Web Services

November 1, 2021
Plan for today

- Web services
  - REST vs SOAP
  - Real REST services
- Information retrieval
  - Basics
  - Precision and recall
  - Taxonomy of IR models
Representational State Transfer (REST)

- Another example of a messaging protocol
- Not really a standard – a style of development
  - Data is represented in JSON or XML, e.g., with a schema
  - Function call interface uses HTTP Requests
    - GET/POST/PUT/PATCH/DELETE
    - Server is to be stateless
  - And the HTTP request type specifies the operation
    - e.g., GET http://my.com/rest/service1
    - e.g., POST http://my.com/rest/service1 {body} adds the body to the service
Example: The Facebook Graph API
https://developers.facebook.com/docs/graph-api

- Like many other web systems, Facebook offers API access to its system

- Programs can use the API to:
  - Read data from profiles and pages
  - Navigate the graph (e.g., via friends lists)
  - Issue queries (for posts, people, pages, ...)
  - Add or modify data (e.g., create new posts)
  - Get real-time updates, issue batch requests, ...

- How you can access it:
  - Graph API, Marketing API
Another standard for data interchange

- "JavaScript Object Notation"; MIME type application/json
- Basically legal JavaScript code; can be parsed with eval()
  - Caution: Security!
- Often used in AJAX-style applications
- Data types: Numbers, strings, booleans, arrays, "objects"
The Graph API

- Requests are mapped directly to HTTP:
  - https://graph.facebook.com/(identifier)?fields=(fieldList)

- Response is in JSON

```json
{
  "id": "1074724712",
  "age_range": {
    "min": 21
  },
  "locale": "en_US",
  "location": {
    "id": "101881036520836",
    "name": "Philadelphia, Pennsylvania"
  }
}
```
The Graph API

- Uses several HTTP methods:
  - GET for reading
  - POST for adding or modifying
  - DELETE for removing

- IDs can be numeric or names
  - /1074724712 or /vincent.liu
  - Pages also have IDs

- Authorization is via 'access tokens'
  - Opaque string; encodes specific permissions (access user location, but not interests, etc.)
  - Has an expiration date, so may need to be refreshed
Other API options

- Facebook Query Language (FQL)
  - SQL-style queries over the data provided via the Graph API
  - Example: SELECT name FROM user WHERE uid=me()
  - Supports 'multi-queries' (JSON-encoded dictionary of queries)
  - 72 different tables are available for querying

- Legacy REST API
  - See description of REST in the previous lecture
  - In the process of being deprecated
  - Large number of methods available
    - Examples: friends.get, comments.add, status.set, video.upload, ...
Other Services

- Google Maps, Google Search, Kaggle, many other sites support REST services

- Almost all require you to request an “access token” and have a limit on free requests

- Many use the OAuth standard for user authentication – log in e.g., with your Google or Facebook ID
REST Implementation in Java

REST – REpresentational State Transfer

- Basic idea: use HTTP navigation to abstract parameters in a hierarchy
  http://my.service.com/lookup/{user}/{folder}/{file}

Typically use JSON to encompass the request + response. Two useful tools used in HW3:

- Jackson allows us to serialize/de-serialize Java objects in JSON
  ObjectMapper om ...;
  str = om.writeValueAsString(myObj);
  obj = om.readValue(str);

- “Route” based programming via Spark Java

```java
public static void main(String[] args) {
    get("/myfn/:arg", (req, res) -> handle(req.body(), req.params(":arg")));
}
```
Summary

- Web services are an HTTP-based version of Remote Procedure Calls
  - Calls are *synchronous*, ie they wait for the server to return

- Key issue: how do we marshal parameters?
  - Pass by value, but may need to pass along typing info, references, etc.
  - Formalized W3C standards (SOAP, WSDL) vs ad hoc REST (+ JSON)
Plan for today

- Web services
- Information retrieval
  - Basics
  - Precision and recall
  - Taxonomy of IR models
- Classic IR models
  - Boolean model
  - Vector model
  - TF/IDF
Web search

- Goal is to find information relevant to a user’s interests - and this is hard!

- Challenge 1: Data quality
  - A significant amount of content on the web is not quality information
  - Many pages contain nonsensical rants, etc.
  - The web is full of misspellings, multiple languages, etc.
  - Many pages are designed not to convey information – but to get a high ranking (e.g., SEO, clickbait, fake news etc.)

