Enterprise Search

Lama Saeeda

lama.saeeda@fel.cvut.cz

Winter Term 2019

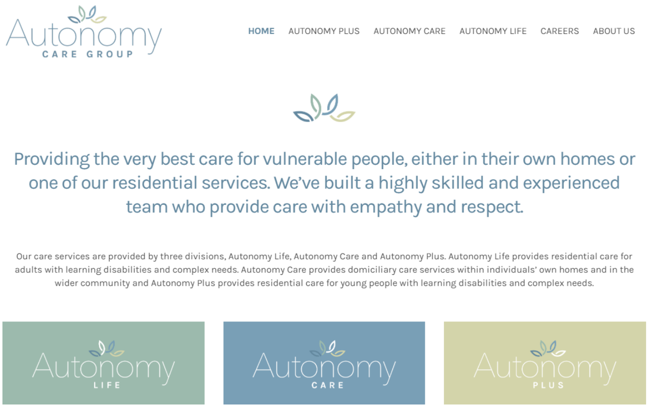
Overview

1. Motivation
2. Definition
3. Differences from Normal Web Search
4. Enterprise Search Components
5. Examples of Enterprise Search Platforms and Libraries

Motivation

Motivation

Where is the search box?



Providing the very best care for vulnerable people, either in their own homes or one of our residential services. We've built a highly skilled and experienced team who provide care with empathy and respect.

Our care services are provided by three divisions, Autonomy Life, Autonomy Care and Autonomy Plus. Autonomy Life provides residential care for adults with learning disabilities and complex needs. Autonomy Care provides domiciliary care services within individuals' own homes and in the wider community and Autonomy Plus provides residential care for young people with learning disabilities and complex needs.

Information systems need **Search** feature

Definition

Definition

Enterprise search

- is the practice of identifying and enabling specific content across the enterprise to be **indexed**, **searched**, and **displayed** to **authorized users**.
- is the organized retrieval of **structured** and **unstructured** data within your application.

Differences from Normal Web Search

Enterprise vs. Web search (Intranet vs. Internet)

- **Multiple data sources**
  Websites, files, email, etc.
- **Collecting and indexing data**
  Missed a key page?
- **Relevance and ranking algorithms**
  Popular hits and page rank
- **Users**
  - Searchers are Knowledge workers
  - Context available: department, job, location...
- **Security**
  Authenticated users
- **Single site, Single best document**
  Federated search

Enterprise Search Components

Enterprise Search Components

- Content awareness and collecting data
- Content processing and analysis
- Indexing
- Query processing
- Matching

Collecting Data

- Finding content and pulling it into the system
- **Crawlers** retrieve documents and other content
    - Over protocols like HTTP
    - Use adapters to connect to relational databases, document management systems, etc.

Content Processing

Identification Sentences determined by periods or other punctuation marks

The operator operates successfully!

Tokenization Breaking up text into tokens (words, phrases, symbols, etc.)

[The] [operator] [operates] [successfully]

Normalization Tokens to lower case to provide case-insensitive search

[the] [operator] [operates] [successfully]

Content processing II

Stop-words removing meaningless tokens, (there, so, other, etc.)

_ *[operator] [operates] [successfully]*

Stemming and lemmatization to get the normal form of the word

_ *[operate] [operate] [success]*

⋮

Synonym expansion Controlled vocabulary, manually or automatically derived thesaurus, etc. **Wordnet**

Part-of-speech tagging the **book** on the table (noun), to **book** a flight (verb)

Indexing

- The resulting terms are stored in an index, instead of storing the full text of the document
- Contains the dictionary of all unique words in the corpus
- Groups information into logical categories that in turn can be searched and return results to users
- TF-IDF

Indexing - TF-IDF

- **TF: Term Frequency**, how frequently a term occurs in **one document**.
  TF = (Number of times term t appears in a document / Total number of terms in the document)
- **IDF: Inverse Document Frequency**, how important a term is in the **corpus** IDF = log (Total number of documents / Number of documents with term t in it)

Indexing - TF-IDF

$$TF - IDF(w) = TF(w) \times \frac{1}{DF(w)}$$

The word is more popular when it appears several times in a document

The word is more important if it appears in fewer documents

- $TF(w) \rightarrow$ term frequency (number of times a term occurs in a single document)
- $DF(w) \rightarrow$ document frequency (number of documents a term occurs in within the corpus)
- $TF - IDF \rightarrow$ relative importance of the word in the document

