Information Retrieval

CS 144 Web Application

TA: Jin Wang and Zhehan Li

02/23/2018

Overview

- Project 4
  - Install all packages
  - Setting Up Server app.js
  - Starting Server and Testing
  - Creating and Connecting to Database
  - Implementing Routing Middleware
- Information Retrieval
  - Boolean Model
  - Vector Model
    - TF-IDF
    - Cosine Similarity
  - Corpus size estimation

Project 4: Dependencies

Install all packages needed in package.json

- You can add any packages as needed in the project
- $ npm install --save mongodb
  - For database connection
- $ npm install --save commonmark
  - For generating markdown string
- $ npm install --save bcrypt
  - For encrypting user password
- $ npm install --save jsonwebtoken
  - For implementing JWT

Project 4: Dependencies

- Other recommended packages
  - **mongoose:** promise mechanism in mongoDB connection
    https://github.com/Automattic/mongoose
  - **passport:** use to authenticate requests
    https://www.npmjs.com/package/passport
  - **express-jwt:** validate JSON Web tokens
    https://www.npmjs.com/package/express-jwt

Project 4: Build a RESTful API

- Correction on last week's slides:
- You should use **JSON Web Token** for authentication sent by Session-Cookies
- Reference:

  https://www.sitepoint.com/using-json-web-tokens-node-js/

Project 4: Build a RESTful API

Setting Up Server app.js

- Configure app to use middleware
  - app.use(bodyParser.json());
  - app.use('/blog', blogMiddleware);
  - app.use('/api', apiMiddleware);
- Listen to the port:
  - For server-side: app.listen(8080);
  - For the whole application: port number should be 3000

Test the Server only

- node app.js
- You can use the tool Postman for testing:
  https://www.getpostman.com/

Project 4: Build a RESTful API

Creating and Connecting to Database

- var mongoose   = require('mongoose');
- mongoose.connect('mongodb://localhost:27017/BlogServer’);

Implementing Routing Middleware

- blogMiddleware.js // handle ‘/blog’ request
- apiMiddleware.js // handle ‘/api’ request
- Remember to make authentication!

  Example: app.get(‘/api/:name’,[authen function],[method]);

Example of using Mongoose:

https://scotch.io/tutorials/build-a-restful-api-using-node-and-express-4

Overview

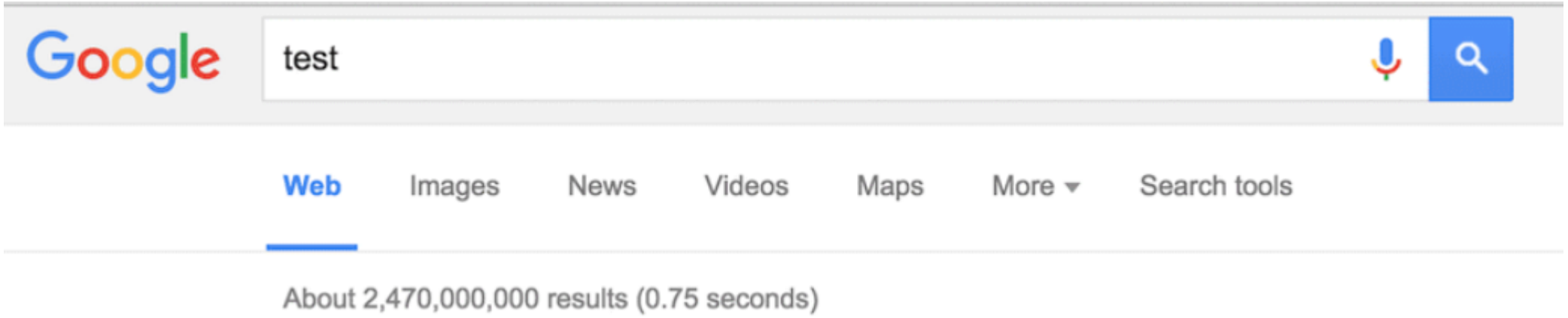
- Project 4
  - Install all packages
  - Setting Up Server app.js
  - Starting Server and Testing
  - Creating and Connecting to Database
  - Implementing Routing Middleware

- Information Retrieval
  - Boolean Model
  - Vector Model
    - TF-IDF
    - Cosine Similarity
  - Corpus size estimation

Information Retrieval

Keyword Search

- How does search engine like Google work?
- How could it be so fast?
- Which website should be the first page?



