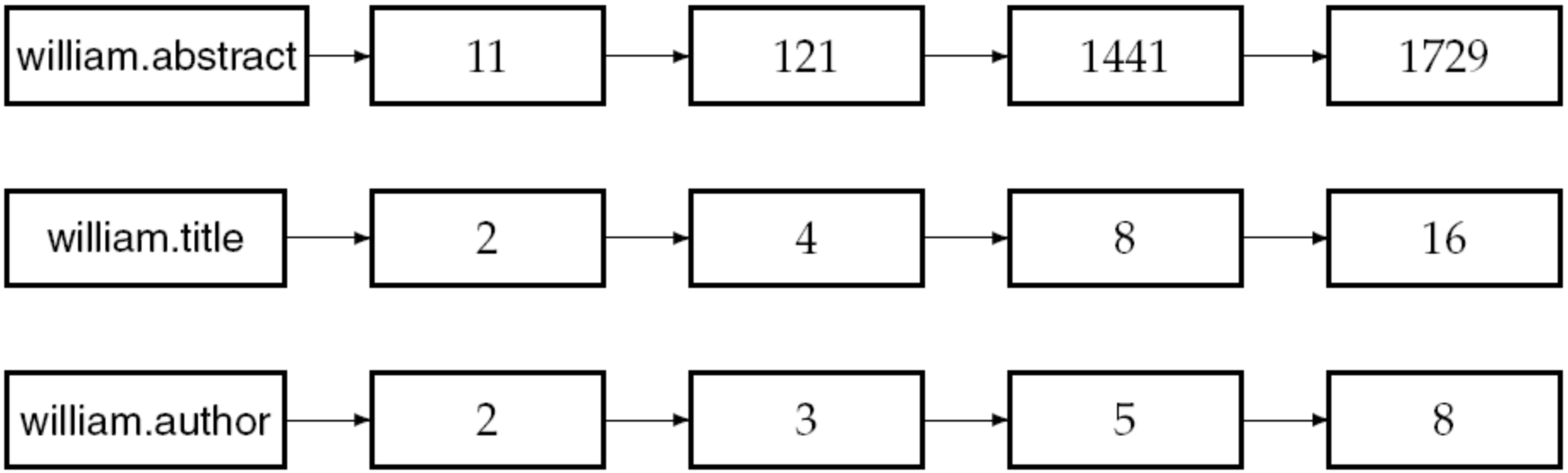
- A user may specify requirements on fields and zones

Bibliographic Search

Search category	Value
Author	*Example:* Widom, J *or* Garcia-Molina
Title	Also a part of the title possible
Date of publication	*Example*: `1997` *or* `<1997` *or* `>1997` limits the search to the documents appeared in, before and after 1997 respectively
Language	Language the document was written in: English
Project	ANY
Type	ANY
Subject group	ANY
Sorted by	Date of publication
	Start bibliographic search

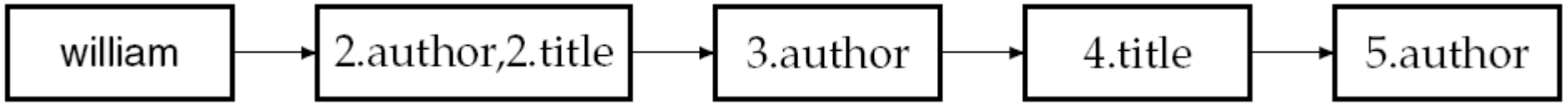
Parametric Indexes

- One parametric index for each field/zone



Zone Index

- Encoding fields and zones where a term occurs
- Advantages
  - Reducing the size of dictionary
  - Efficient query answering

