INFORMATION RETRIEVAL

DISCUSSION SESSIONS 1A AND 1B

Session 4

Based on Young Cha guidelines

PROJECT

Project 2: Due today at 11:55 p.m.

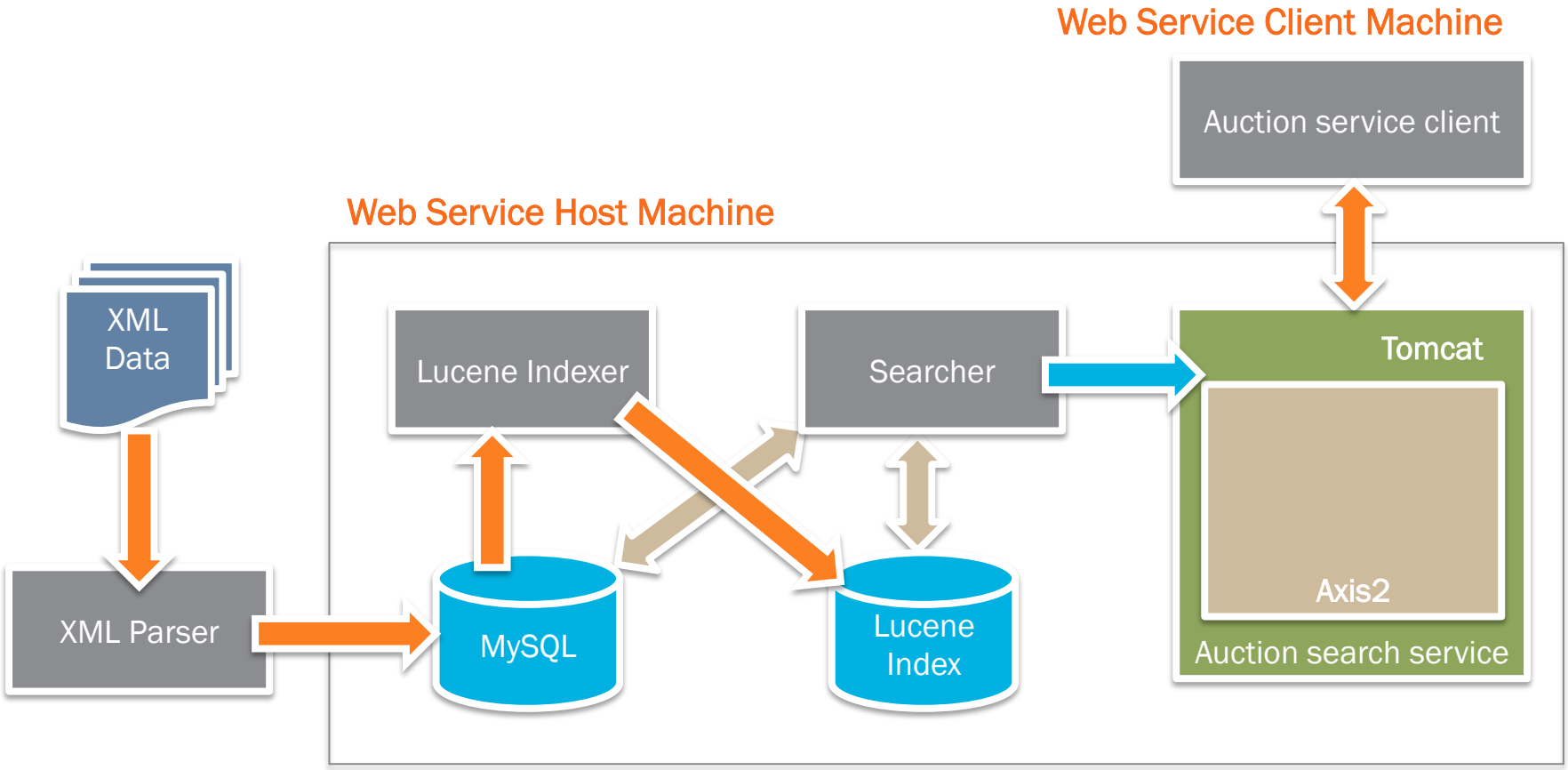
Project 3: To be released. A two-part submission:

- Building indexes (No grace period!) – Today's topic.
- Implement Java search functions.
- Deploy a web service.