Indexing - TF-IDF

Consider a document containing 100 words wherein the word `Example` appears 3 times. The term frequency (i.e., TF) for `Example` is calculated as follows:

$$TF_{example} = \frac{3}{100} = 0.03$$

Assume we have 10 million documents and the word `Example` appears in one thousand of these. Then, the inverse document frequency (i.e., IDF) is calculated as follows:

$$IDF_{example} = log(\frac{10\,000\,000}{1\,000}) = 4$$

. TF-IDF weight is the product of these quantities:

$$TF - IDF = 0.03 \times 4 = 0.12$$

Searching

Enterprise search applications may allow

- General free-form keyword searching
- Specialized query syntax to allow more specific queries
- A standardized query language like SQL or SPARQL

The query parser converts the query into a representation which can be used, along with the index, to determine matching results.

Query expansion for better performance (recall and precision)

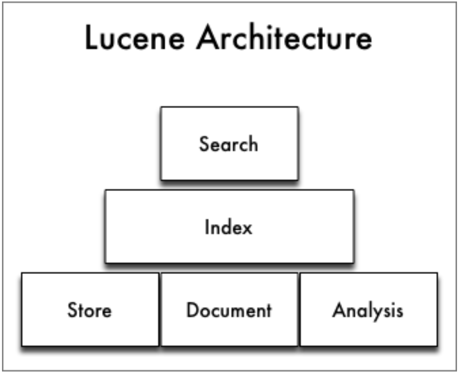
Next Level

- Search by term instead of keyword
    - Term has semantics, keyword is just a word
    - *Construction* – process of constructing something vs. *Construction* – structure (physical or logical)

Examples of Enterprise Search Platforms and Libraries

Lucene

- A powerful open-source Java full-text search library
- Makes it easy to add full-text search capability to applications
- **Not** a complete application **but** a code library and API



Lucene Architecture

Search

Index

Store

Document

Analysis

Inverted Index

- Lucene uses *inverted index* of terms
- Terms point to the documents in which they appear

Doc#1 → _ *[operate] [operate] [success]*

```
"operate": [1, 47, 72]
"success": [1, 55, 92, 107]
"search": [34, 92, 119]
"zebra": [15, 34, 55, 107]
```

Lucene - Simple Indexing Example

- In-memory index from some strings.

Indexing

```
StandardAnalyzer analyzer = new StandardAnalyzer();
Directory index = new RAMDirectory();

IndexWriterConfig config = new IndexWriterConfig(analyzer);

IndexWriter w = new IndexWriter(index, config);
addDoc(w, "Lucene in Action", "193398817");
addDoc(w, "Lucene for Dummies", "55320055Z");
addDoc(w, "Managing Gigabytes", "55063554A");
addDoc(w, "The Art of Computer Science", "9900333X");
w.close();
```

Lucene - Simple Indexing Example II

- `addDoc()` is what actually adds documents to the index
- Use of `TextField` for content we want tokenized, and `StringField` for id fields and the like, which we don't want tokenized.

Indexing - addDoc()

```
private static void addDoc(IndexWriter w, String title, String isbn)
  throws IOException {
   Document doc = new Document();
   doc.add(new TextField("title", title, Field.Store.YES));
   doc.add(new StringField("isbn", isbn, Field.Store.YES));
   w.addDocument(doc);
}
```

Lucene - Simple Query Example

- We read the query from stdin, parse it and build a Lucene `Query` out of it.

Query

```
String querystr = "your query keywords";
Query query = new QueryParser("title", analyzer).parse(querystr);
```

Lucene - Simple Search Example

- Using the `Query` we create a `Searcher` to search the index. Then a `TopScoreDocCollector` is instantiated to collect the top 10 scoring hits.

Search

```
int hitsPerPage = 10;
IndexReader reader = DirectoryReader.open(index);
IndexSearcher searcher = new IndexSearcher(reader);
TopDocs docs = searcher.search(query, hitsPerPage);
ScoreDoc[] hits = docs.scoreDocs;
```

Elasticsearch

- Open source search server powered by Lucene under the hood
- Written in Java
- Cross platform
- Scalability and distributed architecture
- HTTP REST API
- Schema-less JSON documents
- Developed by *Elastic NV*
- Near real-time search

Elasticsearch Users

- Wikimedia
- Quora
- SoundCloud
- GitHub
- Netflix
- Uber
- Slack

.

