COMP4650/6490 Document Analysis

Introduction to Ranked Retrieval

ANU School of Computing

Administrative Matters

- Voting for COMP6490 Class Representatives
  - If you are enrolled in COMP6490
  - Cast your vote on Wattle (in Week 2)
  - Please vote by 2pm Friday 4 August
- Assignment 1
  - Released: Monday 31 July
  - Due: 5pm Wednesday 16 August, AEST (UTC +10)
  - Start working on it

So far…

We looked at:

- Boolean retrieval
- Inverted index data structure
- Tokenisation and other preprocessing steps

Australian National University

Outline

- From Boolean retrieval to ranked retrieval
  - Bag-of-Words (BoW) model
  - Field or zone indexes
  - Boolean retrieval with field
  - Limitations of Boolean retrieval
- Ranked retrieval
  - What is ranked retrieval
  - Weighted field scoring
  - Term frequency & inverse document frequency
  - TF-IDF variants
  - Vector space model

Australian National University

Outline

- From Boolean retrieval to ranked retrieval
  - Bag-of-Words (BoW) model
  - Field or zone indexes
  - Boolean retrieval with field
  - Limitations of Boolean retrieval
- Ranked retrieval
  - What is ranked retrieval
  - Weighted field scoring
  - Term frequency & inverse document frequency
  - TF-IDF variants
  - Vector space model

Bag-of-Words (BoW) Model

- You may notice that we did not care about the ordering of tokens in document
- Bag-of-Words (BoW) assumption: A document is a collection (multiset) of words
  - Doc1: The cat ate the fish
  - Doc2: The fish ate the cat
  - These two documents are the same under BoW assumption
- We will use the BoW assumption throughout the IR part of the course
- Real world systems (e.g. Google) do have to consider ordering
- The NLP module will cover approaches that consider ordering

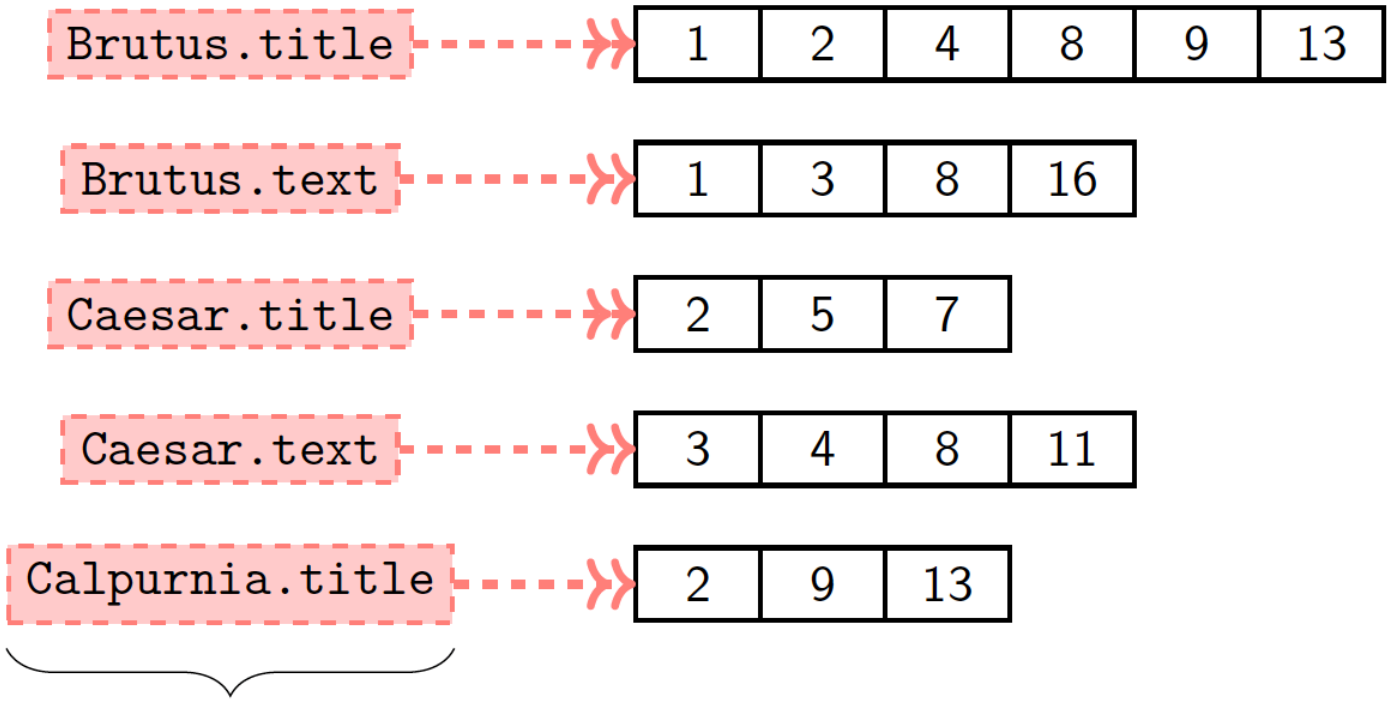
Field or Zone in Document

- Documents may be semi-structured:
  - Title
  - Author(s)
  - Date of publication
  - Abstract
  - Body
  - …
- Users may want to limit search scope to a certain field or zone
- A partial solution to the weakness of BoW

