Book Characterization using Project Gutenberg's Open Library with Goodreads Reviews

Information Processing and Retrieval

Group
68

Work done so far

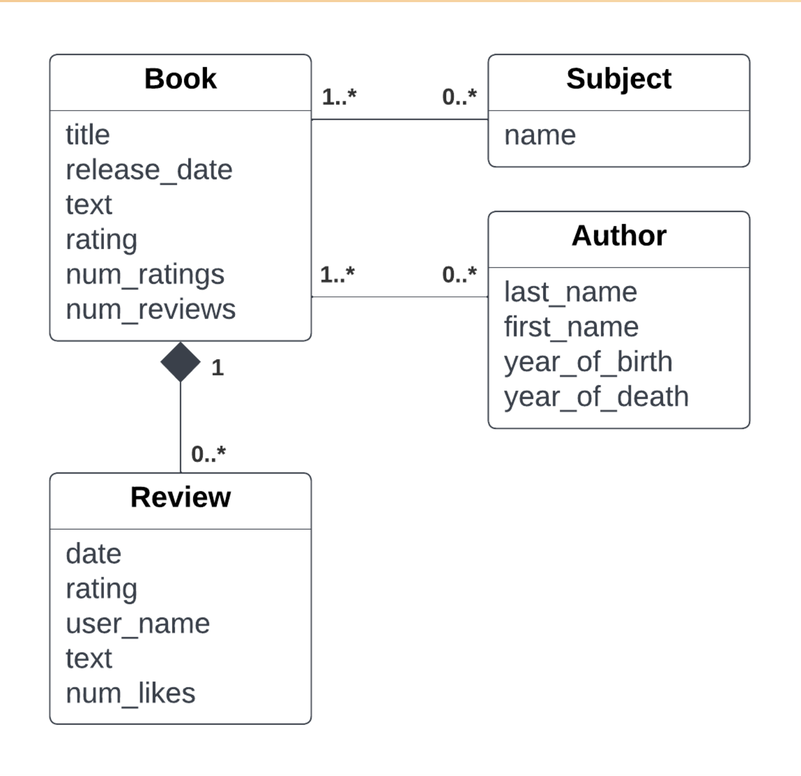


Fig. 1: Dataset Conceptual Model

- Three types of documents: Books, Authors, Reviews
- Two schemas: base schema and schema with synonyms
- Basic and improved field querying

Search scenarios

> I want to find **excerpts** about a specific concept or event.
> I want to find context on a **book quote**.
> I want to read a book from a certain **time period**.
> I want to understand what other **people think** of a book.
> I want to **browse** an extensive library of copyright-free literature.

Improvements

- **Expanded dataset** by collecting additional books and reviews
- **Named Entity Recognition** with OpenNLP
- Implementation of higher-level **REST API** and **frontend**
- **Faceting** for category suggestions
- **Highlighting** for quote search
- **Spelling correction**
- **Recomendation** of related books

Expanded Dataset

Before

- 8000 books
- 8300 reviews (10x830)
- 3.2 GB

After

- 14581 books
- 19420 reviews (~5x3884)
- 5.4 GB

Named Entity Recognition

```
{
    "name": "NLPText",
    "class": "solr.TextField",
    "analyzer": {
        "tokenizer": {
            "class": "solr.OpenNLPTokenizerFactory",
            "sentenceModel": "en-sent.bin",
            "tokenizerModel": "en-token.bin"
        },
        "filters": [
            {
                "class": "solr.OpenNLPPOSFilterFactory",
                "posTaggerModel": "en-pos-maxent.bin"
            },
            {
                "class": "solr.OpenNLPChunkerFilterFactory",
                "chunkerModel": "en-chunker.bin"
            },
            {
                "class": "solr.TypeAsPayloadFilterFactory"
            }
        ]
    }
},
```