Notes:

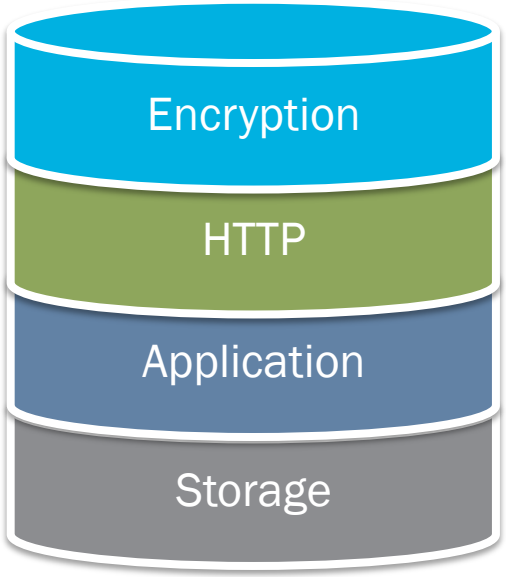
- Penalty of up to 20% for not submitting first part.
- Part A is not graded, but if you resubmit it, please document the changes in the `README.txt` file.

PROJECT 3 OVERVIEW



WEB SERVER ARCHITECTURE

- Why is it necessary?



THE BOOLEAN MODEL

Three documents:

- Doc1: *Alice visits Wonderland*
- Doc2: *Wonderland welcomes Alice*
- Doc3: *Alice! Run, Alice!*

How does the inverted index look like?

Why is it called Boolean model?

- Boolean queries: AND | OR | NOT

It's a Bag of Words

- The order of the keywords in a query doesn't matter.

VECTOR MODEL

- A document is considered an *n*-dimensional vector, where n is the number of different terms in the lexicon.
- A query is also an *n*-dimensional vector.

	alice	visits	wonderland	welcomes	run
Alice visits wonderland					
Wonderland welcomes Alice					
Alice! Run, Alice!					

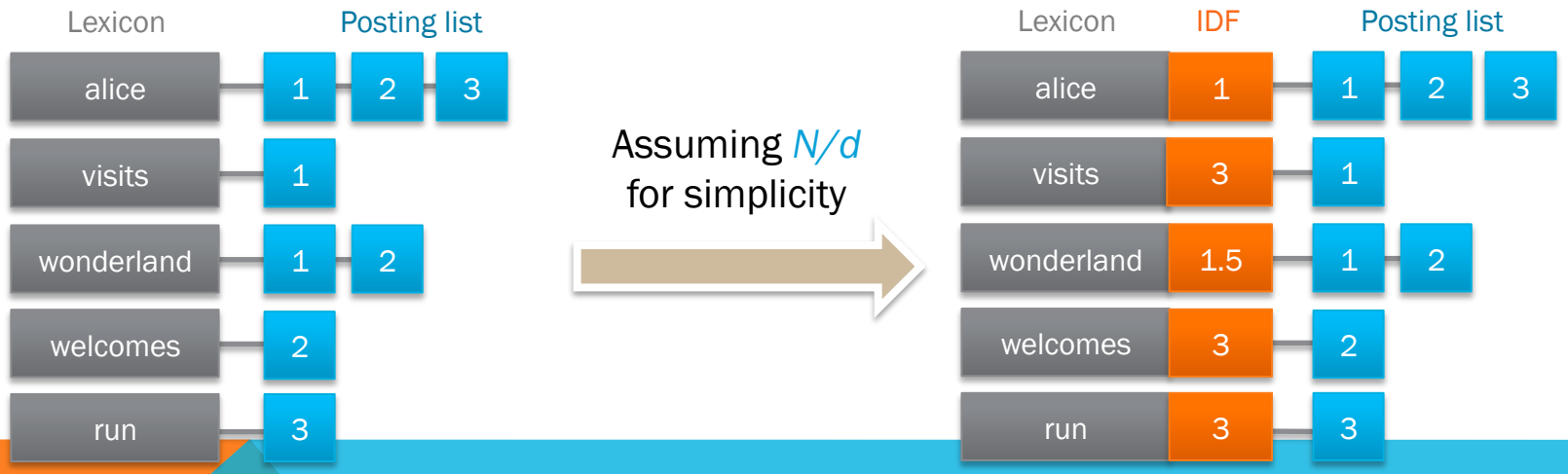
How do we fill in the components of each document's vector?