.

.

Elasticsearch - Introduction Example

- **Download** the latest distribution from `https://www.elastic.co/downloads/elasticsearch`
- **Unpack** it on your machine
- **Run** it, by launching **elasticsearch**
- **Launch** it from the web browser `http://localhost:9200`

Elasticsearch - Introduction Example

Result in the Browser

```
{
 "name" : "LAPTOP-1B98U3HM",
 "cluster_name" : "elasticsearch",
 "version" : {
  "number" : "7.5.0",
  "build_flavor" : "default",
  "build_hash" : "e9ccaed468e2fac2275a3761849cbee64b39519f",
  "build_date" : "2019-11-26T01:06:52.518245Z",
  "lucene_version" : "8.3.0",
 },
 "tagline" : "You Know, for Search"
}
```

Elasticsearch - Building a Basic Search App

Create an Index

```
PUT /myapp?pretty
```

Index a Document

```
PUT /myapp/tweet/1?pretty
{
   "name": "Jane Doe",
   "tweet": "I think elasticsearch is AWESOME",
   "date": "2013-06-03",
   "loc": {
      "lat": 13.4,
      "lon": 52.5
   }
}
```

Create an Index – Response

```
{
 "_index" : "myapp",
 "_type" : "tweet",
 "_id" : "1",
 "_version" : 1,
 "result" : "created",
 "_seq_no" : 0,
 "_primary_term" : 1
}
```

Get the Document

```
GET /myapp/tweet/1?pretty
```

Get the Document – Response

```
{
 "_index" : "myapp",
 "_type" : "tweet",
 "_id" : "1",
 "_version" : 1,
 "found" : true,
 "_source" : { ...OUR TWEET... }
}
```

Update the Document

```
PUT /myapp/tweet/1?pretty
{
   "name": "Jane Doe",
   "tweet": "I think elasticsearch is AWESOME",
   "date": "2013-06-03",
   "loc": {
      "lat": 13.4,
      "lon": 52.5
   }
}
```

Delete the Document – Response

```
Delete /myapp/tweet/1?pretty
```

Index Info – Mapping

GET `/myapp?pretty`

```
{
   "date": {"type": "date"},
   "loc": {
      "properties": {
         "lat": {"type": "float"},
         "lon": {"type": "float"}
      }
   },
   "name": {"type": "text"},
   "tweet": {"type": "text"}
   }
}
```

- Possible to upload mapping for new fields
- Do not change the mapping of existing fields

Mapping

Full text: (default)

```
{ "type": "text", index: true }
```

Exact value only:

```
{ "type": "keyword", index: true }
```

Not searchable

```
{ "type": "text", index: false }
```

Search the Index – Empty Search

```
GET /myapp/_search
{
 "query": { "match_all": {} }
}
```

POST can be used as well

Response

```
{
 "took" : 2,
 "timed_out" : false,
 "_shards" : {
  "total" : 5,
  "successful" : 5,
  "failed" : 0
 },
 "hits" : {
 "total" : 14,
  "max_score" : 1.0,
  "hits" : [ {...}]
}}
```