Fig. 2: OpenNLP type for the text field

```
<updateRequestProcessorChain name="add-unknown-fields-to-the-schema" default="${update.autoCreateFields:true}"
    <processor class="solr.LogUpdateProcessorFactory"/>
    <processor class="solr.DistributedUpdateProcessorFactory"/>
    <processor class="solr.OpenNLPExtractNamedEntitiesUpdateProcessorFactory">
      <str name="modelFile">conf/en-ner-person.bin</str>
      <str name="analyzerFieldType">NLPText</str>
      <str name="source">text</str>
      <str name="dest">person_book</str>
    </processor>
    <processor class="solr.OpenNLPExtractNamedEntitiesUpdateProcessorFactory">
      <str name="modelFile">conf/en-ner-location.bin</str>
      <str name="analyzerFieldType">NLPText</str>
      <str name="source">text</str>
      <str name="dest">location_book</str>
    </processor>
    <processor class="solr.OpenNLPExtractNamedEntitiesUpdateProcessorFactory">
      <str name="modelFile">conf/en-ner-date.bin</str>
      <str name="analyzerFieldType">NLPText</str>
      <str name="source">text</str>
      <str name="dest">date_book</str>
    </processor>
    <processor class="solr.RunUpdateProcessorFactory"/>
</updateRequestProcessorChain>
```

Fig. 3: Named Entity Extractor Configuration

```
{
    "name": "person_book",
    "type": "string",
    "indexed": true,
    "multiValued": true,
    "stored": true
},
{
    "name": "location_book",
    "type": "string",
    "indexed": true,
    "multiValued": true,
    "stored": true
},
{
    "name": "date_book",
    "type": "string",
    "indexed": true,
    "multiValued": true,
    "stored": true
}
```

Fig. 4: Field definition for NER

```
{
    "name": "text",
    "type": "NLPText",
    "indexed": true
},
```