The Boolean model

Three documents:

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

Bag of words assumption: The order of the words doesn’t matter.

How does the inverted index look like?

The Boolean model

Three documents:

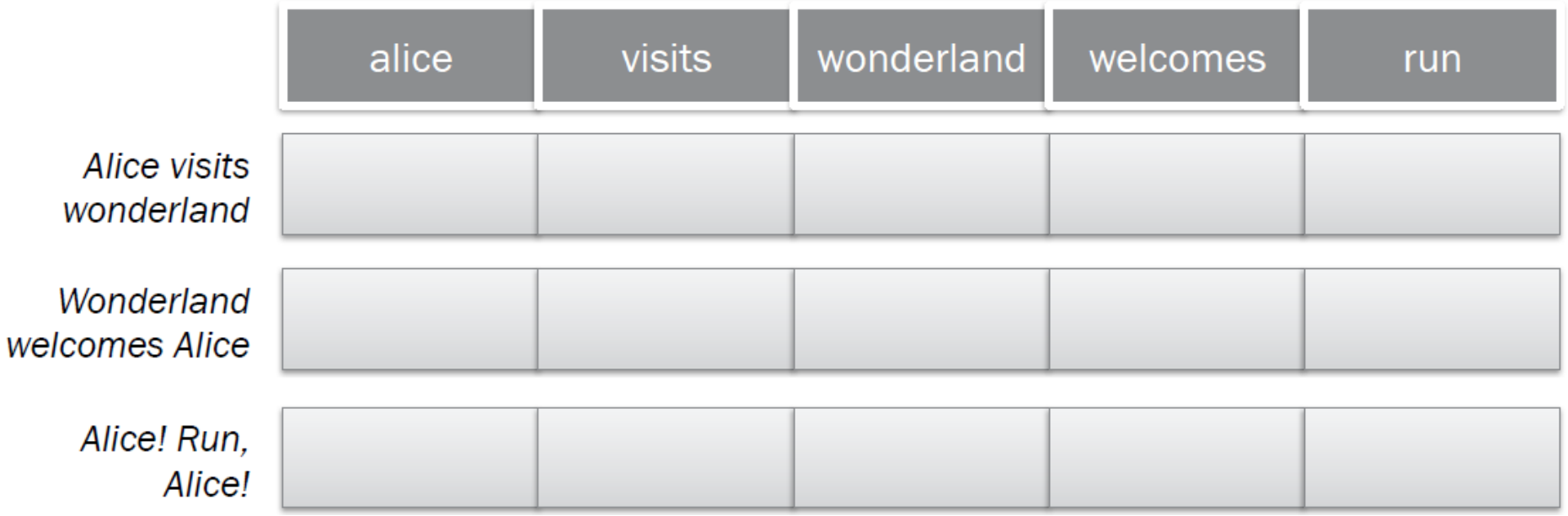
- Doc1: Alice visits Wonderland
- Doc2: Wonderland welcomes Alice
- Doc3: Alice! Run, Alice!

Why is it called Boolean model?

- Is the word in this document? - True or False
- Boolean queries: AND | OR | NOT