Filters vs. Queries

Filters

- Exact matching
- Binary yes/no
- Fast
- Cacheable

Queries

- Full text search
- Relevance scoring
- More difficult
- Not cacheable

Query:

```
{ "match": {"tweet": "search" }}
```

Filter:

```
{ "term": {"date": "2013-06-03" }}
```

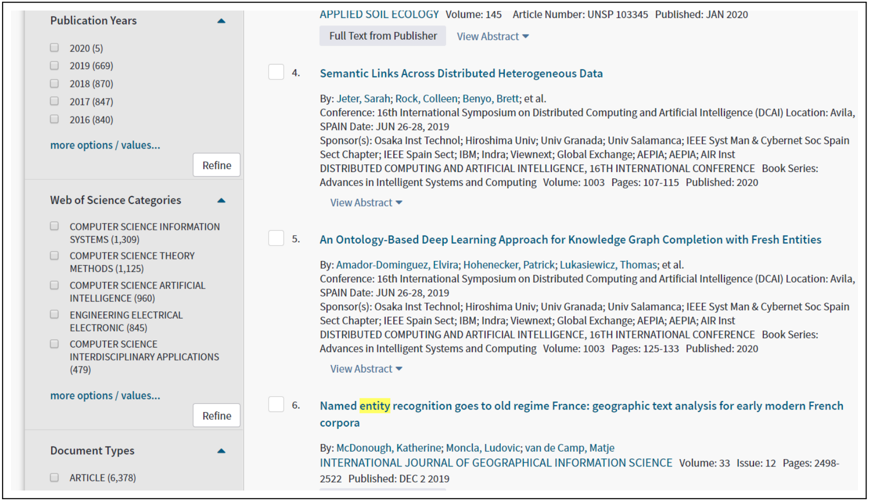
Filtered Queries & Boolean Queries

- Boolean queries allow to combine search criteria
  - `must` – criterion must match
  - `should` – criterion should match
  - `must_not` – criterion must not match
- `filter` to filter query results

```
GET /myapp/_search
{
   "query": {
      "bool": {
         "must": {
            "term": {"date": "2013-06-03"}
         },
         "filter": {
            "range": {
               "loc.lat": { "gte": 10, "lte": 15 }
            }
         }
      }
   }
}
```

Aggregations

Web of Science



Publication Years

- □ 2020 (5)
- □ 2019 (669)
- □ 2018 (870)
- □ 2017 (847)
- □ 2016 (840)

more options / values...

Web of Science Categories

- □ COMPUTER SCIENCE INFORMATION SYSTEMS (1,309)
- □ COMPUTER SCIENCE THEORY METHODS (1,125)
- □ COMPUTER SCIENCE ARTIFICIAL INTELLIGENCE (960)
- □ ENGINEERING ELECTRICAL ELECTRONIC (845)
- □ COMPUTER SCIENCE INTERDISCIPLINARY APPLICATIONS (479)

more options / values...

Document Types

- □ ARTICLE (6,378)

APPLIED SOIL ECOLOGY Volume: 145 Article Number: UNSP 103345 Published: JAN 2020

□ 4. **Semantic Links Across Distributed Heterogeneous Data**

By: Jeter, Sarah; Rock, Colleen; Benyo, Brett; et al.
Conference: 16th International Symposium on Distributed Computing and Artificial Intelligence (DCAI) Location: Avila, SPAIN Date: JUN 26-28, 2019
Sponsor(s): Osaka Inst Technol; Hiroshima Univ; Univ Granada; Univ Salamanca; IEEE Syst Man & Cybernet Soc Spain Sect Chapter; IEEE Spain Sect; IBM; Indra; Viewnext; Global Exchange; AEPIA; AEPIA; AIR Inst
DISTRIBUTED COMPUTING AND ARTIFICIAL INTELLIGENCE, 16TH INTERNATIONAL CONFERENCE Book Series: Advances in Intelligent Systems and Computing Volume: 1003 Pages: 107-115 Published: 2020

□ 5. **An Ontology-Based Deep Learning Approach for Knowledge Graph Completion with Fresh Entities**

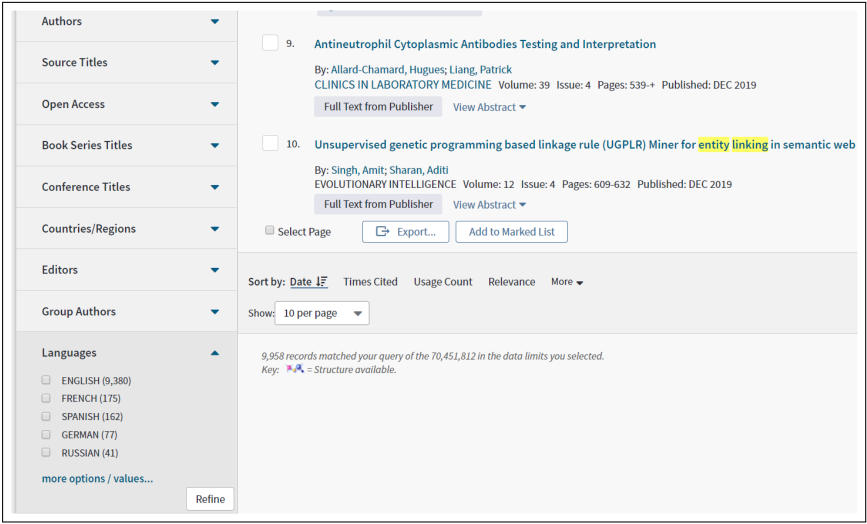
By: Amador-Dominguez, Elvira; Hohenecker, Patrick; Lukasiewicz, Thomas; et al.
Conference: 16th International Symposium on Distributed Computing and Artificial Intelligence (DCAI) Location: Avila, SPAIN Date: JUN 26-28, 2019
Sponsor(s): Osaka Inst Technol; Hiroshima Univ; Univ Granada; Univ Salamanca; IEEE Syst Man & Cybernet Soc Spain Sect Chapter; IEEE Spain Sect; IBM; Indra; Viewnext; Global Exchange; AEPIA; AEPIA; AIR Inst
DISTRIBUTED COMPUTING AND ARTIFICIAL INTELLIGENCE, 16TH INTERNATIONAL CONFERENCE Book Series: Advances in Intelligent Systems and Computing Volume: 1003 Pages: 125-133 Published: 2020

