Harvard University Extension School
"Principles of Big Data Processing"
CSCI E-88, Spring 2022
Final Project
by Goldberg, Dan

Project Goal and Problem Statement

The ongoing efforts to develop a generally capable artificial intelligence require a great deal of collaboration between multiple disciplines, and sub-disciplines, not least among them psychology and computer science. Increased understanding of psychological mechanisms increasingly drive new powerful developments in machine learning.

Knowing which psychological research might be relevant however requires a great deal of subject matter expertise. If you for example were looking for inspiration on making an algorithm more creative, you might not be familiar with some of the underlying theories and studies on neurological mechanisms of creativity.

The goal of this project is to explore methods for making these searches more user friendly for non experts. I believe that even if an engineer with no background in psychology struggles to understand the research, knowing which questions and who to ask are often the first steps towards collaboration and discovery.

I will begin this exploration by indexing research papers which have used a factor based personality metric alongside other measures. These factor based systems are fairly ubiquitous as they are cheap and easy to administer, and were designed to inventory behaviors precisely so their underlying mechanisms could be better investigated.

The factor based systems were distilled from large sets of adjectives into subsets which are thought to represent the overall category. For example the NEOPI system uses one general factor of “Openness to Experience”, with sub-factors called “facets” of Fantasy, Aesthetics, Feelings, Actions, Ideas, and Values. Those sub-facets were arrived at by factoring the aforementioned large sets of adjectives people commonly use to describe themselves and others. I suspect this will help to make literature searches more natural for a non psychology expert.

YouTube Video URL

https://youtu.be/bqduIdYDmbQ

Big Data Source

I downloaded over 50gb of article data from Springer Nature open access API which: **“**Provides metadata for 14 million online documents (e.g., journal articles, book chapters, protocols).”, and wrote a custom algorithm in nodejs to parse the separate jsonl files so they could be mapped and indexed for mongo and elastic

Expected results

I created an end to end MERN stack with search which leverages the big-five factors to expand relevant psychology searches

Processing Pipeline

Implementation Details

First I acquired a nearly 50gb dataset from Springer Nature with metadata, descriptions, authors for scientific research papers, going back to 1894!
https://dev.springernature.com/

I then created a local MERN stack: Mongo, React, Elastic Search, Node.js discussed below

Mongo:

1. Downloaded and installed mongo community https://www.mongodb.com/try/download/community
2. Configured my local database as a replica (basically a shard)
    a. This is necessary in order to have it trigger code needed to update Elastic
        i. However this didn’t work and cost me days. The version support is very old. I ended up needing to do all the mapping manually after trying everything else. I will cover this below
    b. It is also necessary for deployment though so:
        i. https://www.mongodb.com/docs/manual/reference/replica-configuration/
3. Confirmed functionality using basic commands in console. See installation instructions above

Node.js:

1. Installed Node.js

    a. https://nodejs.org/en/download/
2. Updated NPM (node package manager)
    a. https://www.npmjs.com/package/npm-windows-upgrade
3. Installed Express.js from NPM
    a. https://expressjs.com/
4. Added Mongoose to interface with Mongo through express.js
    a. https://mongoosejs.com/docs/
5. Set up a basic mongoose/mongo schema to handle articles
6. Set up basic routing to handle get/post requests

React.js

1. Downloaded create react app from npm
    a. https://create-react-app.dev/
    b. It essentially lets you generate a barebones react framework with some options
        i. I chose to add a simple router, some css bootstrapping, axios for ajax calls
2. Added a few views for testing basic article views
3. Tested those views with dummy data on the front end
4. Tested those views with dummy data on the backend
5. Wrote and tested a push to the backend
    a. Yelled “why aren’t you working!?!?!” 50 times… iterated…

Once I had a complete round trip working from react, to node.js, to mongo, I added elastic search. This is where things somewhat fell apart. More on that later

Elastic and Kibana

1. https://www.elastic.co/guide/en/elasticsearch/reference/current/zip-windows.html

We went over Elastic and Kibana just a few days ago so I’d like to tell you where I ran into problems instead of going through the blow by blow.