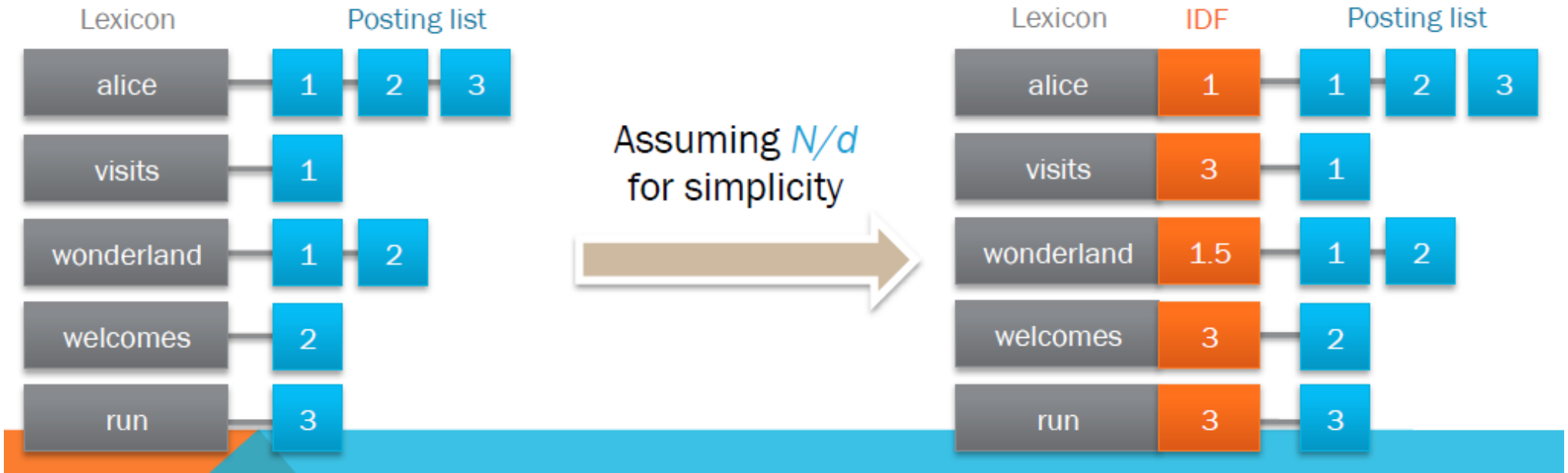
The Vector model

- 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.
- What should be the value of each vector below?
  - TF-IDF



TF-IDF

- TF (Term Frequency): # occurrence of t in D
- IDF (Inverse Doucment Frequency): log (N/DF)
  - DF (Document Frequency): # of documents where t appears.
  - More common keywords are less specific



TF-IDF

- Entry in the vector = TF × IDF:
  - e.g. “Alice” happens 2 times in third document -> TF = 2. The IDF of word “Alice” is 2
    - => TF × IDF = 2 × 1 = 2

	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

- Given the query "**Run, Wonderland!**", which document would be ranked #1?
- $cos(\theta) = \frac{<D_i, Q>}{||D_i|| \times ||Q||}$
  - D_i is the document vector
  - Q is the query vector
  - $||D_i||$ is the length of document
  - $||Q||$ can be ignored since it is same in for all documents.
- Query Vector:

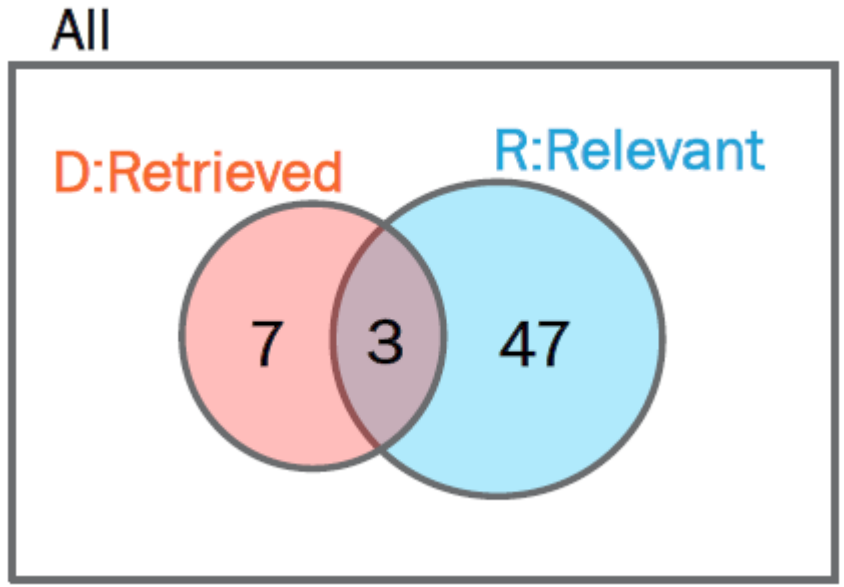
	alice	visits	wonderland	welcomes	run
Run, Wonderland!	0	0	1 x 1.5	0	1 x 3

- Document #3: "Alice! Run, Alice!" ranks #1

Corpus Size Estimation

- Given that
  - 100 M docs
  - 5 KB/doc
  - 400 unique words/doc
  - 20 bytes/word
  - 10 bytes/docid
- Size of Document Collection ?
  - 100 M docs $\times$ 5 KB / doc $\cong$ 500 GB
- Size of Inverted Index ?
  - Size of postings list?
    - 100M docs x 400 unique words/doc x 10B/docid $\cong$ 400GB
  - Size of lexicon?
    - # of unique word = $C \cdot n^k$, where n is # of documents
    - Assume C=1, k=0.5
    - $(100M)^{0.5}$ x 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