Australian National University

COMP4650/6490 Document Analysis

Autumn - 2021

Ranked Retrieval

Research School of Computer Science, ANU

Table of Contents

Last time…

- We learned
  - Boolean retrieval
  - Inverted index data structure
  - Tokenization and other preprocessing steps

Bag-of-Words

- 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 of words
  - Doc1: Mary married John
  - Doc2: John married Mary
  - These two documents are the same under BoW assumption
- We will use the BoW assumption throughout IR part
- NLP part will cover other approaches that care about ordering

Field (Zone) in Document

- Document is a semi-structured data
  - Title
  - Author
  - Published date
  - Body
  - …
- Someone may want to limit search scope within a certain field
  - Partially solve the problem with BoW

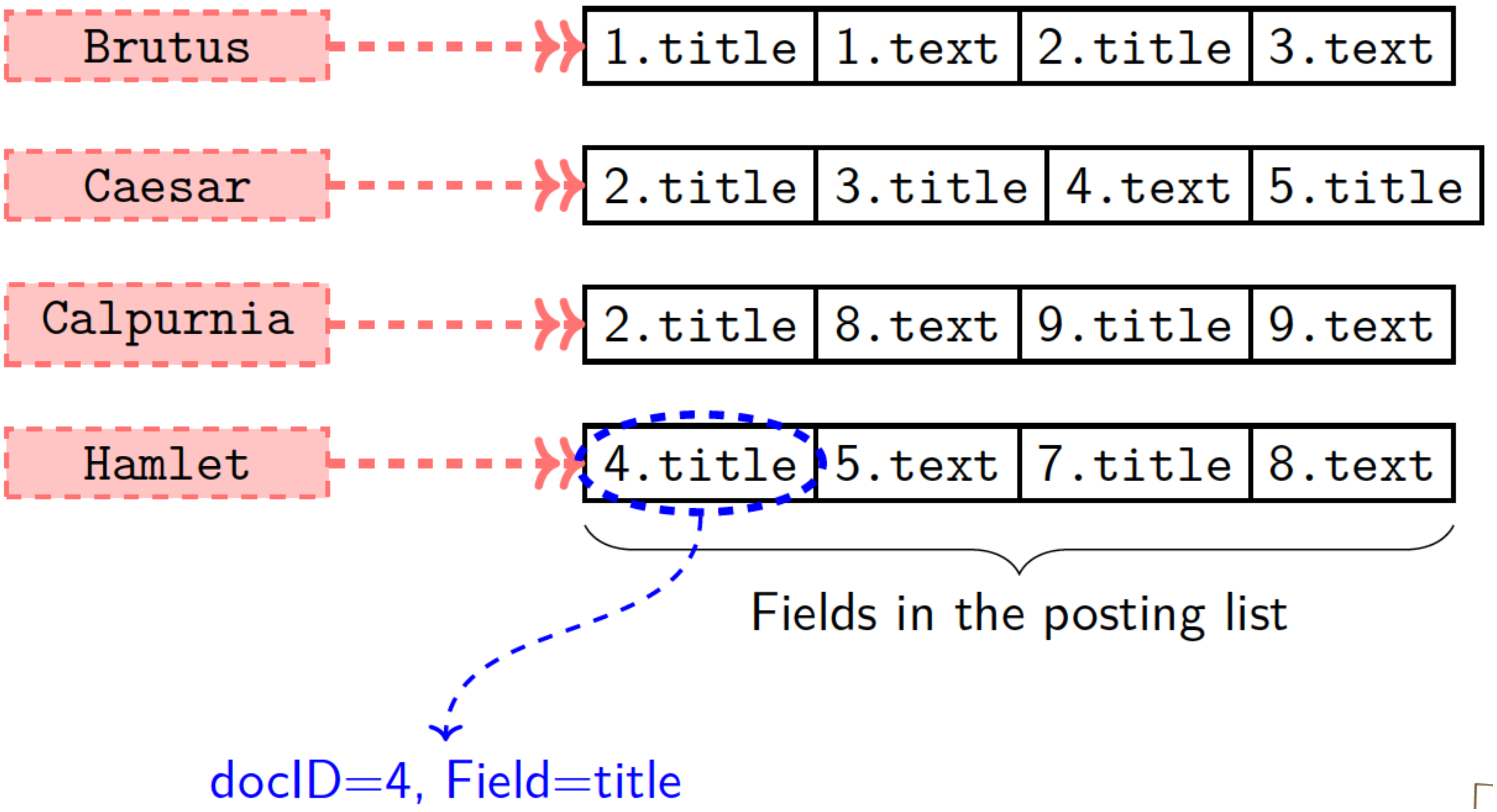
Basic Field Index

`Brutus.title` →	1	2	4	8	9	13
`Brutus.text` →	1	3	8	16		
`Caesar.title` →	2	5	7			
`Caesar.text` →	3	4	8	11		
`Calpurnia.title` →	2	9	13			