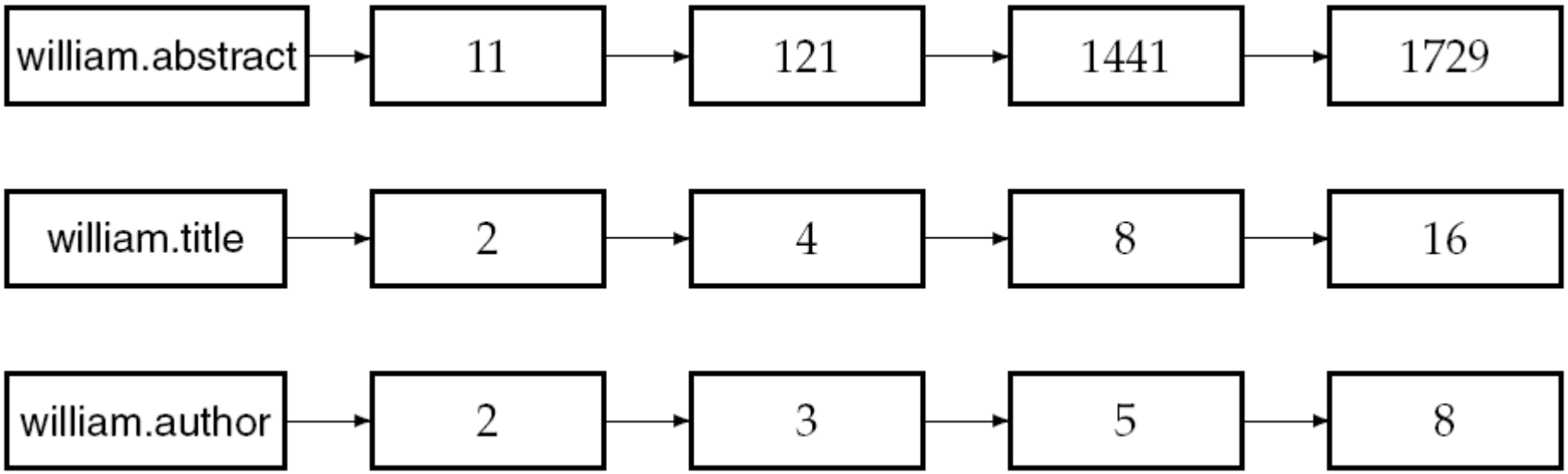


Weighted Zone Scoring

- Different fields/zones may have different importance in evaluating how a document matches a query
- For a query q and a document d, weighted zone scoring assigns to pair (q, d) a score in range [0, 1] by computing a linear combination of zone scores
- Suppose each document has l zones, let $g_1, \ldots, g_l \in [0, 1]$ such that $\Sigma_{i=1}^{l} g_i = 1$
  - Each field/zone of the document contributes a Boolean value – let s_i be the Boolean score denoting a match or absence between q and the i-th zone
  - The weighted zone score is $\Sigma_{i=1}^{l} g_i \times s_i$

Example

- Let $g_{abstract} = 0.5$, $g_{title} = 0.3$, and $g_{author} = 0.2$
- Score(D2) = 0.3 + 0.2 = 0.5
- Score(D11) = 0.5



Learning Weights

- Using training examples that have been judged editorially
- Each training example is a tuple consisting of a query q and a document d, and a relevance judgment for d on q
  - The judgment can be binary – relevant or not
  - A judgment score can also be used
- Compute the weights such that the learned scores approximate the relevance judgments as much as possible
  - An optimization problem

A Simple Case

- Only two zones: title and body

$$score(d,q) = g \cdot s_{Title}(d,q) + (1-g)s_{Body}(d,q)$$

- A training example is a tuple $\Phi_j = (d_j, q_j, r(d_j, q_j))$
  - $r(d_j, q_j)$ is either relevant or irrelevant

Example	DocID	Query	s_T	s_B	Judgment
Φ_1	37	linux	1	1	Relevant
Φ_2	37	penguin	0	1	Non-relevant
Φ_3	238	system	0	1	Relevant
Φ_4	238	penguin	0	0	Non-relevant
Φ_5	1741	kernel	1	1	Relevant
Φ_6	2094	driver	0	1	Relevant
Φ_7	3191	driver	1	0	Non-relevant

Learning

- Using examples

$$score(d_j, q_j) = g \cdot s_{Title}(d_j, q_j) + (1-g) s_{Body}(d_j, q_j)$$

- Error of scoring function

$$\varepsilon(g, \Phi_j) = (r(d_j, q_j) - score(d_j, q_j))^2$$

- Optimization goal: minimizing the total error

$$\sum_j \varepsilon(g, \Phi_j)$$

Optimal Weights

s_T	s_B	Score
0	0	0
0	1	$1-g$
1	0	g
1	1	1

- Let n_{01r} (n_{01i}) be the numbers of training examples that $S_{Title} = 0$ and $S_{Body} = 1$ and the judgment is relevant (irrelevant). The contribution of those examples that $S_{Title} = 0$ and $S_{Body} = 1$ to the total error is

$$[1-(1-g)]^2 n_{01r} + [0-(1-g)]^2 n_{01i}$$

- The total error is $(n_{01r} + n_{10i})g^2 + (n_{10r} + n_{01i})(1-g)^2 + n_{00r} + n_{11i}$
- By differentiating with respect to g and seting the result to 0, the optimal value of g is $\frac{n_{10r} + n_{01i}}{n_{01r} + n_{10i} + n_{10r} + n_{01i}}$