□ 6. **Named entity recognition goes to old regime France: geographic text analysis for early modern French corpora**

By: McDonough, Katherine; Moncla, Ludovic; van de Camp, Matje
INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE Volume: 33 Issue: 12 Pages: 2498-2522 Published: DEC 2 2019

Aggregations

Web of Science



□ ENGLISH (9,380)
□ FRENCH (175)
□ SPANISH (162)
□ GERMAN (77)
□ RUSSIAN (41)

□ 9. Antineutrophil Cytoplasmic Antibodies Testing and Interpretation

By: Allard-Chamard, Hugues; Liang, Patrick
CLINICS IN LABORATORY MEDICINE Volume: 39 Issue: 4 Pages: 539-+ Published: DEC 2019

□ 10. Unsupervised genetic programming based linkage rule (UGPLR) Miner for entity linking in semantic web

By: Singh, Amit; Sharan, Aditi
EVOLUTIONARY INTELLIGENCE Volume: 12 Issue: 4 Pages: 609-632 Published: DEC 2019

Show: 10 per page

9,958 records matched your query of the 70,451,812 in the data limits you selected.
Key: = Structure available.

Aggregations

```
GET /bank/_search
{
  "size": 0,
  "aggs": {
    "group_by_state": {
      "terms": {
        "field": "state"
      }
    }
  }
}
```

Aggregations

```
GET /bank/_search

{
 "size": 0,
 "aggs": {
  "group_by_age": {
   "range": {
    "field": "age",
    "ranges": [
     {
      "from": 20,
      "to": 30
     },
     {
      "from": 30,
      "to": 40
     },
     {
      "from": 40,
      "to": 50
     }
    ] },
```

```
"aggs": {
    "group_by_gender": {
     "terms": {
      "field": "gender.
         keyword"
     },
     "aggs": {
      "average_balance": {
       "avg": {
        "field": "balance"
       }
      }
     }
    }
   }
  }
 }
}
```

Solr

- Also built on Lucene
  - Similar feature set
  - Also exposes Lucene functionality, like Elasticsearch, so easy to extend.
- A part of the Apache Lucene project
- Perfect for single server search
- Clustering is there. But it's definitely not as simple as ElasticSearch
- Solr is for text search while Elasticsearch is for filtering and grouping, the analytical query workload, and not just text search.

Evaluation of Search System

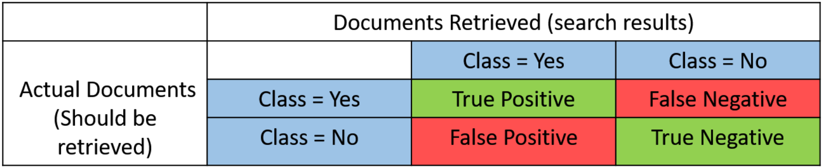
Evaluation of Search System

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$



	Documents Retrieved (search results)		
Actual Documents (Should be retrieved)		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

What is Bad Search?

- No search box
- Too many hits: Return 10 000 hits when the average user looks at the top 20 only
- Bad scoring: The most relevant item is not at the top of the list
- Poor duplicate detection: Too many similar documents
- Inability to judge user's intent: spell checking, recommendation system, auto complete.

The End

Thank You

Resources

- Enterprise Search - David Hawking in Ricardo Baeza-Yates and Berthier Ribeiro-Neto (Ed.s), Modern Information Retrieval, 2nd Ed.. Pearson Educational, pp. 641-684.
- https://lucene.apache.org/
- https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html
- https://lucene.apache.org/solr/
- https://blog.parse.ly/post/1691/lucene/