Boolean Retrieval

Basic Field Index

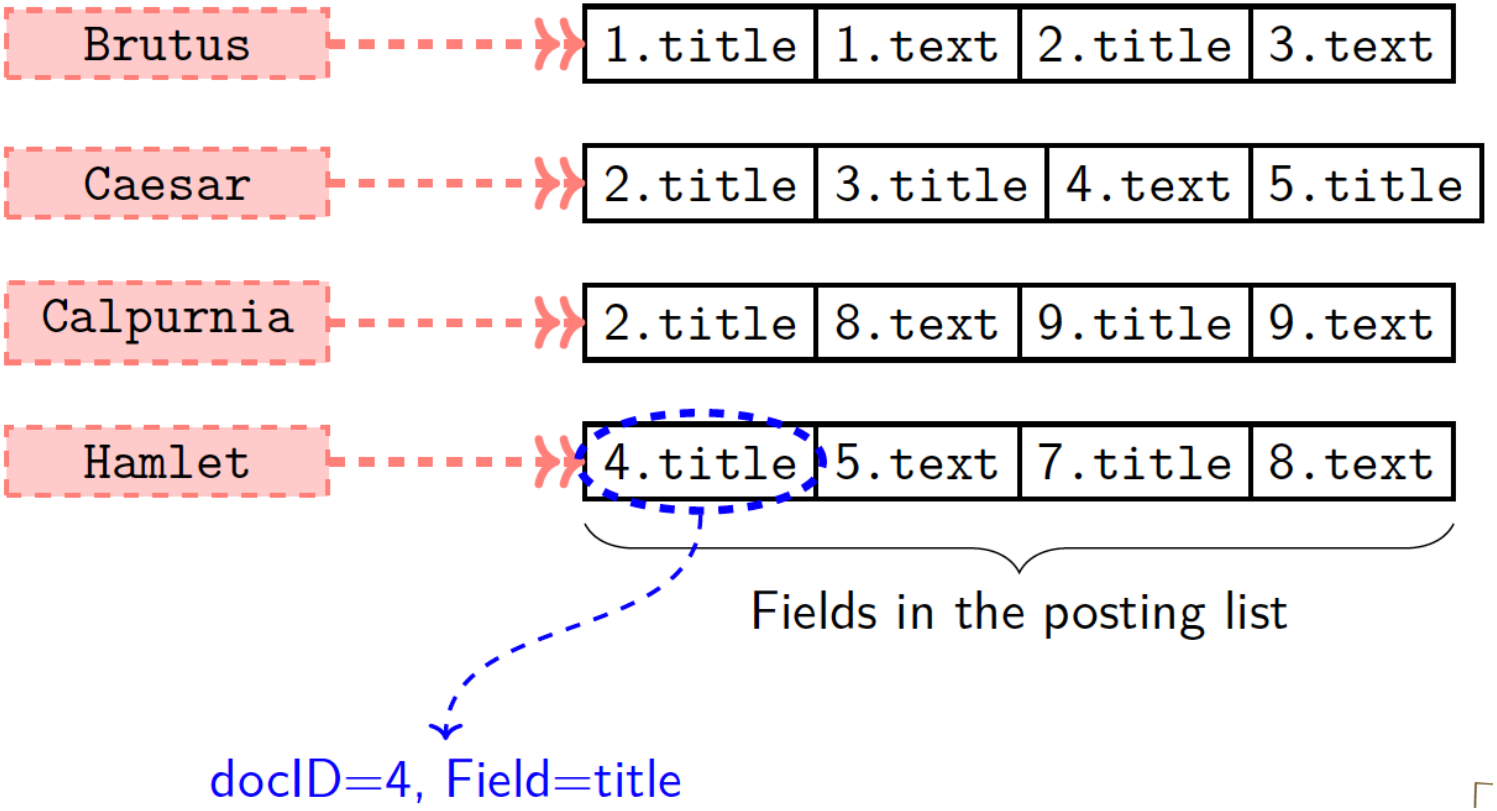
Equivalent to a separate index for each field