Term Frequency and Weighting

- A document or zone that mentions a query term more often is likely more relevant to the query → a higher score
- Term frequency TF(t, d): the number of occurrences of term t in document d
- Are all terms in a document equally important?
  - Word "the" is frequent in many documents → even though "the" is frequent in d, "the" may not capture the topic of d
  - We also need to consider how popular a term is in the whole document collection

Document Frequency

- Document frequency DF(t): the number of documents in the collection that contain a term t
- Collection frequency CF(t): the total number of occurrences of a term t in the collection
- Document frequency is more meaningful
  - We want the few documents that contain "insurance" to get a higher boost for a query on "insurance" than the many documents containing "try" to get from a query on "try"

Word	cf	df
try	10422	8760
insurance	10440	3997

Inverse Document Frequency

- $IDF(t) = \log \frac{N}{DF(t)}$, N: the total number of documents in the collection
- IDF(t) is high if t is a rare term
- IDF(t) is likely low if t is a frequent term

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

TF/IDF Weighting

- TF(t, d) measures the importance of a term t in document d
- IDF(t) measures the importance of a term t in the whole collection of documents
- TF/IDF weighting: putting TF and IDF together TFIDF(t, d) = TF(t, d) x IDF(t)
  - High if t occurs many times in a small number of documents, i.e., highly discriminative in those documents
  - Not high if t appears infrequent in a document, or is frequent in many documents, i.e., not discriminative
  - Low if t occurs in almost all documents, i.e., no discrimination at all
- A query may contain multiple query words
  - Score(q, d) = $\Sigma_{t \in q}$ TFIDF(t, d)

Document Vectors

- Each term can be treated as a dimension
- A document is a vector $\vec{V}(d)$ with value TF(t_i) on term t_i
- Can the similarity between two documents be measured as the magnitude of the vector difference?
  - If one document is long and the other is short, their absolute term frequencies may differ substantially
  - However, the relative distributions of terms may be similar

Cosine Similarity

- Measure the similarity using the cosine of the vector representations

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| \times |\vec{V}(d_2)|}$$

- Dot product $\vec{V}(d_1) \cdot \vec{V}(d_2) = \sum_t TF(t, d_1) \times TF(t, d_2)$
- Euclidean length $|\vec{V}(d_1)| = \sqrt{\sum_t TF(t, d_1)^2}$

- Length-normalization $\vec{v}(d_1) = \frac{\vec{V}(d_1)}{|\vec{V}(d_1)|}$

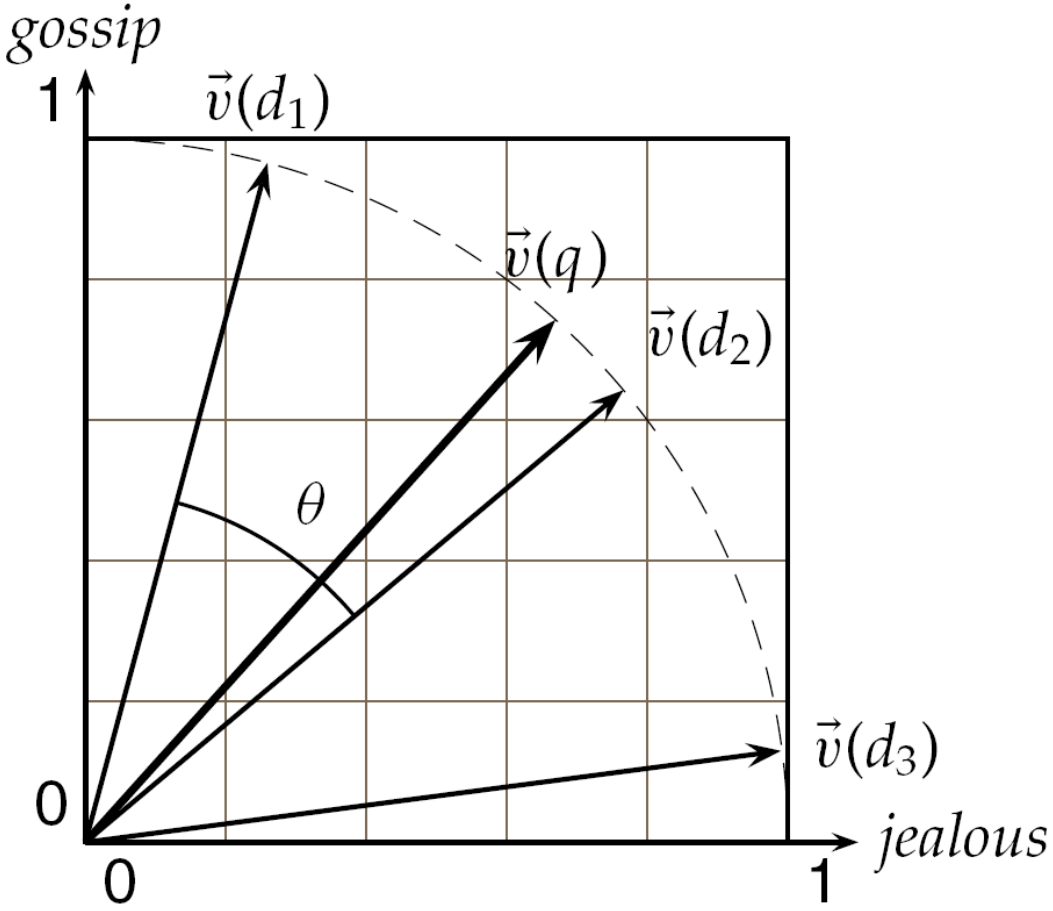
$$sim(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2)$$

