Building Babel

Large Scale Data Collection in the Cloud

Ian Wesley-Smith
iwsmith@uw.edu

DataLab

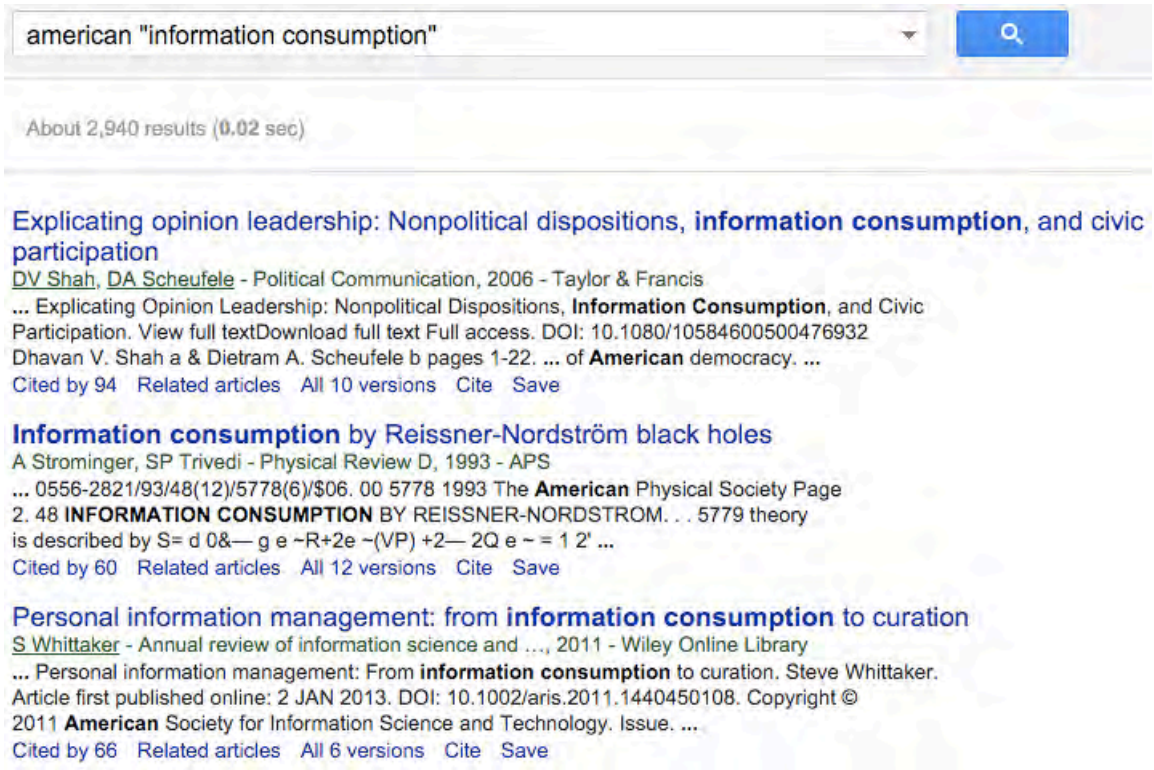
Information School
UNIVERSITY of WASHINGTON

Scholarly Article Recommendation

- Information Overload
  - 50m – 150m articles in existence

Google Scholar

- Recommendation vs Search
  - Serendipity
- Homonymity
- Synonymity



Netflix/Spotify/Amazon

- User ratings (explicit, implicit)
- Density
  - # user-item interactions >> # items
- Netflix Competition (2006)[1]
  - 100m ratings
  - 480k users
  - 17k movies

1: http://www.netflixprize.com/community/viewtopic.php?id=68

Barriers to Research

- Hard to get datasets
- Difficult to measure effectiveness
  - Judges
  - Citation prediction

Enter Babel

- Provide access to private data sets
- Provide scholarly article recommendations, freely to anyone
  - Feedback data in return
- Evaluate recommenders using usage data
  - With enough traffic could be very fast

Audience

- Publishers
  - Offload expensive research into recommender systems to academia
  - Better recommendations drive more traffic/purchases
- Tool Developers
- Researchers