1. You can’t send mongo keys to elastic, so I needed to figure out how to match those up. I ended up using the doi for research papers as that is a unique key
2. Elastic, even though installed completely locally, wanted a lot of authentication with every call, and this took me hours to slog through. Finally I disabled it entirely, which also took awhile. I woudl recommend for local development: switching off security and making a ticket for later
3. I tried four different methods of getting data into Elastic from Mongo
    a. Mongooser : a javascript library which no longer works
    b. mongo-connector: supposed to connect and auto-update between elastic and mongo… good luck
    c. Manually streaminig the data from mongo
    d. Manually querying mongo and updating elastic step by step

e. Finally I deleted both, reinstalled (to be sure) and wrote a low level text parser to parse the files, map them to objects, and send updates to both mongo and elastic. With new records as I am the single source of truth this should be fine.

Once I had data in both, matched on DOI I then tracked down the five-factor model lists of adjectives. I wrote a map which extracts words from search terms on the front end, strips out all the common connector words like “the” and “of”, matched against the nltk set I pulled from a python library: https://pythonspot.com/nltk-stop-words/ . The resulting list is then fed into an elastic search fuzzy query.

At time of writing I’m simply returning those results, but it should use the matched DOI foreign key shortly. Those results are displayed in react!

Implementation

My Node and React repository : https://github.com/DanG-Mergin/facet-search

React Search Page:

```
import React, { Component } from 'react';
import '../App.css';
import axios from 'axios';
import { Link } from 'react-router-dom';
import ArticleCard from './ArticleCard';

class ShowArticleList extends Component {
    constructor(props) {
        super(props);
        this.state = {
            searchText: '',
            articles: []
        };
    }

    onChange = e => {
        this.setState({ [e.target.name]: e.target.value });
    };

    onSearchClick = e => {
        // e.preventDefault();
```

```
        debugger;
        const data = {
            searchText: this.state.searchText
        };
        console.log(this.state)

        axios
            .get(
                'http://localhost:8082/api/articles/search/', {
                params: {
                    data
                }
            })
            .then(res => {
                // update parent here
                debugger;
                this.setState({
                    // searchText: ''
                    articles: res.data.articles
                })
                // I think this may be tied to form submission and can be
removed
                this.props.history.push('/');
            })
            .catch(err => {
                console.log("Error in Search!");
            })

    }

    render() {
        const articles = this.state.articles;
        console.log("PrintArticle: " + articles);
        let articleList;

        if (!articles) {
            articleList = "there is no article recored!";
        } else {
            articleList = articles.map((article, k) =>
                <ArticleCard article={article} key={k} />
```

```
            );
        }

        return (

            <div className="ShowArticleList">
                <div className="container">
                    <div className="searchBar form-group">
                        <div className="row">
                            <label class="form-control">
                                <input type="checkbox"
name="checkbox-checked" checked />
                                More Creative
                            </label>
                            <label class="form-control">
                                <input type="checkbox"
name="checkbox-checked" checked />
                                More Conservative
                            </label>
                        </div>
                        <div className='row'>
                            <input
                                type='text'
                                placeholder='How would you like to improve
your robot?'
                                name='searchText'
                                className='form-control col'
                                defaultValue={this.state.searchText}
                                onChange={this.onChange}
                            />
                            <div class="search-icon-svg"
onClick={this.onSearchClick.bind(this)}></div>
                        </div>

                    </div>
                </div>
                <div className="container">
                    <div className="row">
                        <div className="col-md-12">
                            <br />
```

```
                            <h2 className="display-4 text-center"></h2>
                        </div>
                    </div>

                    <div className="list">
                        {articleList}
                    </div>
                </div>
            </div>
        );
    }
}

export default ShowArticleList;
```

React article card:

```
import React from 'react';
import { Link } from 'react-router-dom';
import '../App.css';

const ArticleCard = (props) => {
    const article = props.article;

    return (
        <div className="card-container mycard">
            <div className="desc">
                <h2>
                    <Link to={`/show-article/${article._id}`}>
                        {article.id}
                    </Link>
                </h2>

                <h3>{article.author}</h3>
                <p>{article.description}</p>
                <h3>Key Words</h3>
                <p>article.keywords</p>
            </div>
        </div>
    )
};
```

```
export default ArticleCard;
```