Example

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

	Doc1	Doc2	Doc3
car	0.88	0.09	0.58
auto	0.10	0.71	0
insurance	0	0.71	0.70
best	0.46	0	0.41

Finding Similar Documents



Term-Document Matrix

- Each row represents a term (dimension), in total M terms
- Each column represents a document, in total N documents

D_1 Tropical Freshwater Aquarium Fish.
D_2 Tropical Fish, Aquarium Care, Tank Setup.
D_3 Keeping Tropical Fish and Goldfish in Aquariums, and Fish Bowls.
D_4 The Tropical Tank Homepage - Tropical Fish and Aquariums.

Terms	Documents			
	D_1	D_2	D_3	D_4
aquarium	1	1	1	1
bowl	0	0	1	0
care	0	1	0	0
fish	1	1	2	1
freshwater	1	0	0	0
goldfish	0	0	1	0
homepage	0	0	0	1
keep	0	0	1	0
setup	0	1	0	0
tank	0	1	0	1
tropical	1	1	1	2

Query Vector

- A query can be viewed as a very short document
  - Query vector: the vector of the query document
- Query answering: find the documents which are most similar to the query vector

$$score(q,d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)| \times |\vec{V}(d)|}$$

  - Can be very expensive in practice – many words in a document and many documents

Sublinear TF Scaling

- If a term appears 20 times, the significance may not be 20 times of a single occurrence
  - The marginal increase of significance decreases
- Using the logarithm of the term frequency

$$WF(t,d) = \begin{cases} 1 + \log TF(t,d) & \text{if } TF(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$WFIDF(t,d) = WF(t,d) \times IDF(t)$$

Maximum TF Normalization

- A document repeating a term may gain TF
  - Anomaly: if a document copies itself, the TF score doubles but the relevance to any query should not!
- Solution: normalize the TF weights of all terms occurring in a document by the maximum TF in that document
- For document d, let $TF_{\max}(d) = \max_{\tau \in d} TF(\tau, d)$
- Normalized term frequency $NTF(t,d) = a + (1-a)\frac{TF(t,d)}{TF_{\max}(d)}$
  - Smoothing term a = 0.4

Maximum TF Normalization: Cons

- Hard to tune: a change in the stop word list can dramatically alter term weightings and thus rankings
- A document may contain an outlier term with an unusually large TF but not representative of the content of the document
- Distribution of TF in a document should be considered: a document where the most frequent term appears roughly as often as many other terms should be treated differently from one with a more skewed distribution

Document/Query Weighting Schemes

- Notion: ddd.qqq – ddd for documents, qqq for queries
  - Example: lnc.ltc

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)	$\frac{1}{\sqrt{w_1^2+w_2^2+\cdots+w_M^2}}$
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)	$1/u$ (Section 17.4.4)
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)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Summary

- Parametric search and parametric indexes
- TF/IDF weighting
- Document vectors
- Document and query weight schemes

To-Do List

- Read Section 7.1