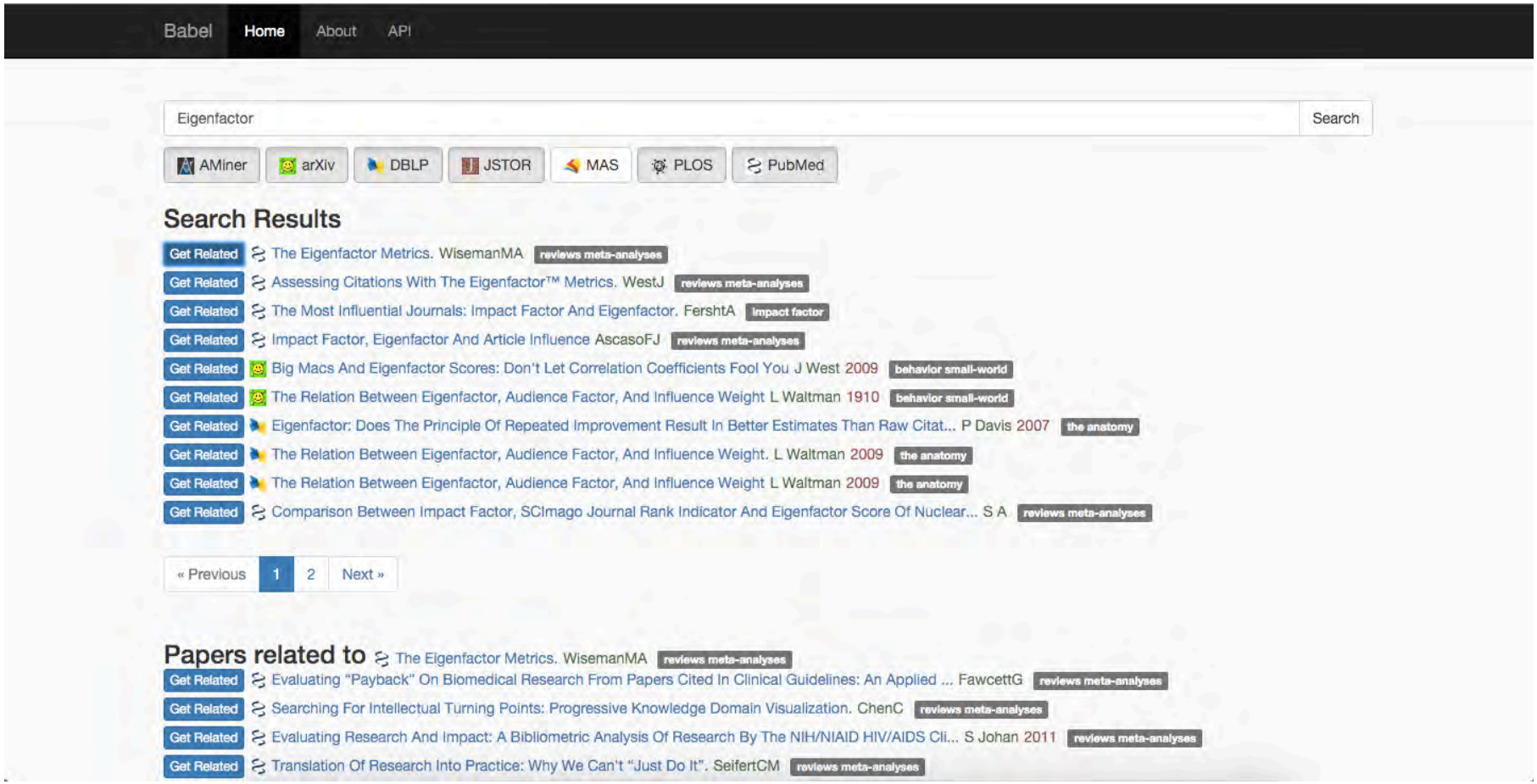
Requirements

- Fast
- Reliable
- Scalable (lots of data!)
- Easy to use
- Cheap

REST API

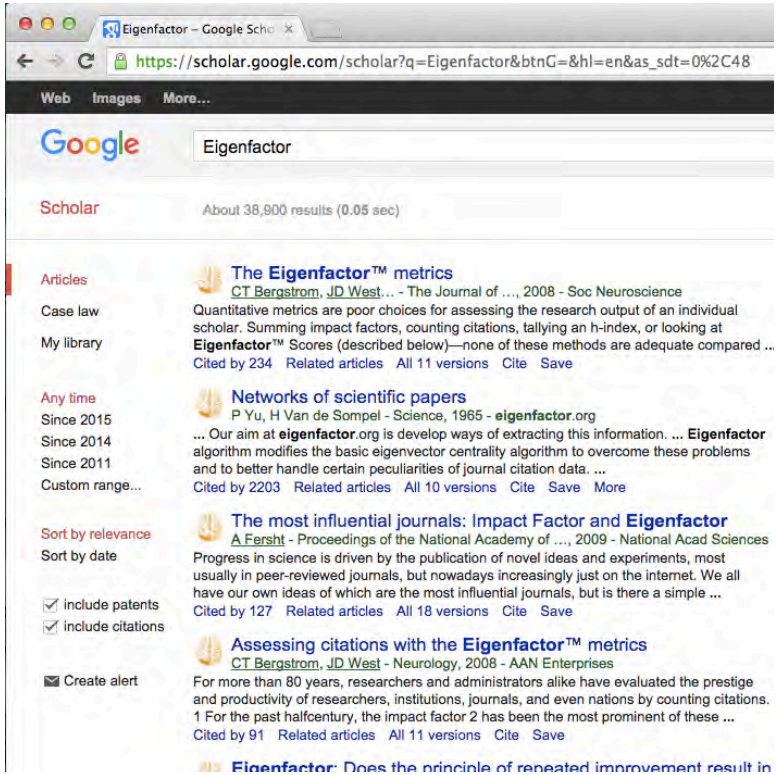
```
curl http://babel-us-east-
1.eigenfactor.org/recommendation/aminer/12345
{
  "transaction_id": "46bb84190e9ddfd17700bfafb500ab3c",
  "results": [
    {
      "paper_id": "672",
      "publisher": "aminer"
    },
    {
      "paper_id": "11274",
      "publisher": "aminer"
    }
  ]
}
```