- Challenge 2: Scale
  - Billions of documents

- Challenge 3: Very little structure
  - No explicit schema
  - However, hyperlinks and tags encode information!
Our discussion of web search

- Begin with traditional information retrieval
  - Document models
  - Stemming and stop words

- Web-specific issues
  - Crawlers and robots.txt (already discussed)
  - Scalability
  - Models for exploiting hyperlinks in ranking
    - Google and PageRank
    - Latent Semantic Indexing
Information Retrieval

- Traditional information retrieval is basically text search
  - A corpus or body of text documents, e.g., in a document collection in a library or on a CD
  - Documents are generally high-quality and designed to convey information
  - Documents are assumed to have no structure beyond words

- Searches are generally based on meaningful phrases, perhaps including predicates over categories, dates, etc.
  - The goal is to find the document(s) that best match the search phrase, according to a search model

- Assumptions are typically different from Web: quality text, limited-size corpus, no hyperlinks
Motivation for Information Retrieval

- Information Retrieval (IR) is about:
  - Representation
  - Storage
  - Organization of
  - And access to “information items”

- Focus is on user’s information need rather than a precise query:
  - User enters: “March Madness”
  - Goal: Find information on college basketball teams which (1) are maintained by a US university and (2) participate in the NCAA tournament

- Emphasis is on the retrieval of information (not data)
Data vs. Information Retrieval

- **Data** retrieval, analogous to database querying: which docs contain a set of keywords?
  - Well-defined, precise logical semantics
    - Example: All documents with ('CIS455' OR 'CIS555') AND ('midterm')
  - A single erroneous object implies failure!

- **Information retrieval:**
  - Information about a subject or topic
  - Semantics is frequently loose; we want approximate matches
  - Small errors are tolerated (and in fact inevitable)

- **IR system:**
  - Interpret contents of information items
  - Generate a ranking which reflects relevance
  - Notion of relevance is most important – needs a model
Basic model

Docs

Index Terms

Information Need

Query

Match

Ranking

© 2021 A. Haeberlen, Z. Ives, V. Liu
Information Retrieval as a field

- IR addressed many issues in the last 30 years:
  - Classification and categorization of documents
  - Systems and languages for searching
  - User interfaces and visualization of results
- Area was seen as of narrow interest – libraries, mainly

- And then – the advent of the web:
  - Universal “library”
  - Free (low cost) universal access
  - No central editorial board
  - Many problems in finding information: IR seen as key to finding the solutions!
The full Information Retrieval process

1. **Text Processing and Modeling**
   - **Text** from the Browser/UI flows into the Text Processing and Modeling stage.
   - User interest influences this process.

2. **Query Operations**
   - A query is generated and sent to the Searching stage.
   - User feedback is also considered.

3. **Searching**
   - The query is processed to retrieve documents.
   - Retrieved documents are available.

4. **Indexing**
   - Indexing involves creating an inverted index.
   - Indexing uses the logical view.
   - Inverted index is a key output.

5. **Ranking**
   - Rankings of documents are produced based on the query.
   - Ranked documents are generated.

6. **Crawler/Data Access**
   - Documents from the Web or DB are accessed and used for retrieval.

The process involves:
- Retrieval of documents
- Query processing
- Index creation
- Ranking of documents
- User interaction and feedback
Terminology

- IR systems usually adopt index terms to process queries

- Index term:
  - a keyword or group of selected words
  - any word (more general)

- Stemming or lemmatization might be used:
  - connect: connecting, connection, connections

- An inverted index is built for the chosen index terms
What is a meaningful result?

- Matching at index term level is quite imprecise
  - Users are frequently dissatisfied
  - One problem: users are generally poor at formulating queries
  - Frequent dissatisfaction of Web users (who often give single-keyword queries)

- Issue of deciding relevance is critical for IR systems: ranking
  - Show more relevant documents first
  - May leave out documents with low relevance
Precision and recall

- How good is our IR system?
- Two common metrics:
  - **Precision**: What fraction of the returned documents is relevant?
  - **Recall**: What fraction of the relevant documents are returned?
  - How can you build trivial systems that optimize one of them?
- **Tradeoff**: Increasing precision will usually lower recall, and vice versa
Rankings

- A ranking is an ordering of the documents retrieved that (hopefully) reflects the relevance of the documents to the user query.

- A ranking is based on fundamental premises regarding the notion of relevance, such as:
  - common sets of index terms
  - sharing of weighted terms
  - likelihood of relevance

- Each set of premises leads to a distinct IR model.
Types of IR Models

User Task

Retrieval:
Adhoc Filtering

Structured Models
Non-Overlapping Lists
Proximal Nodes

Browsing

Flat Structure Guided Hypertext

Classic Models
boolean
vector
probabilistic

Set Theoretic
Fuzzy
Extended Boolean

Algebraic
Generalized Vector
Lat. Semantic Index
Neural Networks

Probabilistic
Inference Network
Belief Network
Classic IR models – Basic concepts

- Each document represented by a set of representative keywords or index terms
- An index term is a document word useful for remembering the document's main themes
- Traditionally, index terms were nouns because nouns have meaning by themselves
- Search engines assume that all words are index terms (full text representation)
Classic IR Models – Weights