Mongo connection

```
const mongoose = require('mongoose');
const config = require('config');
// const db = config.get('mongoURI');
const db = config.get('localMongoURI')

const connectDB = async () => {
    try {
        await mongoose.connect(
            db,
            {
                useNewUrlParser: true
            }
        );

        console.log('MongoDB is Connected...');
    } catch (err) {
        console.error(err.message);
        process.exit(1);
    }
};

module.exports = connectDB;
```

Mongo model:

```
// models/ScholarlyArticle.js

const mongoose = require('mongoose');

const ScholarlyArticleSchema = new mongoose.Schema({
    id: {
        type: String
    },
    title: {
        type: String,

    },
```

```
    isbn: {
        type: String,

    },
    author: {
        type: String,

    },
    description: {
        type: String
    },
    keywords: {
        type: Array
    },
    datePublishedReg: {
        type: Date
    }
}, { collection: 'scholarlyarticles' });

module.exports = ScholarlyArticle = mongoose.model('ScholarlyArticle',
ScholarlyArticleSchema);
```

Node.js startup:

```
const express = require('express');
const connectDB = require('./config/db');
const elasticClient = require('./config/elasticClient')
const cors = require('cors');

// routes
const articles = require('./routes/api/articles');
const books = require('./routes/api/books');

const app = express();

// Connect Database
connectDB();
// const BuildIndexes = require('./models/BuildIndexes');
// BuildIndexes();
// cors
```

```
app.use(cors({ origin: true, credentials: true }));

// Init Middleware
app.use(express.json({ extended: false }));

app.get('/', (req, res) => res.send('Hello world!'));

// use Routes
app.use('/api/articles', articles);

const port = process.env.PORT || 8082;

app.listen(port, () => console.log(`Server running on port ${port}`));
```

Parsing the jsonl files, adding to mongo and elastic:

```
const mongoose = require('mongoose');
const elasticClient = require('../config/elasticClient');
const ScholarlyArticle = require('./ScholarlyArticle');

const { once } = require('events');
const { createReadStream } = require('fs');
const { createInterface } = require('readline');



async function BuildIndexes() {
    let articles = async () => {
        for (var i = 886; i <= 886; i++) {
            try {
                const j = `articles_${i}.jsonl`;
                const rl = createInterface({
                    input: createReadStream(`./articles_${i}.jsonl`),
                    crlfDelay: Infinity
                });

                rl.on('line', (line) => {
                    let {
                        id,
                        title,
                        isbn,
```

```
                description,
                keywords,
                datePublishedReg
            } = JSON.parse(line)
            const art = {
                id,
                title,
                isbn,
                description,
                keywords,
                datePublishedReg
            }
            ScholarlyArticle.create(art)
                .then(function (article) {
                    console.log(article);
                    let { id,
                        title,
                        isbn,
                        description,
                        keywords,
                        datePublishedReg } = article;
                    elasticClient.index({
                        index: 'scholarlyarticles',
                        body: {
                            id,
                            title,
                            isbn,
                            description,
                            keywords,
                            datePublishedReg
                        }
                    })
                })
                .then(article => {

                    console.log({ article })
                })
                .catch(err => console.log(err));
        });
```

```
                await once(rl, 'close');

                console.log('File processed.');
            } catch (err) {
                console.error(err);
            }
        }
    }
    await articles();
    await ScholarlyArticle.find()
        .then(articles => console.log(articles))
        .catch(err => console.log(err));
}

module.exports = BuildIndexes;
```

Elastic Queries:

```
PUT /scholarlyarticles
{
  "mappings": {
    "properties": {
      "id": {
        "type": "text"
      },
      "description": {
        "type": "text"
      },
      "keywords": {
        "type": "text"
      },
      "datePublishedReg": {
        "type": "date"
      }
    }
  }
}
```

Retrieving documents in node:

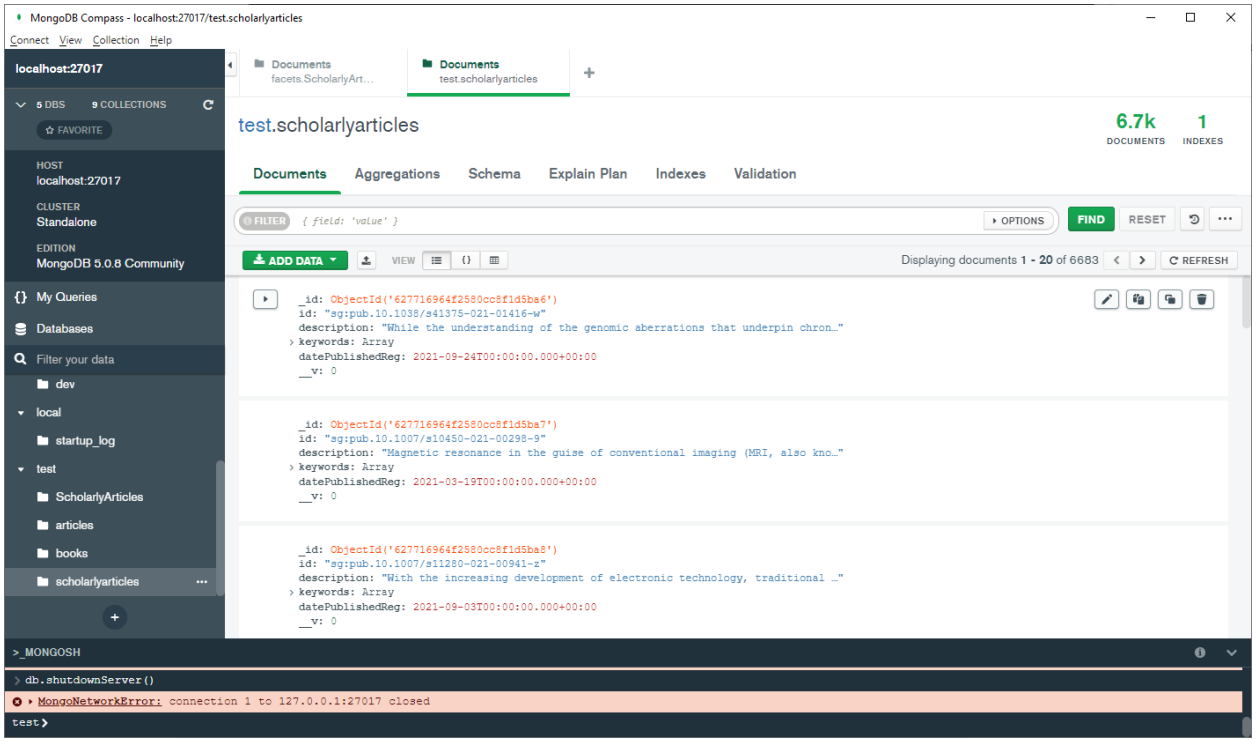
```
// @route GET api/search
```

```
// @description Get articles by key terms
// @access Public
router.get('/search', (req, res) => {
    console.log(req.query.data)
    // TODO: all this parsing should be in another module
    function parseText(text) {
        text = text.toLowerCase();
        console.log(Object.entries(thesaurus));
        // let matchedCategories = {}
        // remove duplicates with Set
        let words = new Set(text.split(/\s+/));
        let noStopWords = [...words].filter(w => !nltkStopWords.has(w))
            .reduce((acc, w) => {
                for (const k in thesaurus) {
                    console.log(thesaurus[k])
                    if (thesaurus[k].has(w)) {
                        acc['keys'] = new Set([...acc['keys'],
...thesaurus[k]])
                    }
                }
                return acc;
            }, { keys: new Set() });
        return noStopWords;
    }
    // discuss the usefulness of semantic search using euclidian distance
    // perhaps this could be a comparison vs human guided in this domain


    if (req.query.data) {
        // TODO: this is a security issue
        const parsed = JSON.parse(req.query.data);
        if (parsed.searchText) {
            const matches = parseText(parsed.searchText)

            if (matches.keys.size) {
                console.log(`found matches ${matches.keys}`);
                console.log(`object ${Object.entries(matches)}`);
                const q = [...matches.keys].join(' ');

                elasticClient.search({
```