http://babel.eigenfactor.org

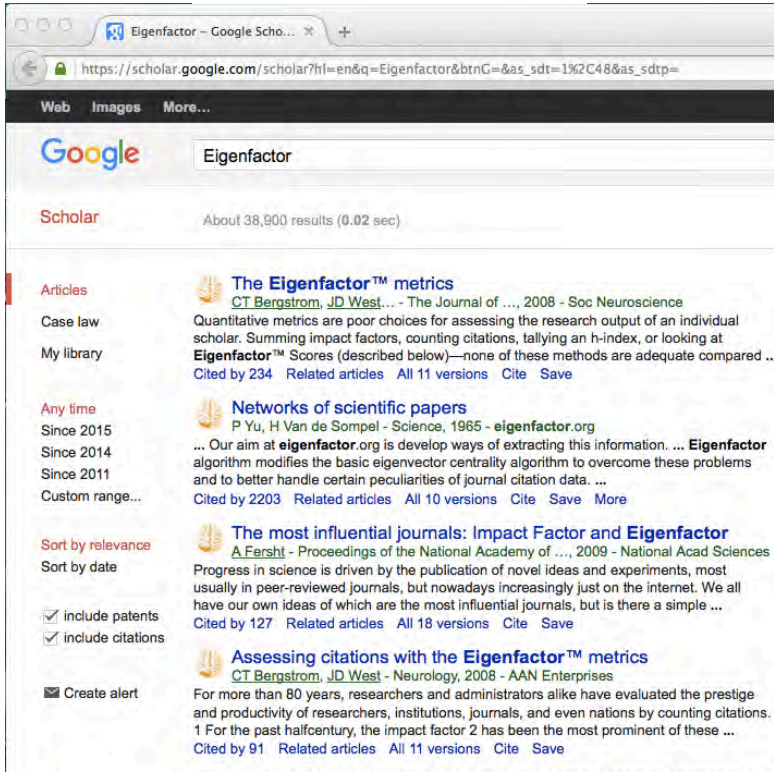


Browser Plugins

Firefox

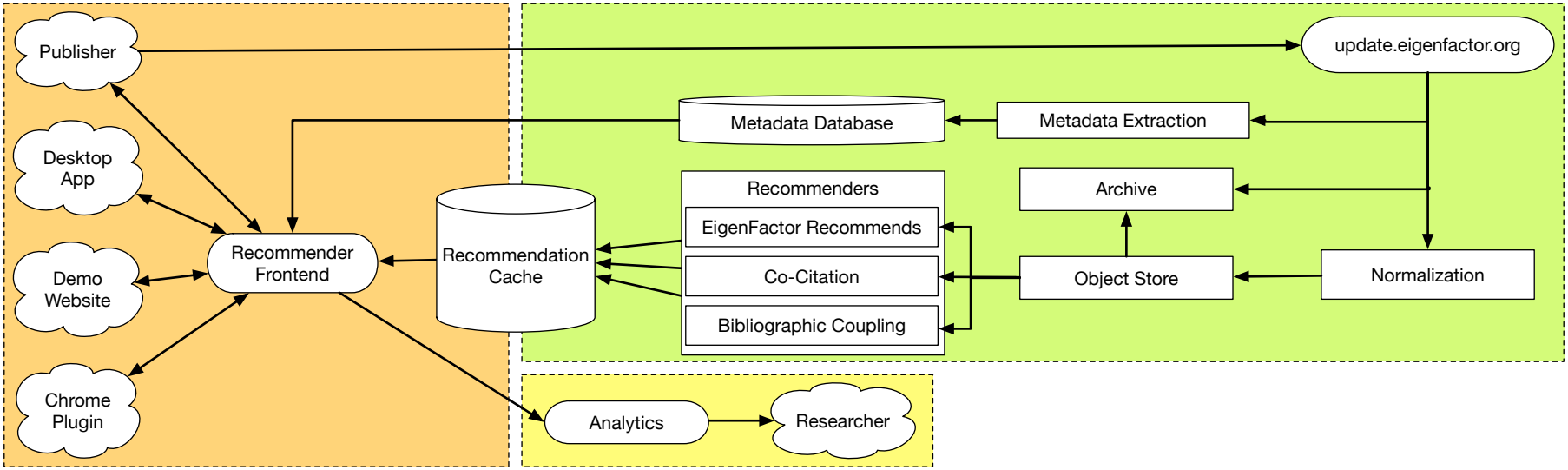


chrome

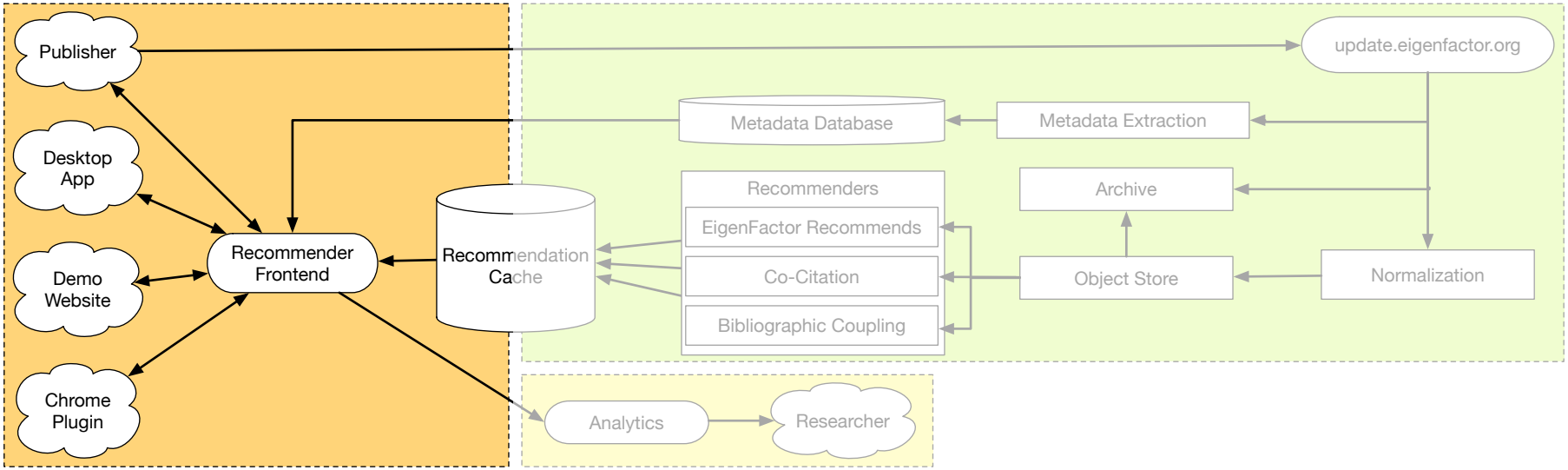


http://labs.jstor.org/sustainability/

Babel Architecture



Frontend

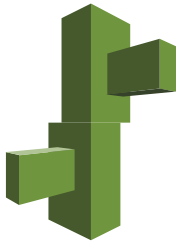

Publisher
Desktop App
Demo Website
Chrome Plugin
Recommender Frontend
Recommendation Cache
update.eigenfactor.org
Metadata Database
Metadata Extraction
Recommenders
EigenFactor Recommends
Co-Citation
Bibliographic Coupling
Archive
Object Store
Normalization
Analytics
Researcher