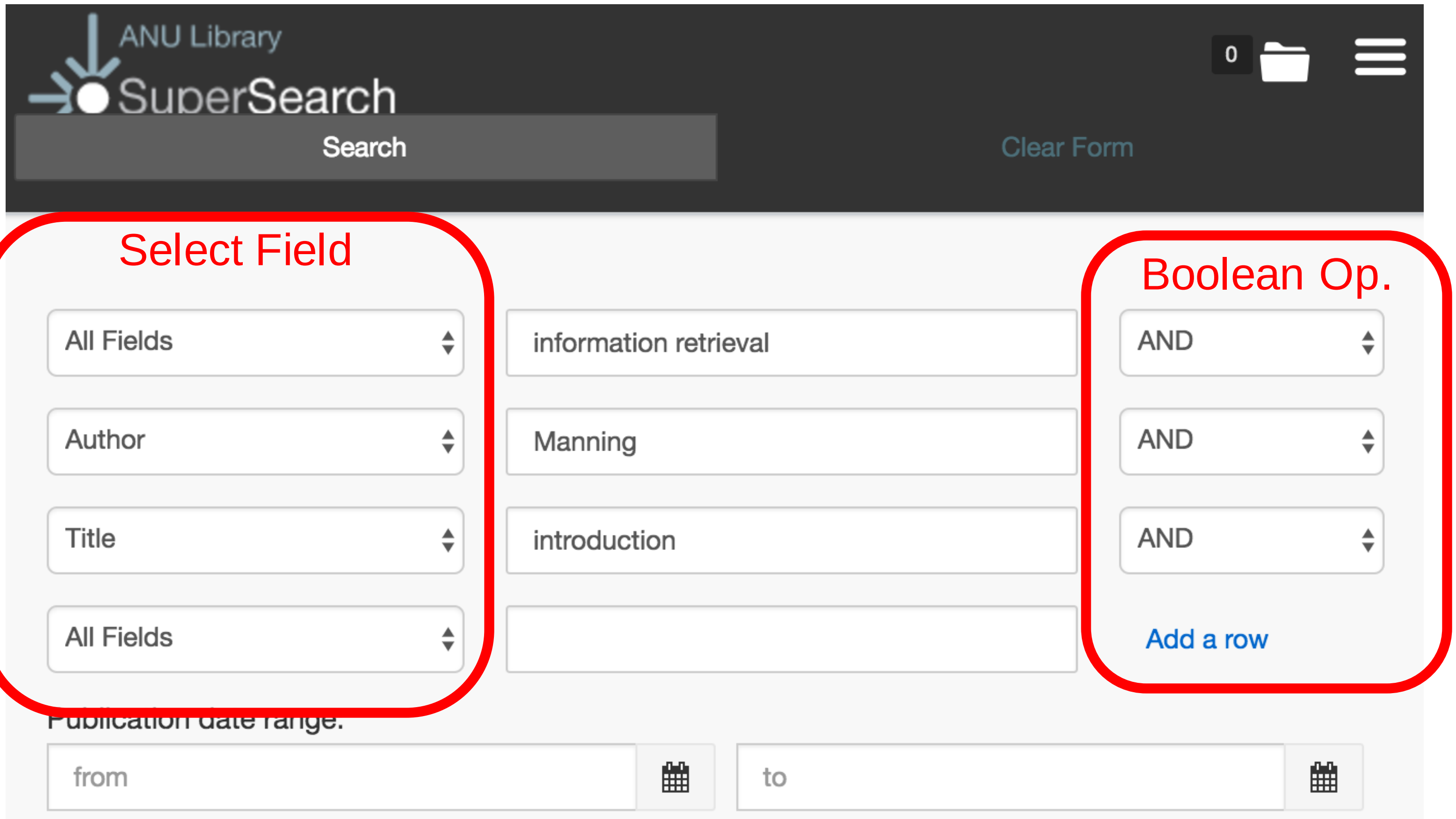
Basic field index

Fields are encoded as extension of dictionary entries

Field in Posting



Boolean Retrieval with Field



ANU Library Advanced Search

Table of Contents

Limitations of Boolean 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
- **Not good for the majority of users**
  - Most users are incapable of writing Boolean queries
  - Or they are, but they think it's too much work
- 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

Ranked Retrieval

Ranked Retrieval

Given a query, rank documents according to some criterion so that the "best" results appear early in the result list displayed to the user.