TFIDF

More common keywords are less specific – Inverse Document Frequency:

$$IDF = \ln\left(\frac{N}{d}\right)$$

where *N* is the number of documents in the corpus, and *d* is the number of documents where the term appears.



Each entry in the vector is the product of the term frequency times the term IDF

	alice	visits	wonderland	welcomes	run
Alice visits wonderland	1 x 1	1 x 3	1 x 1.5	0	0
Wonderland welcomes Alice	1 x 1	0	1 x 1.5	1 x 3	0
Alice! Run, Alice!	2 x 1	0	0	0	1 x 3

COSINE SIMILARITY

So, given the query "Run, Wonderland!," which document would be ranked #1?

- Check the cosine similarity between the query and each of the documents.

$$\cos(\theta) = \frac{\langle D_i, Q \rangle}{\| D_i \| \| Q \|}$$

	alice	visits	wonderland	welcomes	run
Run, Wonderland!	0	0	1 x 1.5	0	1 x 3

CORPUS SIZE ESTIMATION

Given that

- 100 M docs
- 5 KB/doc
- 400 unique words/doc
- 20 bytes/word
- 10 bytes/docid

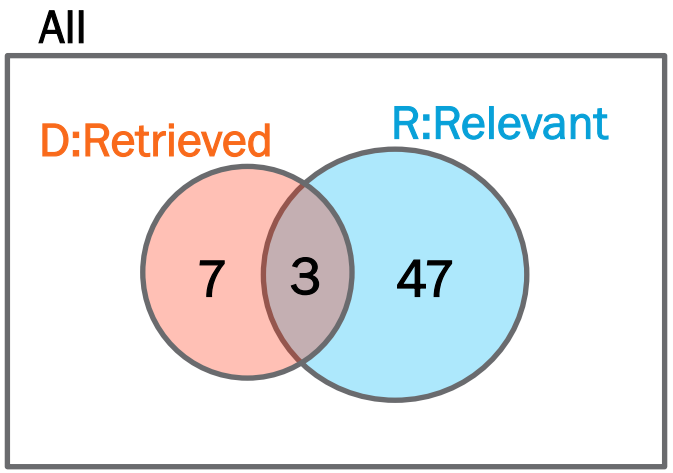
Document collection size? $100M \times 5KB \cong 500GB$

Inverted index size?

- Size of postings list? $100M \times 400 \times 10B \cong 400GB$
- Size of lexicon? (C=1, k=0.5 in $C \cdot n^k$) $(100M)^{0.5} \times 20B \cong 200KB$

PRECISION AND RECALL

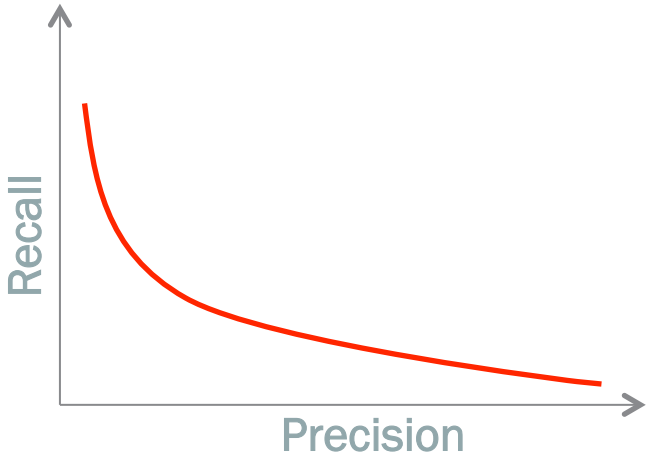
- If we have 1000 documents:
  - 50 relevant documents
- A search engine retrieves 10 documents where
  - 3 are relevant
  - 7 are irrelevant



What is the *Precision* and *Recall* of this search engine?