Frontend

Application

Package

Deploy



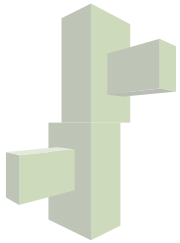
AWS Elastic Bean Stalk

Frontend

Application

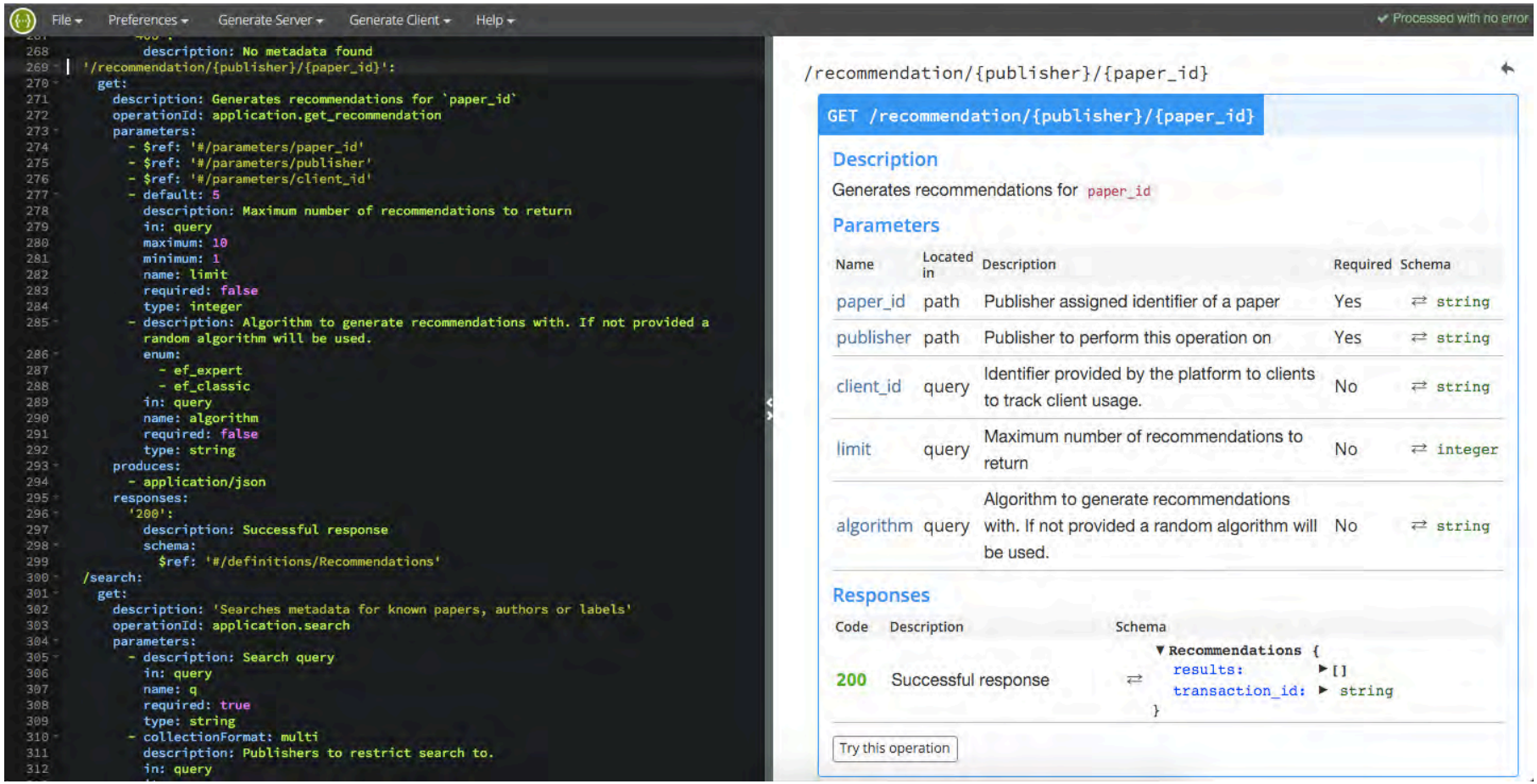
Package

Deploy

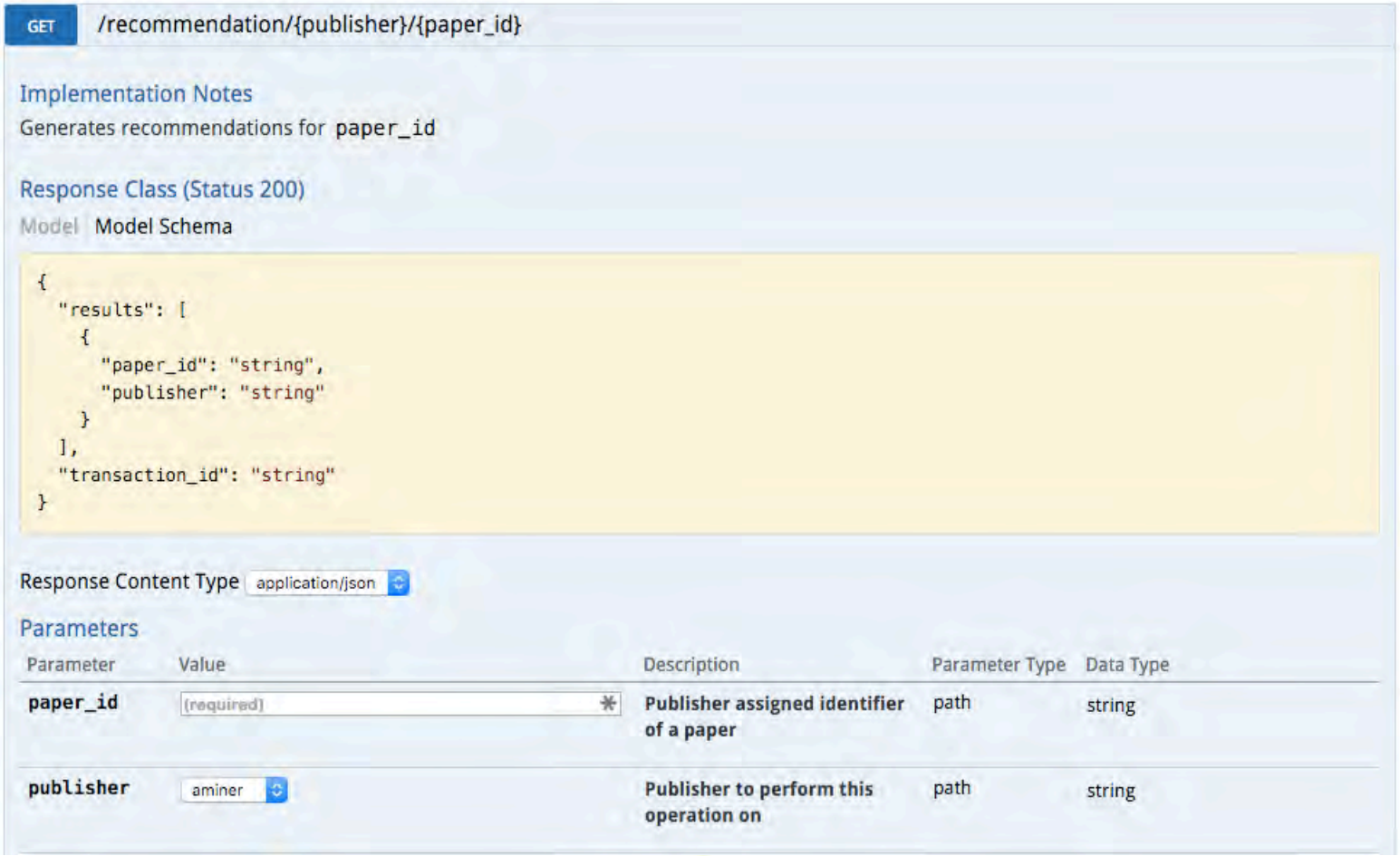


AWS Elastic Bean Stalk

swagger


File
Preferences
Generate Server
Generate Client
Help
Processed with no error
description: No metadata found
'/recommendation/{publisher}/{paper_id}':
get:
description: Generates recommendations for `paper_id`
operationId: application.get_recommendation
parameters:
- $ref: '#/parameters/paper_id'
- $ref: '#/parameters/publisher'
- $ref: '#/parameters/client_id'
- default: 5
description: Maximum number of recommendations to return
in: query
maximum: 10
minimum: 1
name: limit
required: false
type: integer
- description: Algorithm to generate recommendations with. If not provided a random algorithm will be used.
enum:
- ef_expert
- ef_classic
in: query
name: algorithm
required: false
type: string
produces:
- application/json
responses:
'200':
description: Successful response
schema:
$ref: '#/definitions/Recommendations'
/search:
get:
description: 'Searches metadata for known papers, authors or labels'
operationId: application.search
parameters:
- description: Search query
in: query
name: q
required: true
type: string
- collectionFormat: multi