The goal if ranked retrieval is to find a scoring function

$$\mathrm{Score}(d, q)$$

where d is a document q is query.

- When a system produces a ranked result set, large result sets are not an issue.
- We just show the top k(~10) results.
- We don't overwhelm the user.

Table of Contents

- Introduction to Boolean Retrieval
  - Bag-of-Words and Document Fields
- Ranked Retrieval
  - What is ranked retrieval?
  - Weighted Field scoring
  - Term frequency and inverse document frequency
  - Variant tf-idf
  - Vector space model
- Relevance Feedback

Weighted Fields Approach

- Advanced search is for experts
  - Still majority users use a set of keywords as a query
- Importance of term is not the same
  - Terms in headline of news article is more important than terms in main text.

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

Scoring with Weighted Fields

- 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 $[0, 1]$

Example Query: Hamlet

Field	Weight g_i
title	0.5
text	0.2
author	0.3

Hamlet → | 4.title | 4.text | 5.text | 7.author | 7.text |

$$\text{Accumulator}\left\{\begin{array}{ll} \text{Doc4} = 0.5 + 0.2 & = 0.7 \\ \text{Doc5} = 0.2 & = 0.2 \\ \text{Doc7} = 0.3 + 0.2 & = 0.5 \end{array}\right\}\text{Final Score}$$

Table of Contents

- Introduction to Boolean Retrieval
  - Bag-of-Words and Document Fields
- Ranked Retrieval
  - What is ranked retrieval?
  - Weighted Field scoring
  - Term frequency and inverse document frequency
  - Variants of tf-idf
  - Vector space model

Rank by Term Frequency

Definition (Term Frequency (TF))

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

- So far we ignored the frequency of term *t* in document *d*.
- Rank based on the frequency of query terms in documents
- Let q be a set of query terms (t_1, t_2, …, t_m), a term frequency score of documents given query *q* is

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

TF Rank Example

Table: Term frequency of two documents

	car	insurance	auto
doc1	1	2	3
doc2	5	0	2

- If our query is “car insurance”, then score of each document is:
  - Score(doc1, q) = $tf_{car,doc1} + tf_{insurance,doc1} = 3$
  - Score(doc2, q) = $tf_{car,doc2} + tf_{insurance,doc2} = 5$
- Therefore rank of doc2 is higher than doc1
- Every query term has an equal importance
  - What if insurance is more important than car?

Importance of Terms

- In reality, every term has 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 to mitigate the effect of terms that occur too often in the collection?
  - →Use document frequency of term.

Document Frequency

Definition (Document Frequency)

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

- df is a good way to measure an importance of a term.
  - High frequency → not important (like stopwords)
  - Low frequency → important
- Why not collection frequency? (The total number of occurrences of a term in the collection.)

Word	cf	df
try	10422	8760
insurance	10440	3997

Inverse Document Frequency

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:

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

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.
  - E.g., Let N $= 100$, $\mathrm{df}_{car} = 60$, $\mathrm{df}_{insurance} = 10$
  - $\mathrm{idf}_{car} = 0.22$, $\mathrm{idf}_{insurance} = 1$

→ insurance is 4 times more important than car.

TF-IDF

Definition (TF-IDF)

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

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

- With tf-idf weighting scheme, the score of document d given query $q = (t_1, t_2, \ldots, t_m)$ is

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

TF-IDF Example

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 query "car insurance", then 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 tf-based scoring approach, score of doc1 is greater than doc2.

Table of Contents

Limitation of **tf-idf** scoring

- **tf-idf** still heavily relies on the frequency of terms.
- Assume
  - $tf_{car, doc1} = 20$
  - $tf_{car, doc2} = 1$
  - If our *query* contains car, $\mathbf{tf\text{-}idf_{car}}$ score of *doc1* is 20 times significant than doc2.
- Or is there a big difference between frequency 10 and 20?
- Score *linearly increases* with respect to frequency of term
- After a certain frequency, the absolute frequency isn't important.