Fig. 5: Target Field for NE Extraction

Named Entity Recognition

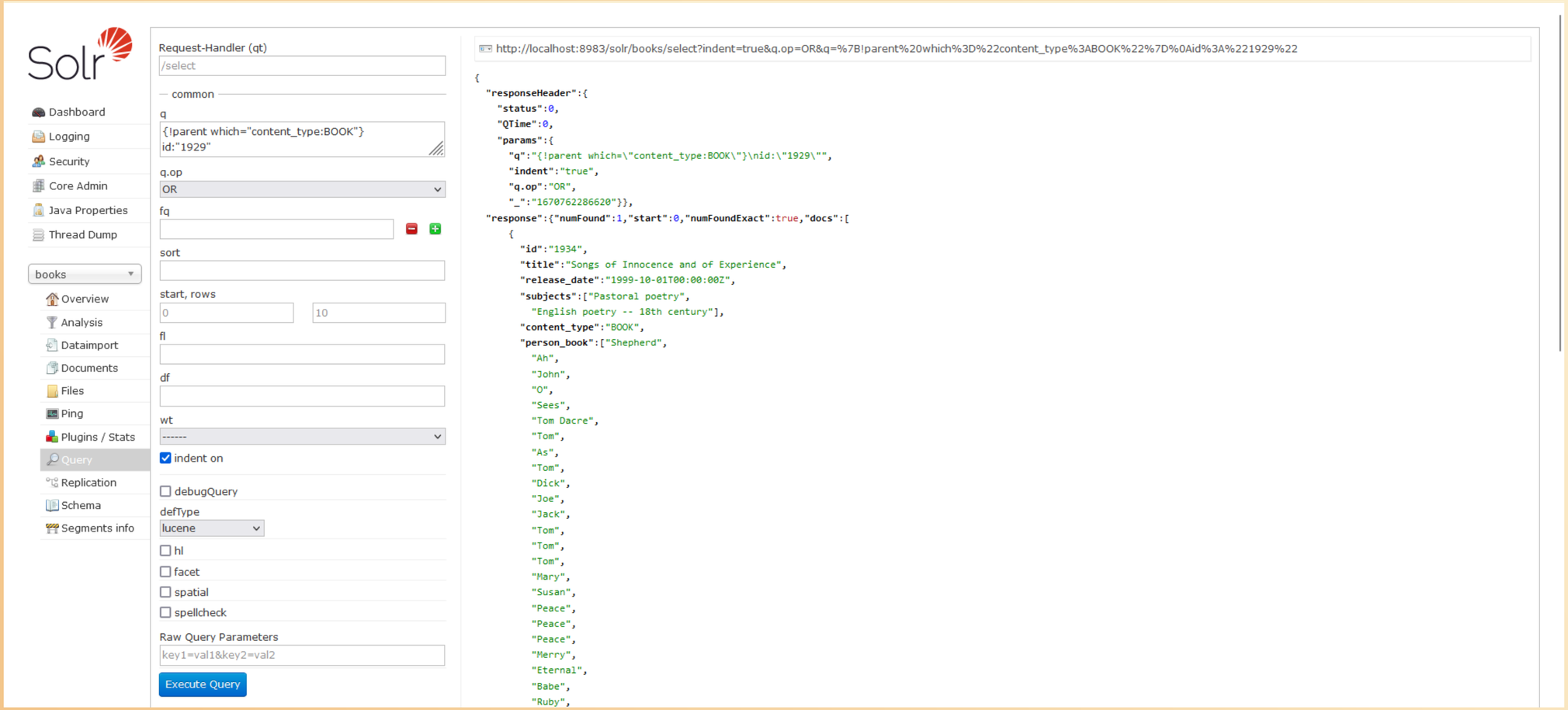


Fig. 6: Output of NER on *Songs of Innocence and of Experience*