Boolean Retrieval

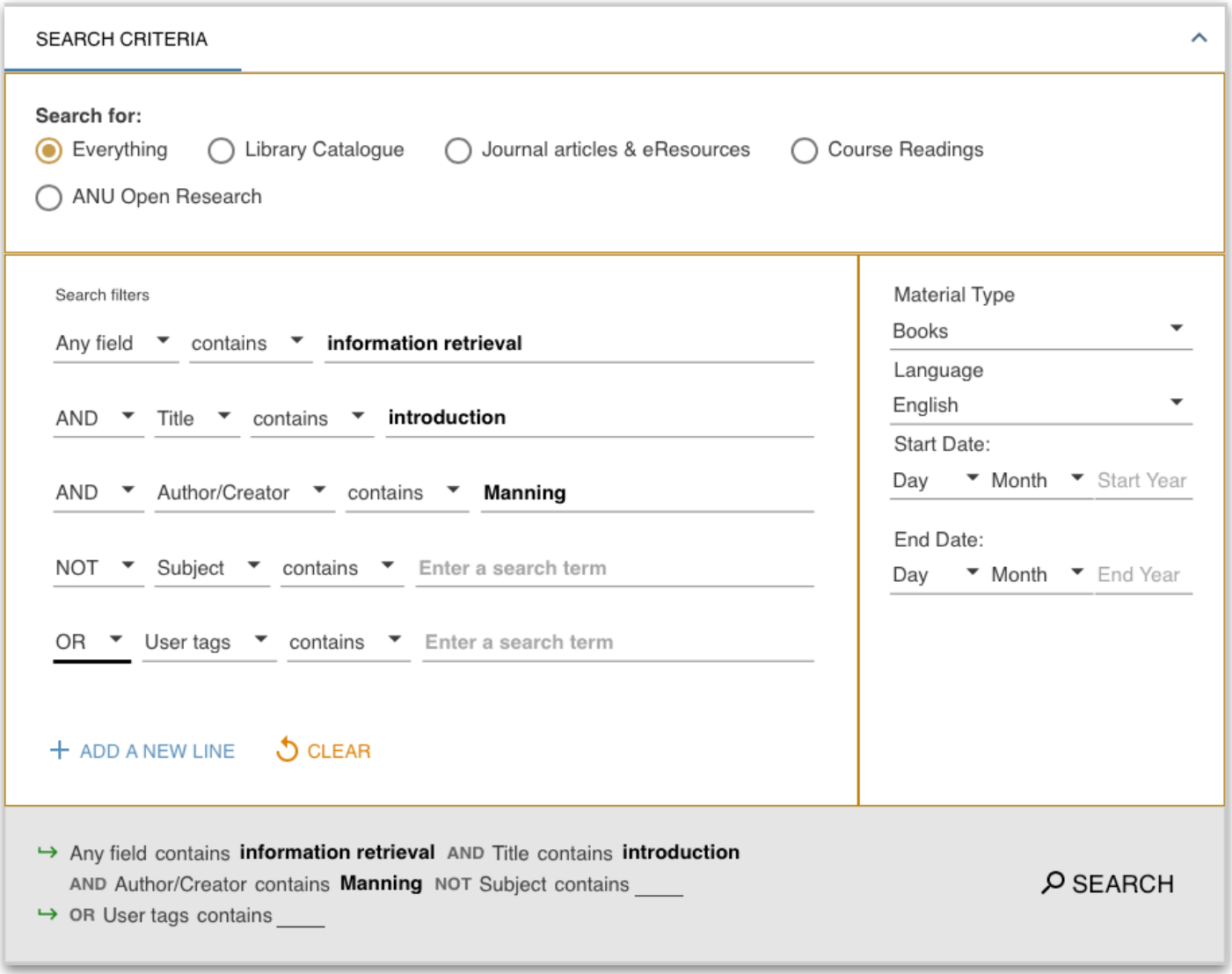
Field in Posting

Reduces size of dictionary & enables efficient weighted field scoring



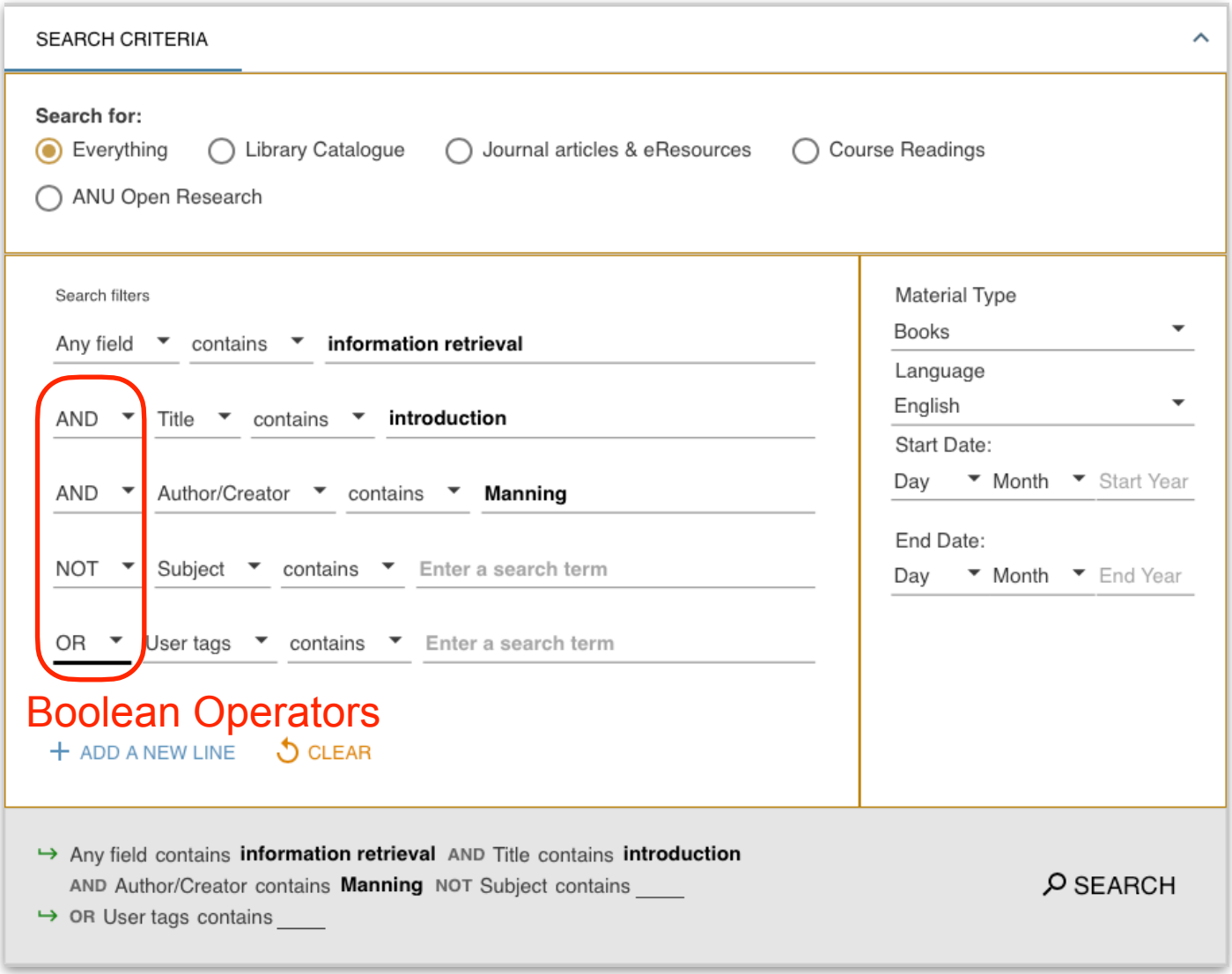
Boolean Retrieval

Boolean Retrieval with Field



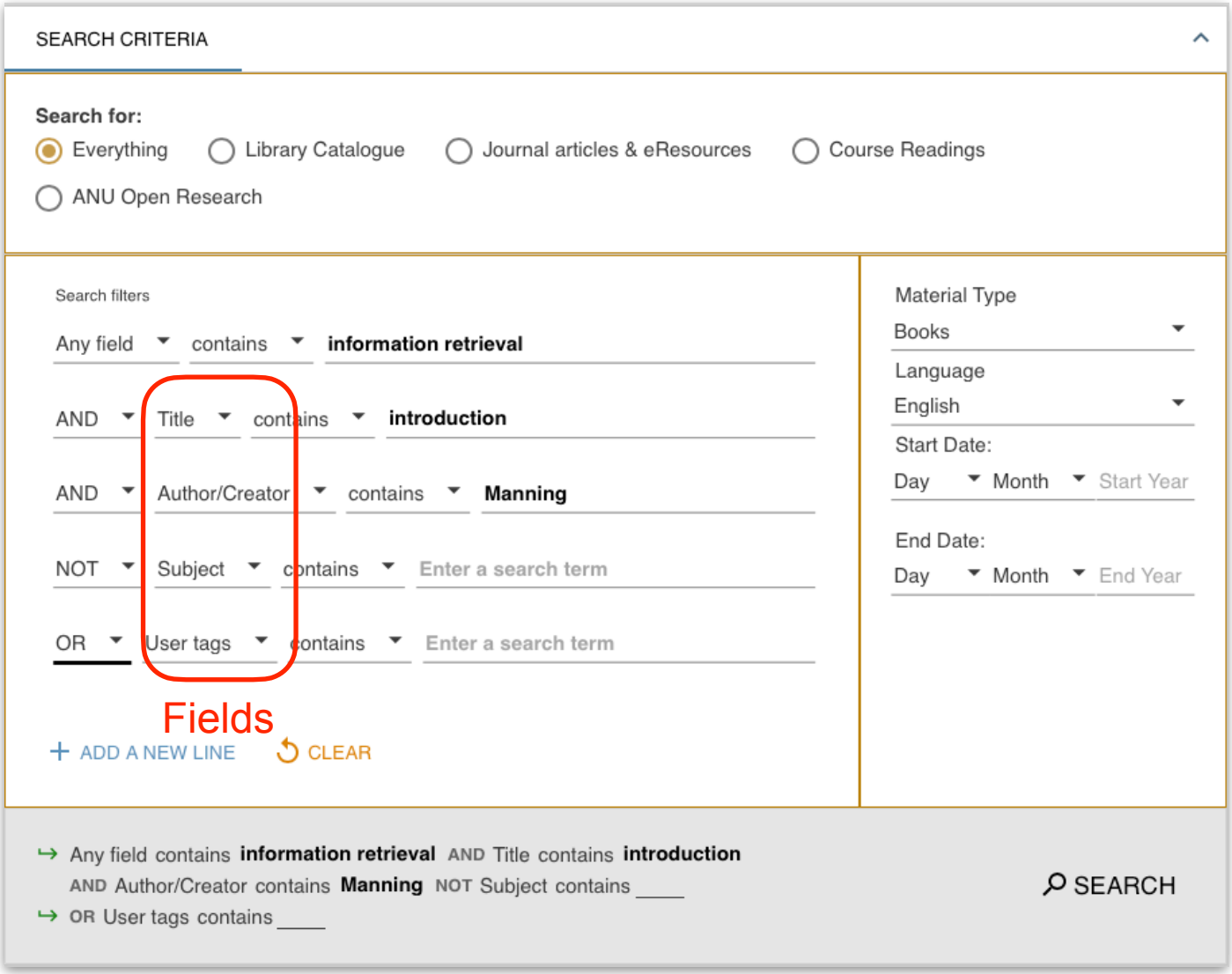
Boolean Retrieval

Boolean Retrieval with Field



Boolean Retrieval

Boolean Retrieval with Field



Boolean Retrieval

Limitations of Boolean Retrieval

- Documents either match or don't
- Good for expert users with *precise* understanding of their needs and the document collection
- Not good for the majority of users
  - Most users are incapable of writing Boolean queries
  - Or they find it takes too much effort
- Boolean queries often result in either *too few* or *too many* results
  - Query1: "bluetooth pairing iphone" => 100,000 hits
  - Query2: "bluetooth pairing iphone sony mdr-xb50" => 0 hits

Australian National University

Outline

- From Boolean retrieval to ranked retrieval
  - Bag-of-Words (BoW) model
  - Field or zone indexes
  - Boolean retrieval with field
  - Limitations of Boolean retrieval
- Ranked retrieval
  - What is ranked retrieval
  - Weighted field scoring
  - Term frequency & inverse document frequency
  - TF-IDF variants
  - Vector space model