- Not all terms are equally useful for representing the document contents: less frequent terms allow identifying a narrower set of documents.
- The importance of the index terms is represented by weights associated to them.
- Let
  - \( k_i \) be an index term
  - \( d_j \) be a document
  - \( w_{ij} \) be a weight associated with \( (k_i, d_j) \)
- The weight \( w_{ij} \) quantifies the importance of the index term for describing the document contents.
Classic IR Models – Notation

$k_i$ is an index term (keyword)

$d_j$ is a document

t is the total number of index terms

$K = (k_1, k_2, \ldots, k_t)$ is the set of all index terms

$w_{ij} \geq 0$ is a weight associated with $(k_i, d_j)$

$w_{ij} = 0$ indicates that term does not belong to doc

$g_i(d_j) = w_{ij}$ is a function which returns the weight associated with pair $(k_i, d_j)$

$d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$ is a weighted vector associated with the document $d_j$
Plan for today

- Web services
- Information retrieval
  - Basics
  - Precision and recall
  - Taxonomy of IR models
- Classic IR models
  - Boolean model
  - Vector model
  - TF/IDF
- HITS and PageRank
Boolean model

- Simple model based on set theory
- Queries specified as boolean expressions
  - precise semantics
  - neat formalism
- Terms are either present or absent. Thus,
  \[ w_{ij} \in \{0, 1\} \]
- An example query
  \[ q = k_a \land (k_b \lor \neg k_c) \]
Boolean model for similarity

Query:
\[ q = k_a \land (k_b \lor \neg k_c) \]

In disjunctive normal form:
\[ q = (k_a \land k_b \land k_c) \lor (k_a \land k_b \land \neg k_c) \lor (k_a \land \neg k_b \land \neg k_c) \]

\[ sim(d_j, q) = \begin{cases} 
1 & \text{if } \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \land (\forall k_i : g_i(d_j) = g_i(\vec{q}_{cc})) \\
0 & \text{otherwise}
\end{cases} \]
Drawbacks of boolean model

- Retrieval based on *binary decision criteria* with no notion of partial matching
- **No ranking** of the documents is provided (absence of a grading scale)
- Information need has to be translated into a Boolean expression, which most users find *awkward*
- The Boolean queries formulated by the users are most often too simplistic
- As a consequence, the Boolean model frequently returns either *too few or too many* documents in response to a user query

- We need something more tolerant!
Plan for today

- Web services
- Information retrieval
  - Basics
  - Precision and recall
  - Taxonomy of IR models
- Classic IR models
  - Boolean model
  - Vector model
  - TF/IDF
- HITS and PageRank
Vector model

- A refinement of the boolean model, which does not focus strictly on exact matching
  - Non-binary weights provide consideration for partial matches
  - These term weights are used to compute a degree of similarity between a query and each document

- Ranked set of documents provides for better matching
Vector model

- Define:
  \[ w_{ij} > 0 \text{ whenever } k_i \in d_j \]
  \[ w_{iq} \geq 0 \text{ associated with the pair } (k_i, q) \]
  \[ \overrightarrow{d_j} = (w_{1j}, w_{2j}, \ldots, w_{tj}) \]
  \[ \overrightarrow{q} = (w_{1q}, w_{2q}, \ldots, w_{tq}) \]
- With each term \( k_i \), associate a vector \( \text{vec}(i) \)
- These vectors (e.g., \( \text{vec}(i) \) and \( \text{vec}(j) \)) are assumed to be orthonormal (i.e., index terms are assumed to occur independently within the documents)
  - Does this assumption ("independence assumption") hold in practice?

- The \( t \) vectors \( \text{vec}(i) \) form an orthonormal basis for a \( t \)-dimensional space
- In this space, queries and documents are represented as weight vectors
Bag of words

- In this model, $w_{ij} > 0$ whenever $k_i \in d_j$
  - Exact ordering of terms in the document is ignored
  - This is called the "bag of words" model

- What will be the vectors for the following two documents?
  - "Ape eats banana"
  - "Banana eats ape"

- What needs to be done to fix this?
### Similarity

<table>
<thead>
<tr>
<th>Term</th>
<th>Sense and Sensibility</th>
<th>Pride and Prejudice</th>
<th>Wuthering Heights</th>
</tr>
</thead>
<tbody>
<tr>
<td>affection</td>
<td>115</td>
<td>58</td>
<td>20</td>
</tr>
<tr>
<td>jealous</td>
<td>10</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>gossip</td>
<td>2</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

In the vector model, queries may return documents that are not a 'perfect match'

- Hence, we need a metric for the similarity between different documents, or between a document and a query
- Could we simply subtract the vectors? (L1 norm of distance)
- Could we use a dot product?
- Does normalization help?

From: An Introduction to Information Retrieval, Cambridge UP
Cosine similarity

\[ \text{sim}(d_j, q) = \cos \theta = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \cdot w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \cdot \sqrt{\sum_{j=1}^{t} w_{i,q}^2}} \]

- All weights are nonnegative; hence, \( 0 \leq \text{sim}(q, d_j) \leq 1 \)