Frontend

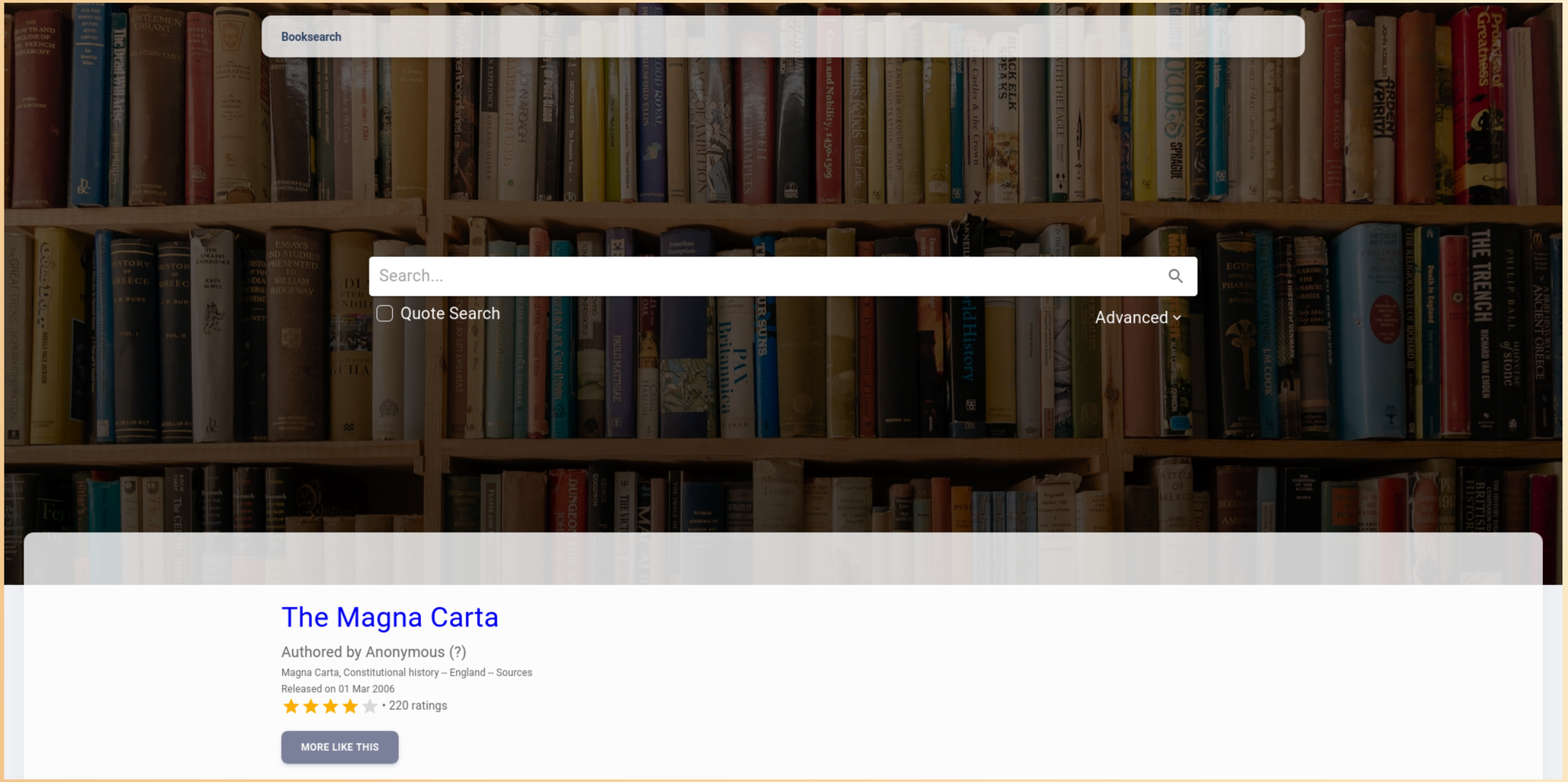


Fig. 7: Front page of the search system

Frontend

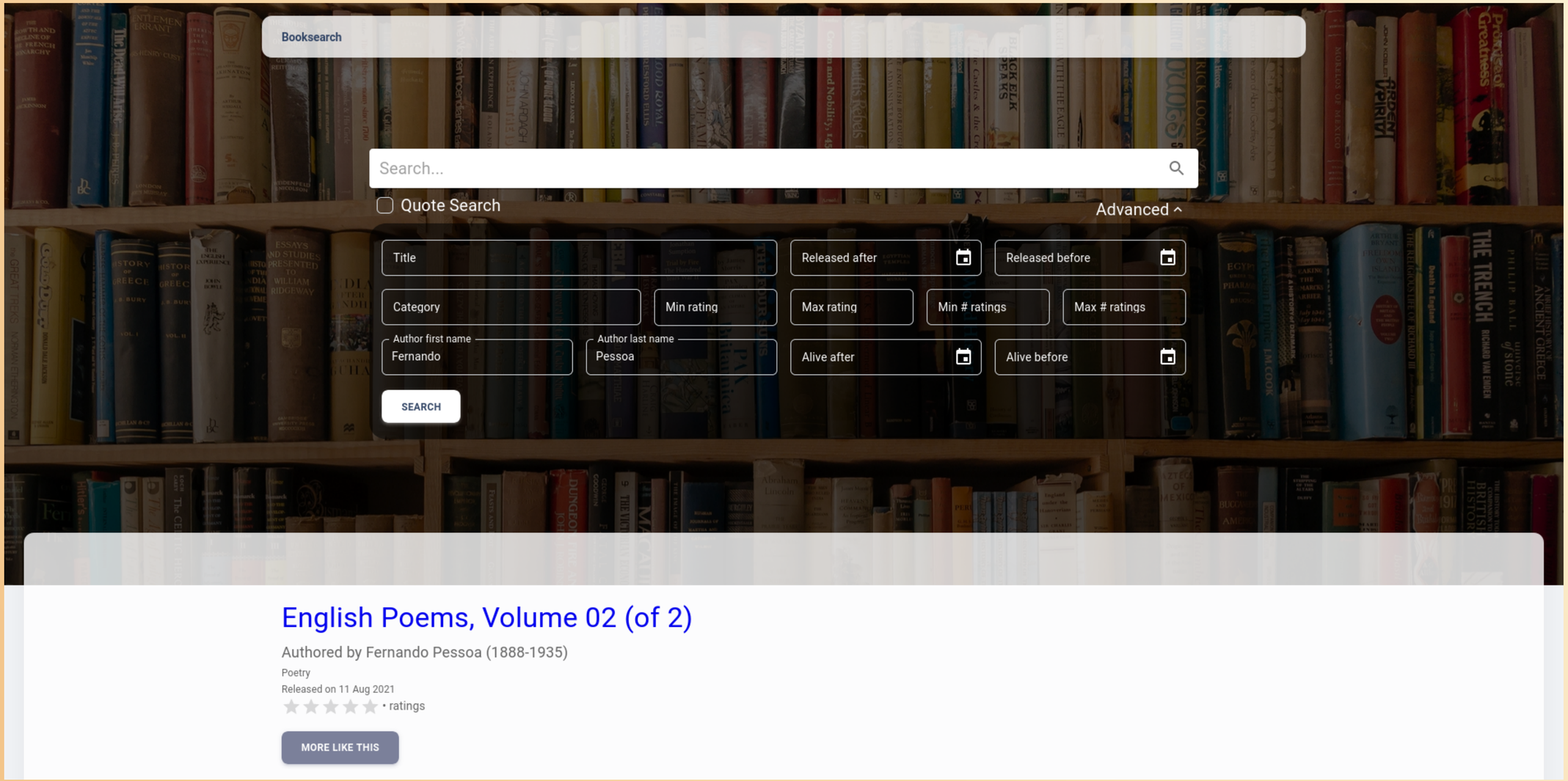