Ranked Retrieval

What is Ranked Retrieval

- Given a query, rank documents so that *best* results are early in the list
  - When results are ranked a large result set is not an issue
  - Only show the top-k (~10) results
  - Won't overwhelm the user
- Need a scoring function
  - Document *d* and query *q*: *score(d, q)*
- Document with a larger score will be ranked before those with smaller scores

Ranked Retrieval

Weighted Field Scoring

- Simple keyword query (most users) vs. advanced search with fields (expert users)
- How can we still make use of the fields without extra effort from the user?
  - Intuitively, term importance depends on its location
  - e.g. terms in the headline of a news article are typically more important than terms in main text

Ranked Retrieval

Weighted Field Scoring

- Simple keyword query (most users) vs. advanced search with fields (expert users)
- How can we still make use of the fields without extra effort from the user?
  - Intuitively, term importance depends on its location
  - e.g. terms in the headline of a news article are typically more important than terms in main text

Assign different weights to terms based on their location (field)!

Ranked Retrieval

Weighted Field Scoring

If a query term t occurs in field i of a document d then add weight g_i to the score for d and t

- ℓ fields, Let g_i be the weight of field i and $\sum_{i=1}^{\ell} g_i = 1$
- t:a query term, d:document

$$\text{Score}(d,t) = \sum_{i=1}^{\ell} g_i \times s_i \quad \text{where} \begin{cases} s_i = 1, \text{if } t \text{ is in field } i \text{ of } d \\ s_i = 0, \text{otherwise} \end{cases}$$

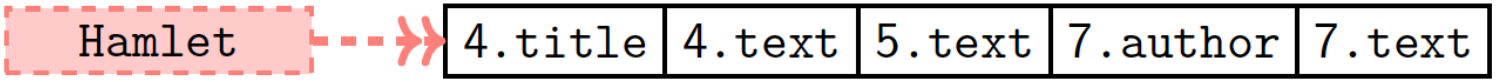
- A score of a query term ranges $[0, 1]$

Ranked Retrieval

Weighted Field Scoring

Example query: *Hamlet*

Field	Weight g_i
title	0.5
text	0.2
author	0.3



Accumulator

$$\begin{aligned} \text{Doc4} &= 0.5 + 0.2 = 0.7 \\ \text{Doc5} &= 0.2 = 0.2 \\ \text{Doc7} &= 0.3 + 0.2 = 0.5 \end{aligned}$$

Final Score

Ranked Retrieval

TF and IDF

Definition (Term Frequency (TF))

$\text{tf}_{t,d}$ is the number occurrences of term t in document d.

Rank documents based on the frequency of query terms

- The frequency of term in document is ignored in our Boolean retrieval model
- Let q be a set of query terms $(t_1, t_2, \ldots, t_m)$, a term frequency score of document d given query q is

$$score_{tf}(d, q) = \sum_{i=1}^{m} tf_{t_i,d}$$

Ranked Retrieval

TF and IDF

Example: Rank by TF

Table: Term frequency of two documents

	car	insurance	auto
doc1	1	2	3
doc2	5	0	2

- If our query q is "*car insurance*", then the score of each document is:
  - $score(doc1, q) = tf_{car,\ doc1} + tf_{insurance,\ doc1} = 1 + 2 = 3$
  - $score(doc2, q) = tf_{car,\ doc2} + tf_{insurance,\ doc2} = 5 + 0 = 5$
  - Therefore, $doc2$ is ranked higher than $doc1$
- Note:
  - Here every query term has an equal importance
  - What words are more important than others?

TF and IDF

Importance of Terms

- Every term could have a different weight
  - e.g. a collection of documents on the *auto industry* is likely to have the term *car* in almost every document
- How can we reduce the weight of terms that occur too often in the collection?
  - Use the *document frequency* of the terms

Ranked Retrieval

TF and IDF

Definition (Document Frequency)

Document frequency df_t: the number of documents in the collection that contain term t.

- Document frequency (df) is a good way to measure an importance of a term
  - High frequency => not important (e.g. stopwords)
  - Low frequency => important
- Why not *collection frequency*?
  - Collection frequency (cf): the total number of occurrences of a term in the collection

Word	cf	df
try	10422	8760
insurance	10440	3997

  - *cf* and *df* behave very differently in this example
  - *cf* is sensitive to documents that contain a word many times

TF and IDF

Definition (Inverse Document Frequency (IDF))

Let df_t be the number of documents in the collection that contain a term t. The inverse document frequency (IDF) can be defined as follows:

$$\text{idf}_t = \log \frac{N}{\text{df}_t}$$

where N is the total number of documents.

