CAPTCHAs
And other APIs
CS518
Dr. Justin F. Brunelle
CAPTCHA, reCAPTCHA

- Completely Automated Public Turing test to tell Computers and Humans Apart
- Prevents spam, blocks robots
- Verifies human vs machine
How it works

- Server-delivered challenges
- User provided answers
- Server-side validation
- Transactions via Ajax
Provenance

- Carnegie Mellon Research Project
- Two words:
  - One known (i.e., some threshold of user agreement reached)
  - One unknown (i.e., user agreement not reached)
- Known-term validation creates a “vote” for unknown term
- Known term is the only evaluation criteria
Example – Setup

• See: https://developers.google.com/recaptcha/intro
• wget [library.zip]
• unzip [library.zip]
• Register for a key https://www.google.com/recaptcha/admin
• Follow instructions
Example – Code

<!-- include the library -->

<script src="https://www.google.com/recaptcha/api.js" async defer></script>

<!-- form for the reCAPTCHA →
<form action="?" method="POST">
   <div class="g-recaptcha" data-sitekey="***********"></div>
   <br/>
   <input type="submit" value="Submit">
</form>

http://www.cs.odu.edu/~jbrunelle/cs518/examples/captcha/
General recommendations for using APIs

- Find canonical and authoritative sources/documentation
- Include libraries
- Follow instructions for hello world
- Adapt
API Information

- Software-to-software actions
- Formal definitions, interactions
- HTTP for data
API Design Guidelines

- Use REST
- Scope data as necessary
  - Large amounts only when expected
  - Use pagination when available
- Use standard (and documented) formats
Keys

- Tokens for access, prevent abuse, track usage, etc.
- Sending token is like sending a session ID
  - Useful for authentication
  - Revoking for expired access
- Public-Private pairs for signing
JSON

• JavaScript Object Notation
• Emerging (emerged?) standard format
• Like XML but less verbose
• Major languages support encoding:
  – PHP's json_decode, json_encode, json_last_error
"opponent": "Hampton",
"score": "19",
"teamInfo": {
    "mascot": "pirate",
    "nickname": "pirates",
    "division": "FCS"
},
"roster": [
    {
        "name": "Justin",
        "position": "bench warmer",
        "number": "00"
    },
    {
        "name": "Johnny",
        "position": "QB",
        "number": "1"
    }
]
GitHub APIs

• Documentation:

• curl -iL
  "https://github.com/login/oauth/authorize?client_id=XXXXX6&scope=repo&state=jbrunelle"
GitHub API: Access

• Register your app (website) at github.com
  - Gives you client id and client secret
• Authorize a user with your client_id, scope, and redirect URI
• Retrieve access token as form of authentication
Gravatar

- Avatar sharing
- No authentication
- Img src:
  https://www.gravatar.com/avatar/" . md5( strtolower( trim( $email ) ) ) . "?&s=" . $size;
Searching

- Relational data base looks for specific string matching
  
  SELECT Date
  FROM opponents
  WHERE name = "Hampton"
  AND score = "19";
Versus...

Precision vs Recall

Set of relevant items in database

Set of items retrieved
Precision vs Recall

- Relevant items - retrieved
- Relevant items - not retrieved
- Irrelevant items - retrieved
How much extra stuff did you get?

C: # of irrelevant records retrieved
A: # of relevant records retrieved

Precision: \( \frac{A}{A + C} \times 100\% \)
How much did you miss?

B: # of relevant records not retrieved
A: # of relevant records retrieved

Recall: \( \frac{A}{A + B} \times 100\% \)
Example

• 10 documents in index are relevant
• Search returns 20 documents, 5 are relevant
• Precision:
  – $P = 5/(5+15)=0.25$
  – 1 out of 4 retrieved documents are relevant
• Recall:
  – $R = 5/(5+5)=0.5$
  – Half of the relevant documents were retrieved
Another way to look at it...

- **Relevant elements**
  - False negatives
  - True negatives
  - True positives
  - False positives

- **Selected elements**

**Precision** = \[rac{\text{True positives}}{\text{True positives} + \text{False positives}}\]

**Recall** = \[rac{\text{True positives}}{\text{True positives} + \text{False negatives}}\]
Fuzzier Matching

- LIKE and REGEXP operator in MySQL
- REGEXP
  - Pattern match succeeds if the pattern matches anywhere in the value being tested.
- LIKE
  - Pattern match succeeds only if the pattern matches the entire value
### Example

<table>
<thead>
<tr>
<th>username</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>pallen</td>
<td>m$ftw</td>
</tr>
<tr>
<td>dknuth</td>
<td>tek!tex</td>
</tr>
<tr>
<td>ada</td>
<td>wtf15b4b</td>
</tr>
<tr>
<td>cmoore</td>
<td>moreM00R3!</td>
</tr>
<tr>
<td>jresig</td>
<td>lnOJS</td>
</tr>
<tr>
<td>atanen</td>
<td>minix!minix</td>
</tr>
<tr>
<td>linus</td>
<td>lUvP3nGu1n5</td>
</tr>
<tr>
<td>aturing</td>
<td>1nfin1t3TAp3</td>
</tr>
<tr>
<td>lwall</td>
<td>oysters&amp;camels</td>
</tr>
<tr>
<td>thewoz</td>
<td>4daK1d5</td>
</tr>
</tbody>
</table>

**SELECT** * FROM USERS WHERE username LIKE 'a%'

<table>
<thead>
<tr>
<th>username</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>ada</td>
<td>wtf15b4b</td>
</tr>
<tr>
<td>atanen</td>
<td>minix!minix</td>
</tr>
<tr>
<td>aturing</td>
<td>1nfin1t3TAp3</td>
</tr>
</tbody>
</table>

**SELECT** * FROM USERS WHERE password REGEXP '^[0-9]{2,}$'

<table>
<thead>
<tr>
<th>username</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>ada</td>
<td>wtf15b4b</td>
</tr>
<tr>
<td>cmoore</td>
<td>moreM00R3!</td>
</tr>
</tbody>
</table>
But it gets better...

- MATCH()...AGAINST()
  - performs a natural language search over index
- Index = set of one or more columns of the same table
  - column must have type FULLTEXT
- MATCH()
  - takes a comma-separated list that names the columns to be searched
- AGAINST()
  - takes a string to search for
- If used in WHERE clause, results returned in order of relevance score
  - relevance: similarity between search string and index row
FULLTEXT

- Can only create FULLTEXT on CHAR, VARCHAR or TEXT columns
- "title" and "body" still available as regular columns
- If you want to search only on "title", you need to create a separate index

CREATE TABLE odu_football ( id INT AUTO_INCREMENT NOT NULL PRIMARY KEY, opponent VARCHAR(200), notes TEXT, date DATE, FULLTEXT (opponent, notes))
Stopwords

- Common words that do not make good search terms
- If a word appears in more than half the rows, it's a stop word
- Stopwords vary on collections/dictionaries/corporuses