description: Publishers to restrict search to.
in: query
/recommendation/{publisher}/{paper_id}
GET /recommendation/{publisher}/{paper_id}
Description
Generates recommendations for paper_id
Parameters
Name
Located in
Description
Required
Schema
paper_id
path
Publisher assigned identifier of a paper
Yes
string
publisher
path
Publisher to perform this operation on
Yes
string
client_id
query
Identifier provided by the platform to clients to track client usage.
No
string
limit
query
Maximum number of recommendations to return
No
integer
algorithm
query
Algorithm to generate recommendations with. If not provided a random algorithm will be used.
No
string
Responses
Code
Description
Schema
200
Successful response
Recommendations {
results: []
transaction_id: string
}
Try this operation

Swagger UI


GET
/recommendation/{publisher}/{paper_id}
Implementation Notes
Generates recommendations for paper_id
Response Class (Status 200)
Model
Model Schema
{
"results": [
{
"paper_id": "string",
"publisher": "string"
}
],
"transaction_id": "string"
}
Response Content Type
application/json
Parameters
Parameter
Value
Description
Parameter Type
Data Type
paper_id
(required)
Publisher assigned identifier of a paper
path
string
publisher
aminer
Publisher to perform this operation on
path
string

Swagger UI

Curl

```
curl -X GET --header "Accept: application/json" "http://babel-us-east-1.eigenfactor.org/recommendation/aminer/12345?lim
```

Request URL

```
http://babel-us-east-1.eigenfactor.org/recommendation/aminer/12345?limit=5
```

Response Body

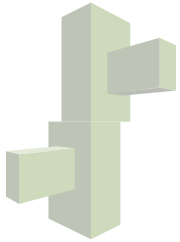

```
{
  "transaction_id": "d403eb04898d9db782f30b14c983bec6",
  "results": [
    {
      "paper_id": "235114",
      "publisher": "aminer"
    },
    {
      "paper_id": "24108",
      "publisher": "aminer"
    },
    {
      "paper_id": "121114",
      "publisher": "aminer"
    },
    {
      "paper_id": "1058620",
      "publisher": "aminer"
    },
    {
```


Frontend

Application

Package

Deploy



AWS Elastic Bean Stalk

20 lines (12 sloc) 462 Bytes Raw Blame History

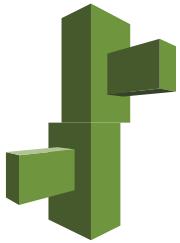
```
1  FROM       python:3-onbuild
2
3  WORKDIR    /var/app
4
5  RUN        pip3 install virtualenv
6  RUN        virtualenv /var/app
7
8  RUN        useradd uwsgi -s /bin/false
9  RUN        mkdir /var/log/uwsgi
10 RUN        chown -R uwsgi:uwsgi /var/log/uwsgi
11
12 ADD        . /var/app
13 RUN        if [ -f /var/app/requirements.txt ]; then /var/app/bin/pip install -r /var/app/requirements.txt; fi
14
15 ENV        BABEL_STAGE beta
16
17 EXPOSE     8080
18
19 CMD        ["python", "./src/application.py"]
```