- The *idf* of a *rare* term is *high*, whereas the *idf* of a *frequent* term is likely to be *low*. For example
  - Let $N = 100$, $df_{car} = 60$, $df_{insurance} = 10$
  - Using log base 10: $idf_{car} = 0.22$, $idf_{insurance} = 1$
  - $insurance$ is 4 times more important than car
- Other log bases are sometimes used such as log base e or log base 2

Ranked Retrieval

TF and IDF

Definition (TF-IDF)

The tf-idf weight of term t in document d is as follows:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

Using *tf-idf* weights, the score of document d given query $q = (t_1, t_2, \ldots, t_m)$ is:

$$\text{Score}_{\text{tf-idf}}(d, q) = \sum_{i=1}^{m} \text{tf-idf}_{t_i,d}$$

Ranked Retrieval

TF and IDF

Example: Rank by TF-IDF

Table: Term frequency of two documents

	car	insurance	auto
doc1	1	2	3
doc2	5	0	2

Table: IDF of three terms

car	0.22
insurance	1
auto	0.8

- Given the query "*car insurance*", the score of each document is:
  - $score(doc1,\ q) = \text{tf-idf}_{car,\ doc1} + \text{tf-idf}_{insurance,\ doc1} = 2.22$
  - $score(doc2,\ q) = \text{tf-idf}_{car,\ doc2} + \text{tf-idf}_{insurance,\ doc2} = 1.1$
- Unlike the *tf* scoring approach, the score of *doc1* is greater than *doc2*

TF and IDF

A question for the audience:

Why does the log base of idf not really matter for ranking documents with the tf-idf score?

Ranked Retrieval

TF-IDF variants

Sublinear TF Scaling

- Limitation: tf-idf relies heavily on TF, it increases *linearly* with TF
  - $tf_{car,\ doc1} = 20,\ tf_{car,\ doc2} = 1$
  - If our query contains "car", then the tf-idf$_{car,\ doc1}$ is 20 times larger than tf-idf$_{car,\ doc2}$
- Claim: the marginal gain of TF should decrease as TF increases (i.e. the relationship should be *non-linear*)
  - Use logarithmically weighted term frequency (wf)

Ranked Retrieval

TF-IDF variants

Sublinear TF Scaling

- Logarithmically weighted term frequency (wf)

$$\text{wf}_{t,d} = \begin{cases} 1 + \log \text{tf}_{t,d}, & \text{if } \text{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{wf-idf}_{t,d} = \text{wf}_{t,d} \times \text{idf}_t$$

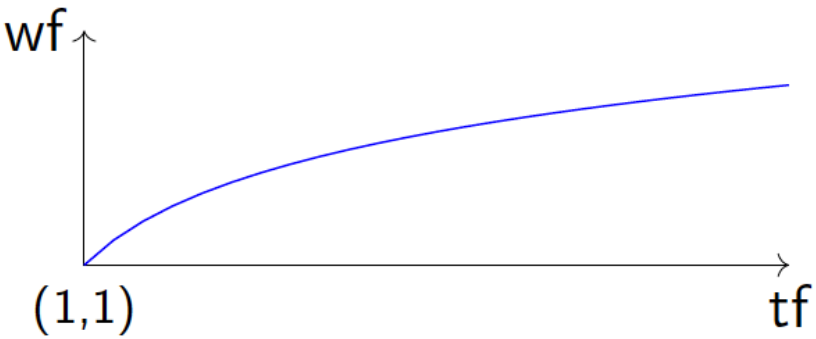


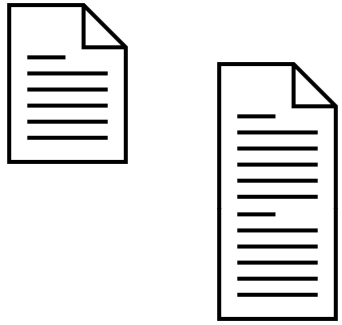
Figure: tf versus wf

Ranked Retrieval

TF-IDF variants

Maximum TF Normalisation

- Limitation: Both *tf-idf* and *wf-idf* scoring prefer *longer* documents
  - Assume we have document d
  - Create a new document d' by appending a copy of d to itself



  - While d' should be no more relevant to any query than d, their scores are different!
  - $\text{score}_{\text{tf-idf}}(d', q) \geq \text{score}_{\text{tf-idf}}(d, q)$
  - $\text{score}_{\text{wf-idf}}(d', q) \geq \text{score}_{\text{wf-idf}}(d, q)$
- Normalise the *tf* weights by the maximum *tf* in a document

Ranked Retrieval

TF-IDF variants

Maximum TF Normalisation

- Let $\mathrm{tf}_{\max}(d)$ be the maximum frequency of any term in document d, i.e. $\mathrm{tf}_{\max}(d) = \max_{t \in d} \mathrm{tf}_{t,d}$
- Normalised term frequency is defined as

$$\mathrm{ntf}_{t,d} = \alpha + (1 - \alpha) \frac{\mathrm{tf}_{t,d}}{\mathrm{tf}_{\max}(d)}$$

  - Smoothing term $0 < \alpha < 1$, usually set to 0.4
  - $\alpha \leq \mathrm{ntf} \leq 1$
- This approach also has limitations, e.g. it can be skewed by a single term that occurs too frequently (outlier or stopword)

Ranked Retrieval

Vector Space Model

Documents as Vectors

- Given a term-document matrix
  - A document can be represented as a vector of length V
  - $V =$ size of vocabulary