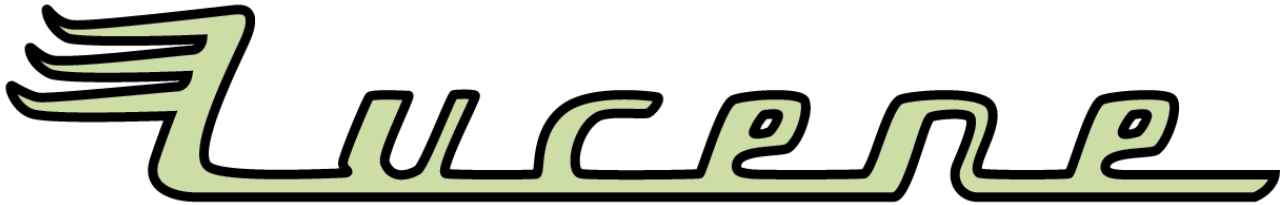
Precision = |D & R| / |D| = 3/7

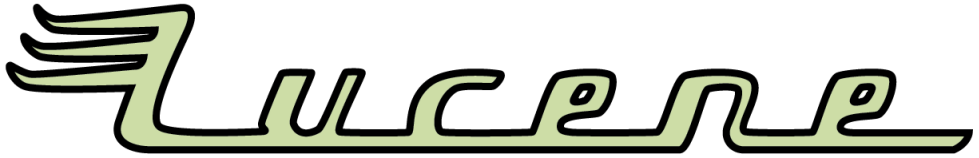
Recall = |D & R| / |R| = 3/50



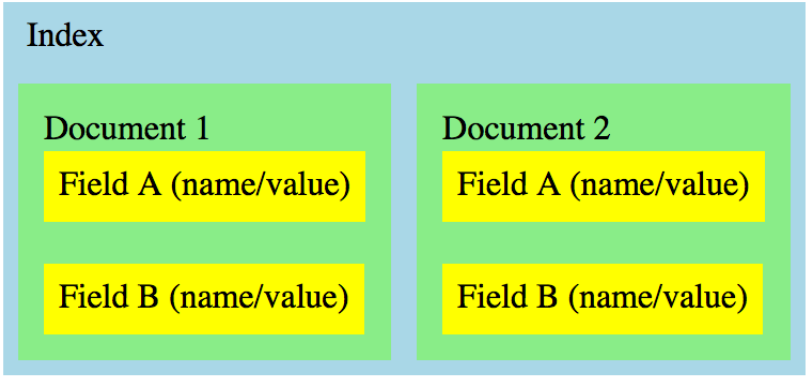
HOW TO CREATE INVERTED INDEXES?

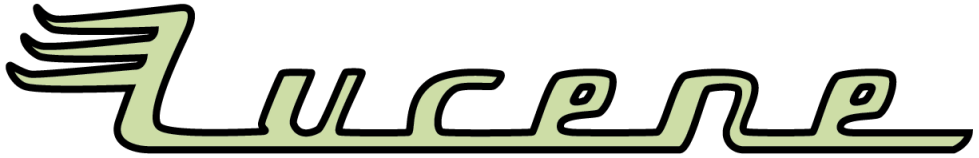
- We want to build an inverted index on Hotels information:
  - Hotel(*id*, *name*, *city*, *description*)
- Our index has to support keyword search for *id*, *name*, *city*, or the *full content* (the union of all fields).
- The search engine should display *id*, *name*, *city* and *description*.
- We'd like to make even more advanced searches, by involving a *Relational Database*.





- Treats each of the business objects as documents, which contain fields to be indexed.
- Fields can be `StringField` or `TextField`
  - A `StringField` is not broken into atomic components.
  - A `TextField` is tokenized.





- We first create the inverted index.

Analyzer	Description
`StandardAnalyzer`	A sophisticated general-purpose analyzer.
`WhitespaceAnalyzer`	A very simple analyzer that just separates tokens using white space.
`StopAnalyzer`	Removes common English words that are not usually useful for indexing. (e.g. ***the, a, is,*** ...)
`SnowballAnalyzer`	An interesting experimental analyzer that works on word roots (a search on ***rain*** should also return entries with ***raining, rained,*** and so on).

- Then, each object (Hotel) is indexed, together with its fields.
- Finally, we can perform searches on the indexed Hotels information.