Frontend

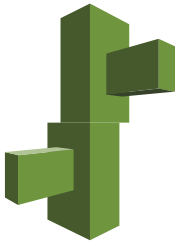
Application

Package

Deploy



AWS Elastic Bean Stalk



AWS Elastic Bean Stalk

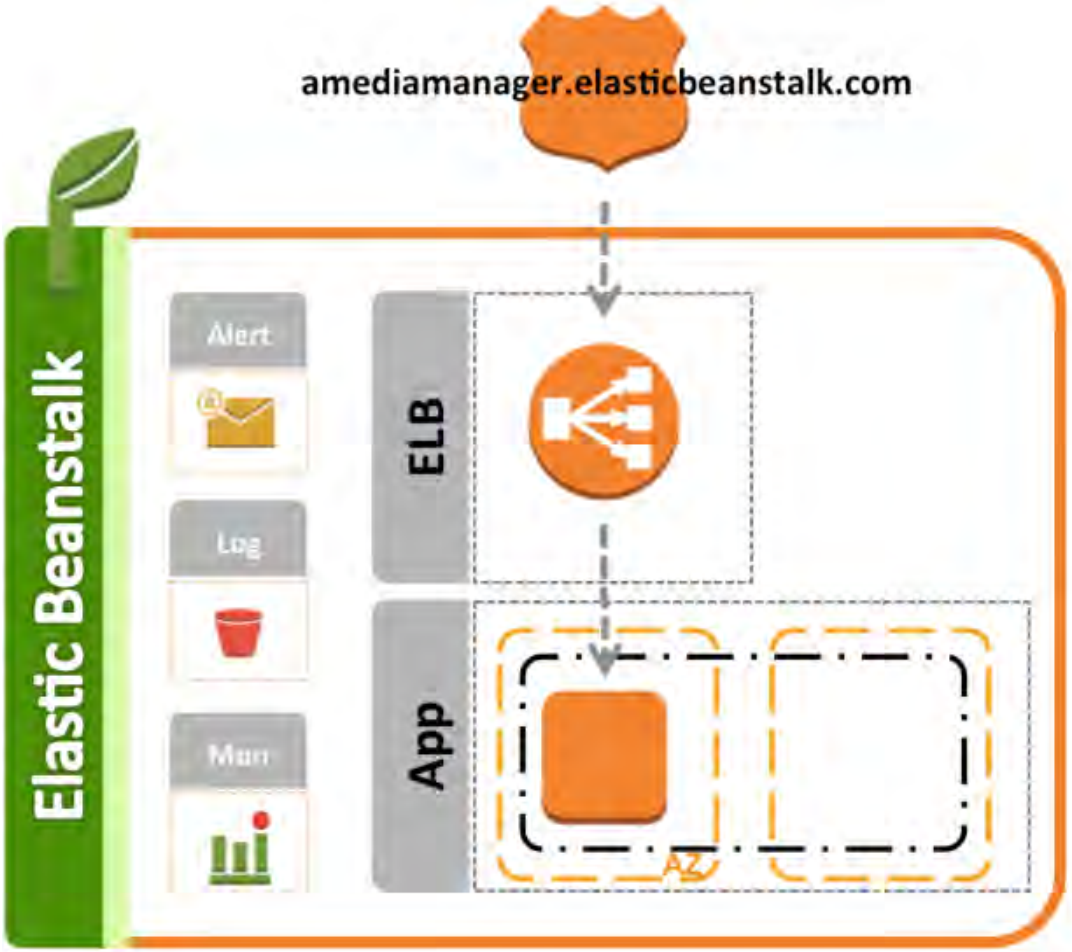
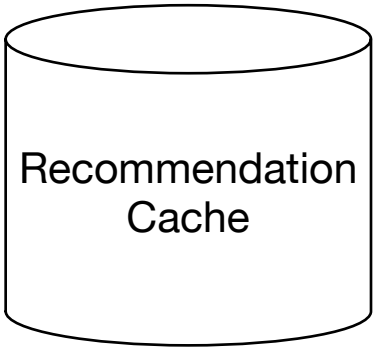


Image: [Part 1: Develop, Deploy, and Manage for Scale with Elastic Beanstalk and CloudFormation Series] by Evan Brown, AWS

DynamoDB

- AWS NoSQL
  - Key-value store
- Very fast (<10ms)
- Very scalable
  - Specify throughput
- Not too expensive



Issues

- Not all AWS services are created equal
  - Data Pipeline
  - Cloud Search
- Documentation
- SDK/Tooling
- Python & GIL
- Access Keys

Future Directions

- Finish backend
- Expand clients (publishers, tool developers)
- Actually get more recommenders
- Babel 3.0 – simple middleware
  - Automatically logs & add transaction info to outgoing requests

http://babel.eigenfactor.org

iwsmith@uw.edu