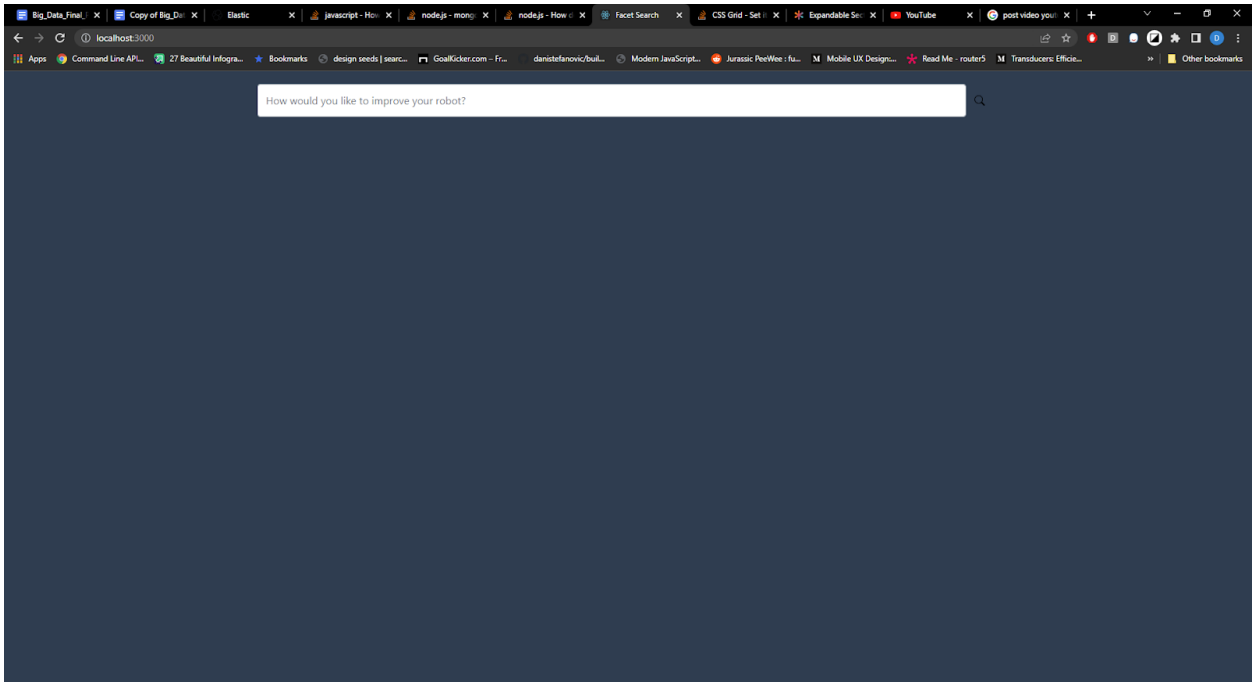
```
                    index: 'scholarlyarticles',
                    body: {
                        query: {
                            match: {
                                description: q
                            }
                        }
                    }
                }).then(response => {
                    console.log(response)
                    if (response['statusCode'] && response['statusCode']
== 200) {
                        let hits = response.body.hits.hits;
                        // TODO: needs to get response from mongoose.  No
time atm
                        obs = hits.map(x => {
                            return x['_source'];
                        })
                        res.json({ articles: obs })

                    }
                })
            }
        }
    }
});
```