	doc1	doc2	doc3
car	1	7	6
insurance	4	2	11

Table: Term-document matrix with tf

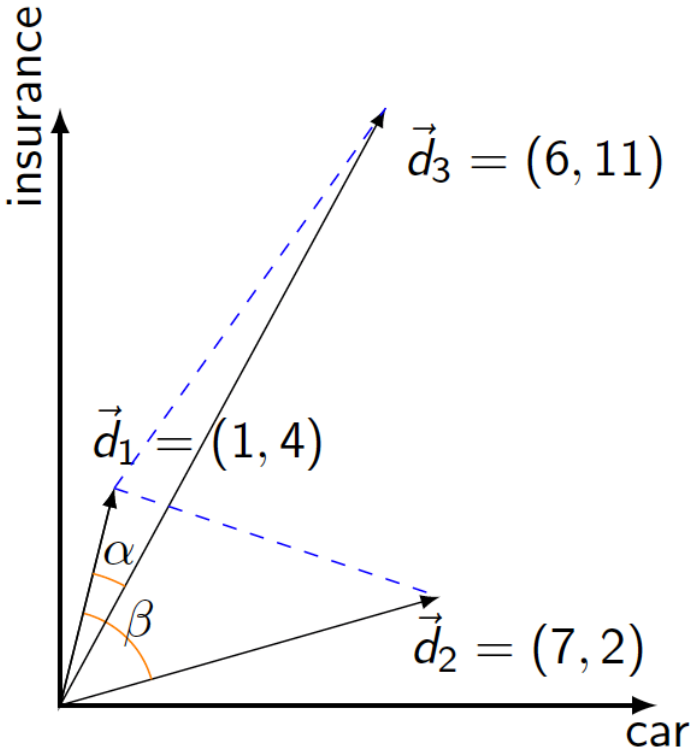
- Document vectors example
  - doc1 $= [1,4]$, doc2 $= [7,2]$, doc3 $= [6,11]$

Ranked Retrieval

Vector Space Model

Document Similarity in Vector Space

- Plot document vectors in vector space
- How to find similar documents in vector space?
  - Distance from vector to vector
  - Angle difference between vectors



Ranked Retrieval

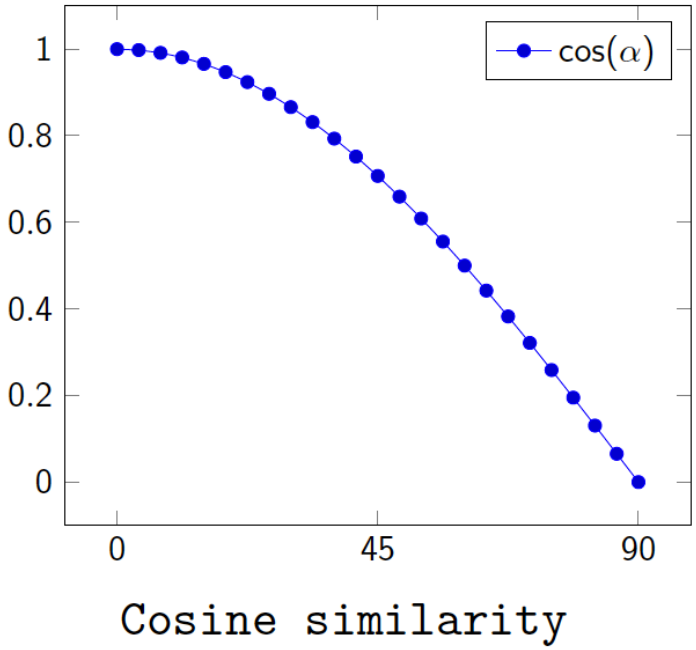
Vector Space Model

Cosine Similarity

- Standard way of quantifying similarity between documents

$$\text{sim}(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{|\vec{d}_1| \times |\vec{d}_2|}$$

  - Numerator: inner product
  - Denominator: product of Euclidean lengths
  - 1 if directions of two vectors are the same
  - 0 if directions of two vectors are orthogonal



Ranked Retrieval

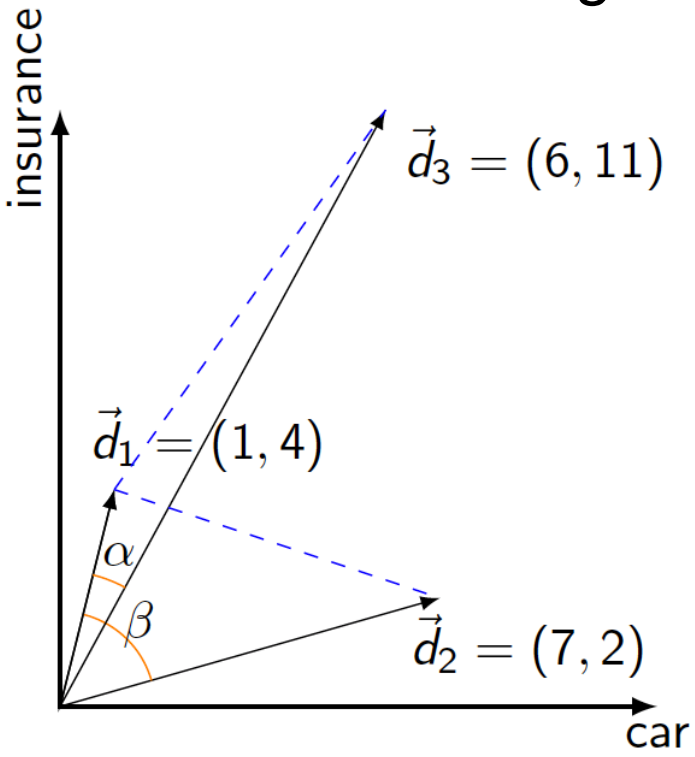
Vector Space Model

Cosine Similarity: Why not Euclidean distance?