Sublinear tf scaling

- Use logarithmically weighted term frequency (wf)

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

- Logarithmic term frequency version of **tf-idf**

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

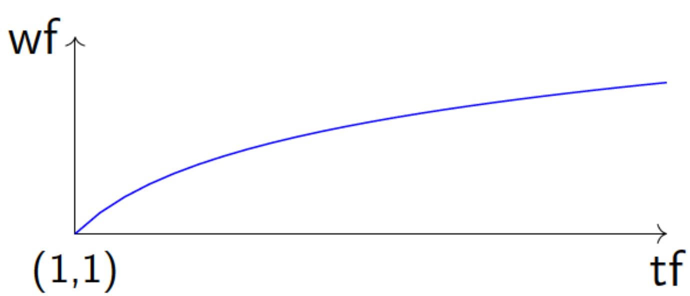


Figure: tf versus wf

Limitation of tf-idf/wf-idf scoring

- Assume we have document *d*
- We create a new document *d'* by appending a copy of *d* to itself (d' = d X 2).
- While d' should be no more relevant to any query than *d*, their scores are different!
  - $Score_{tf\text{-}idf}(d', q) >= Score_{tf\text{-}idf}(d, q)$
  - $Score_{wf\text{-}idf}(d', q) >= Score_{wf\text{-}idf}(d, q)$
  - Both scoring prefers longer documents.

Maximum tf normalization

- Let $tf_{max}(d)$ be the maximum frequency of document *d*
- Normalized term frequency is defined as

$$\mathrm{ntf}_{t,d} = \alpha + (1-\alpha)\frac{\mathrm{tf}_{t,d}}{\mathrm{tf}_{\max}(d)} \quad \text{if } \mathrm{tf}_{t,d} > 0$$

  - Maximum value of ntf is 1
  - Minimum value of ntf is α
- Again, this approach has a limitation too.

Table of Contents

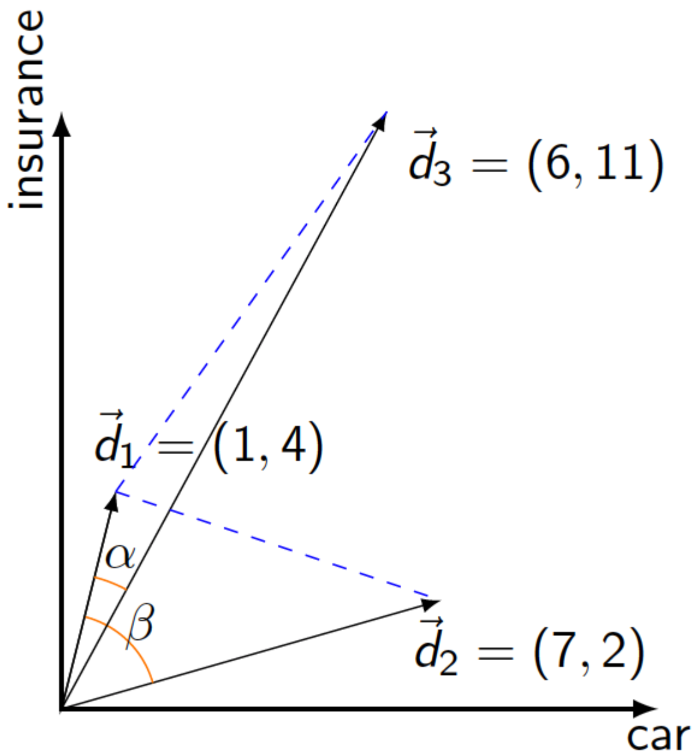
Document as Vectors

	doc1	doc2	doc3
car	1	7	6
insurance	4	2	11

Table: Term-document matrix with tf

- Given a term-document matrix
  - a document can be represented as a vector of length V
  - V = size of vocabulary
- Document vectors:
  - $d_1 = (1, 4)$, $d_2 = (7, 2)$, $d_3 = (6, 11)$

Document Similarity in Vector Space



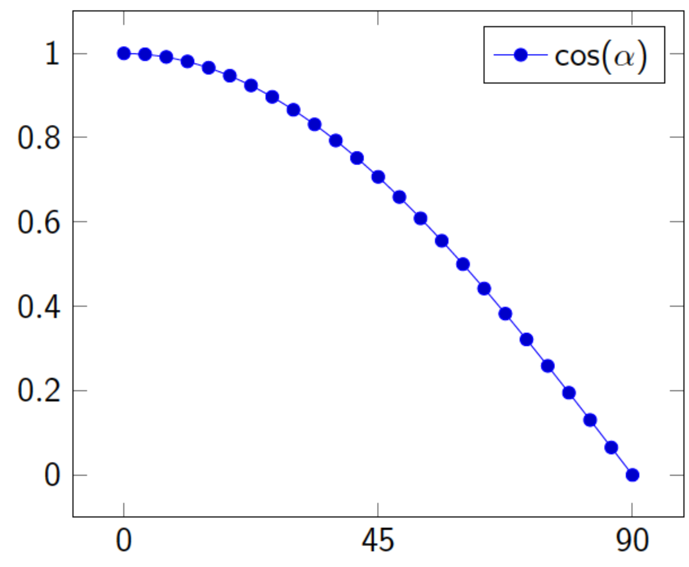
- Plot document vectors in vector space
- How to find similar documents in vector space?
  - Distance from vector to vector
  - Angle difference between vectors