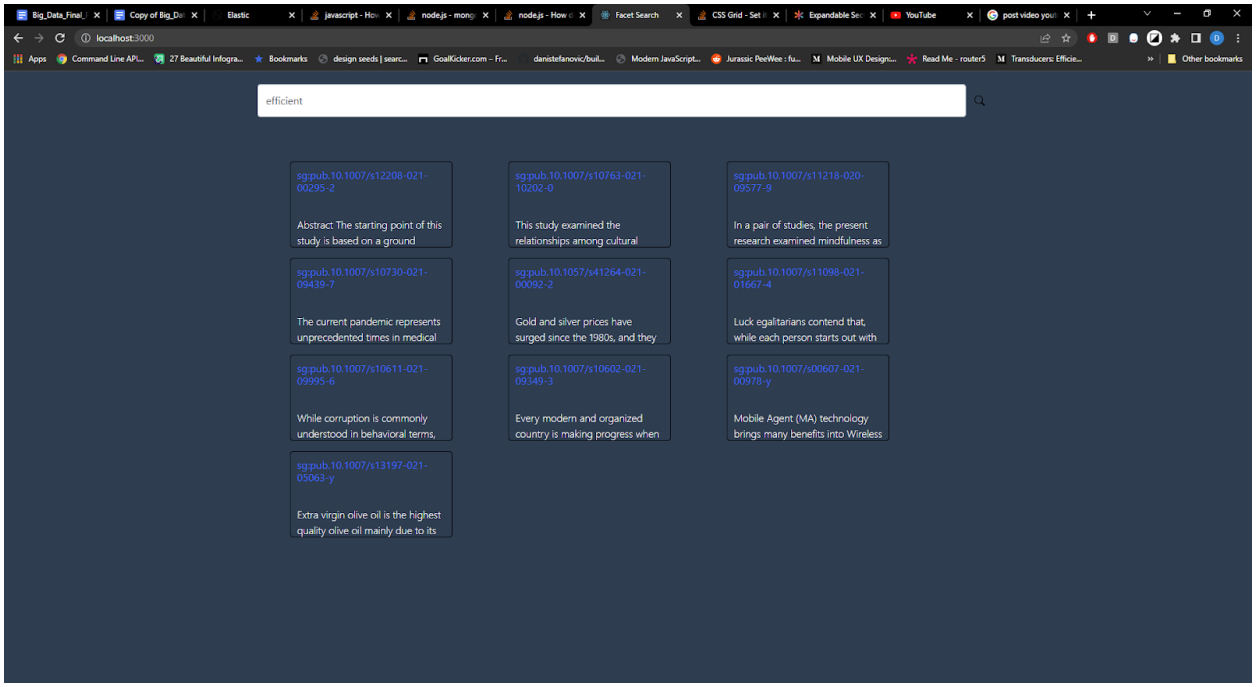
Results

Demo set of mongo articles


MongoDB Compass - localhost:27017/test.scholarlyarticles
Connect View Collection Help
localhost:27017
5 DBS 9 COLLECTIONS
FAVORITE
HOST
localhost:27017
CLUSTER
Standalone
EDITION
MongoDB 5.0.8 Community
My Queries
Databases
Filter your data
dev
local
startup_log
test
ScholarlyArticles
articles
books
scholarlyarticles
Documents
facets.ScholarlyArt…
Documents
test.scholarlyarticles
test.scholarlyarticles
6.7k
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
{ field: 'value' }
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 1 - 20 of 6683
REFRESH
_id: ObjectId('627716964f2580cc8f1d5ba6')
id: "sg:pub.10.1038/s41375-021-01416-w"
description: "While the understanding of the genomic aberrations that underpin chron…"
keywords: Array
datePublishedReg: 2021-09-24T00:00:00.000+00:00
__v: 0
_id: ObjectId('627716964f2580cc8f1d5ba7')
id: "sg:pub.10.1007/s10450-021-00298-9"
description: "Magnetic resonance in the guise of conventional imaging (MRI, also kno…"
keywords: Array
datePublishedReg: 2021-03-19T00:00:00.000+00:00
__v: 0
_id: ObjectId('627716964f2580cc8f1d5ba8')
id: "sg:pub.10.1007/s11280-021-00941-z"
description: "With the increasing development of electronic technology, traditional …"
keywords: Array
datePublishedReg: 2021-09-03T00:00:00.000+00:00
__v: 0
>_MONGOSH
db.shutdownServer()
MongoNetworkError: connection 1 to 127.0.0.1:27017 closed
test>

App





Conclusions and Lesson Learned

I would do everything from scratch. Plugins and connectors are a lie.

Locally the elastic security was extremely problematic. For testing I recommend shutting those off until you’re ready to deploy.