- Euclidean distance is sensitive to the vector magnitude (related to how many terms are in a document)
- Cosine similarity is not sensitive to vector magnitude

- Euclidean distance l of *normalised* vectors vs. cosine similarity $\cos(\alpha)$:

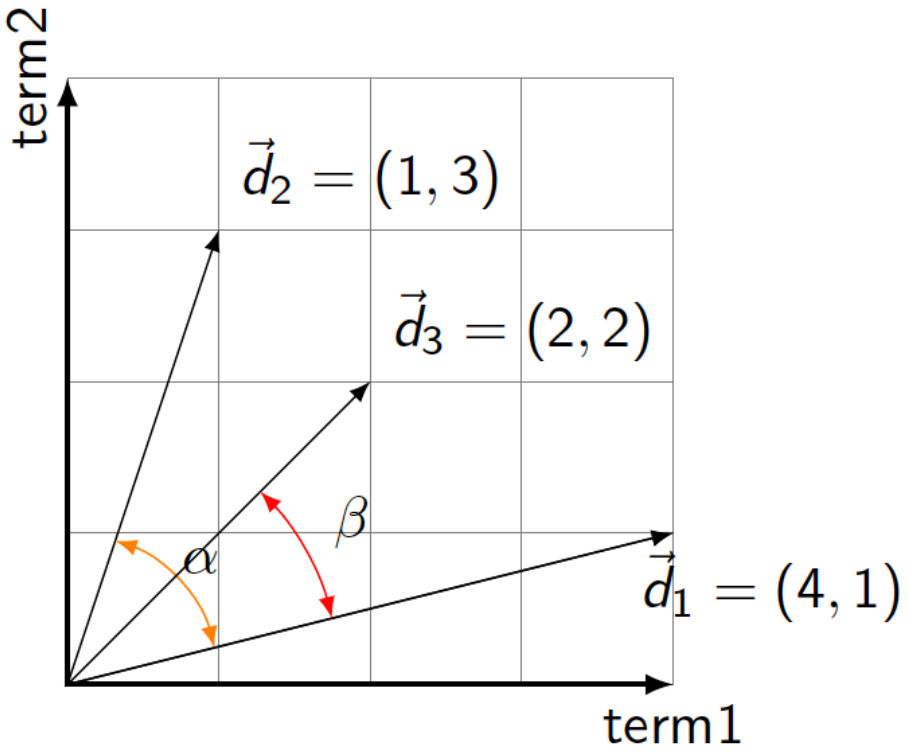
$$l^2 = 2(1 - \cos(\alpha))$$



Ranked Retrieval

Vector Space Model

Cosine Similarity: Example



$$\text{sim}(\vec{d}_1, \vec{d}_2) = \cos(\alpha) = \frac{7}{\sqrt{17}\sqrt{10}} = 0.54$$

$$\text{sim}(\vec{d}_1, \vec{d}_3) = \cos(\beta) = \frac{10}{\sqrt{17}\sqrt{8}} = 0.86$$

Ranked Retrieval

Vector Space Model

Cosine Similarity: Example with TF

	doc1	doc2	doc3
auto	27	4	24
best	3	33	0
car	0	33	29
insurance	14	0	17

Table: Term-document matrix with tf

$$sim(\vec{d_1}, \vec{d_2}) = \frac{27 \times 4 + 3 \times 33}{\sqrt{27^2 + 3^2 + 14^2} \times \sqrt{4^2 + 33^2 + 33^2}}$$

$$\text{sim}(\vec{d}_1, \vec{d}_2) = 0.15$$

$$\text{sim}(\vec{d}_2, \vec{d}_3) = 0.55$$

$$\text{sim}(\vec{d}_1, \vec{d}_3) = 0.70$$

Ranked Retrieval

Vector Space Model

Cosine Similarity: Example with TF-IDF

	doc1	doc2	doc3
auto	6.5	3.2	7.6
best	2.3	4.2	0
car	0	6.7	2.6
insurance	7.5	0	5.4

Table: Term-document matrix with tf-idf

If the vector representations of documents are very sparse (many zeros)! Cosine similarity ignores zeros in either vector.

$$sim(\vec{d_1}, \vec{d_2}) = \frac{6.5 \times 3.2 + 2.3 \times 4.2}{\sqrt{6.5^2 + 2.3^2 + 7.5^2} \times \sqrt{3.2^2 + 4.2^2 + 6.7^2}}$$

Ranked Retrieval

Vector Space Model

Queries as Vectors

- We have represented documents as vectors, a query can also be represented as a vector
  - Pretend it is a document and use query terms to construct the vector
  - When using tf-idf the *document frequency* for the query vector comes from the document collection
- We can compute the similarity between the query vector and a document vector using cosine similarity

Ranked Retrieval

Vector Space Model

Queries as Vectors

- The score function for the vector space model

$$\text{Score}_{\text{vsm}}(d, q) = sim(\vec{d}, \vec{q})$$

- If our query is "`auto insurance`"

	doc1	doc2	doc3	query
auto	27	4	24	1
best	3	33	0	0
car	0	33	29	0
insurance	14	0	17	1

Table: What will be the $\text{sim}(\vec{d}_n, \vec{q})$?

- We can rank documents using the cosine similarities between the query vector and the vector of each document

Summary

- From Boolean retrieval to ranked retrieval
  - Bag-of-Words (BoW) model
  - Field or zone indexes
  - Boolean retrieval with field
  - Limitations of Boolean retrieval
- Ranked retrieval
  - What is ranked retrieval
  - Weighted field scoring
  - Term frequency & inverse document frequency
  - TF-IDF variants
  - Vector space model

References

- Chapter 6, Introduction to Information Retrieval
- Some lecture slides are from:
  - Pandu Nayak and Prabhakar Raghavan, CS276
  - Information Retrieval and Web Search, Stanford University