Angle Difference

- **Cosine similarity**:

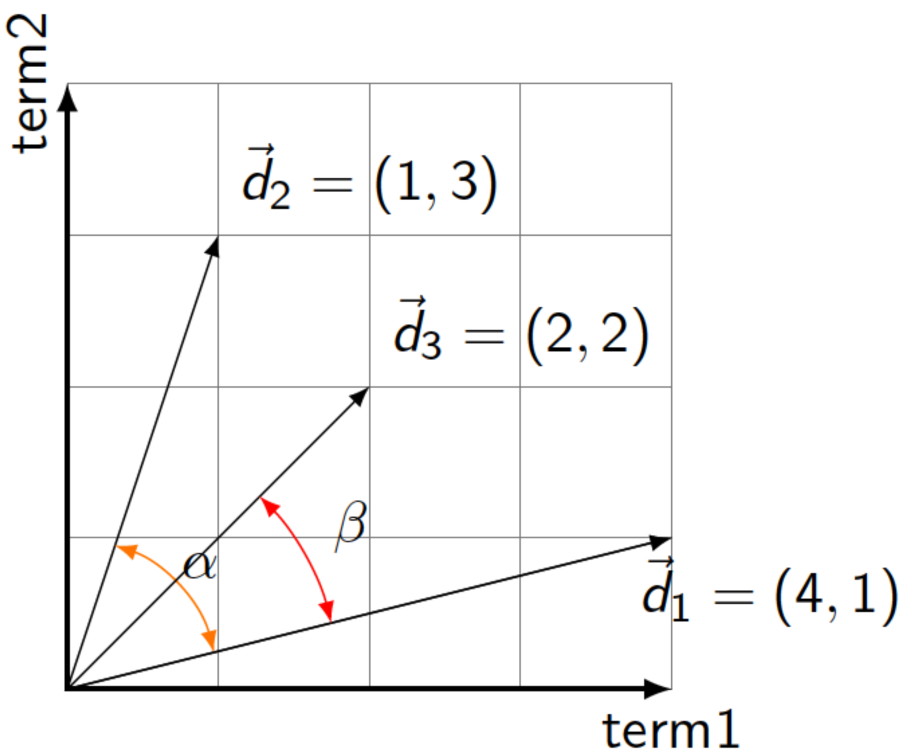
$$\text{sim}(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{|\vec{d}_1| \times |\vec{d}_2|}$$

  - Numerator: inner product
  - Denominator: product of Euclidean lengths
- Standard way of quantifying similarity between documents
  - **1** if directions of two vectors are the same
  - **0** if directions of two vectors are orthogonal (90)



Cosine similarity

Cosine Similarity: Example



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

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

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

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

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

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

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

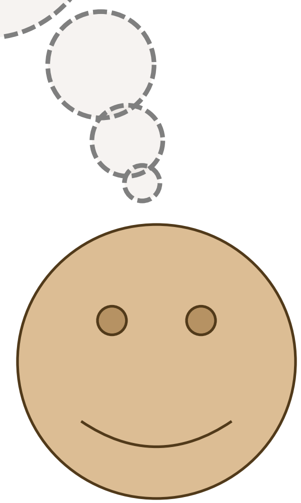
Cosine Similarity: Example with tf-idf

	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

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

Vector representation of documents in reality is very sparse! Dense vector representation? → Latent Semantic Analysis



Query as Document

- So far, we have considered a document as a vector
- Query can be converted as vector too
- Compute the similarity between query and document in the same way

- 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 sim($\vec{d}_n$, $\vec{q}$)?

Score Function of Vector Space Model

- Therefore the score function of the vector space model is

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

Summary

- Introduction to Boolean Retrieval
  - Bag-of-Words and Document Fields
- Ranked Retrieval
  - What is ranked retrieval?
  - Weighted Field scoring
  - Term frequency and inverse document frequency
  - Variants of tf-idf
  - Vector space model

References

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