Fig. 8: Advanced search with match

Frontend

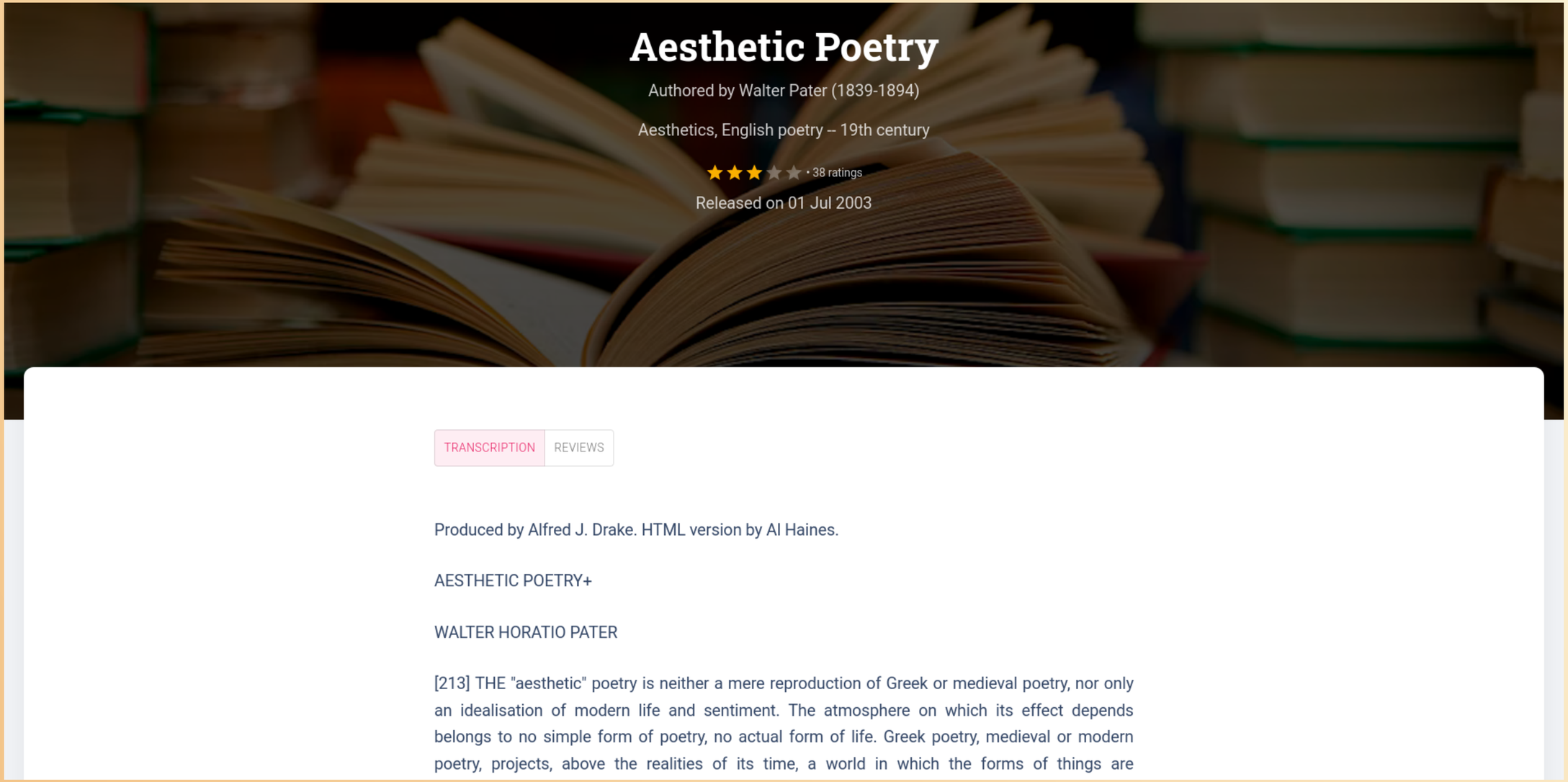


Fig. 9 Book details with transcription

Frontend

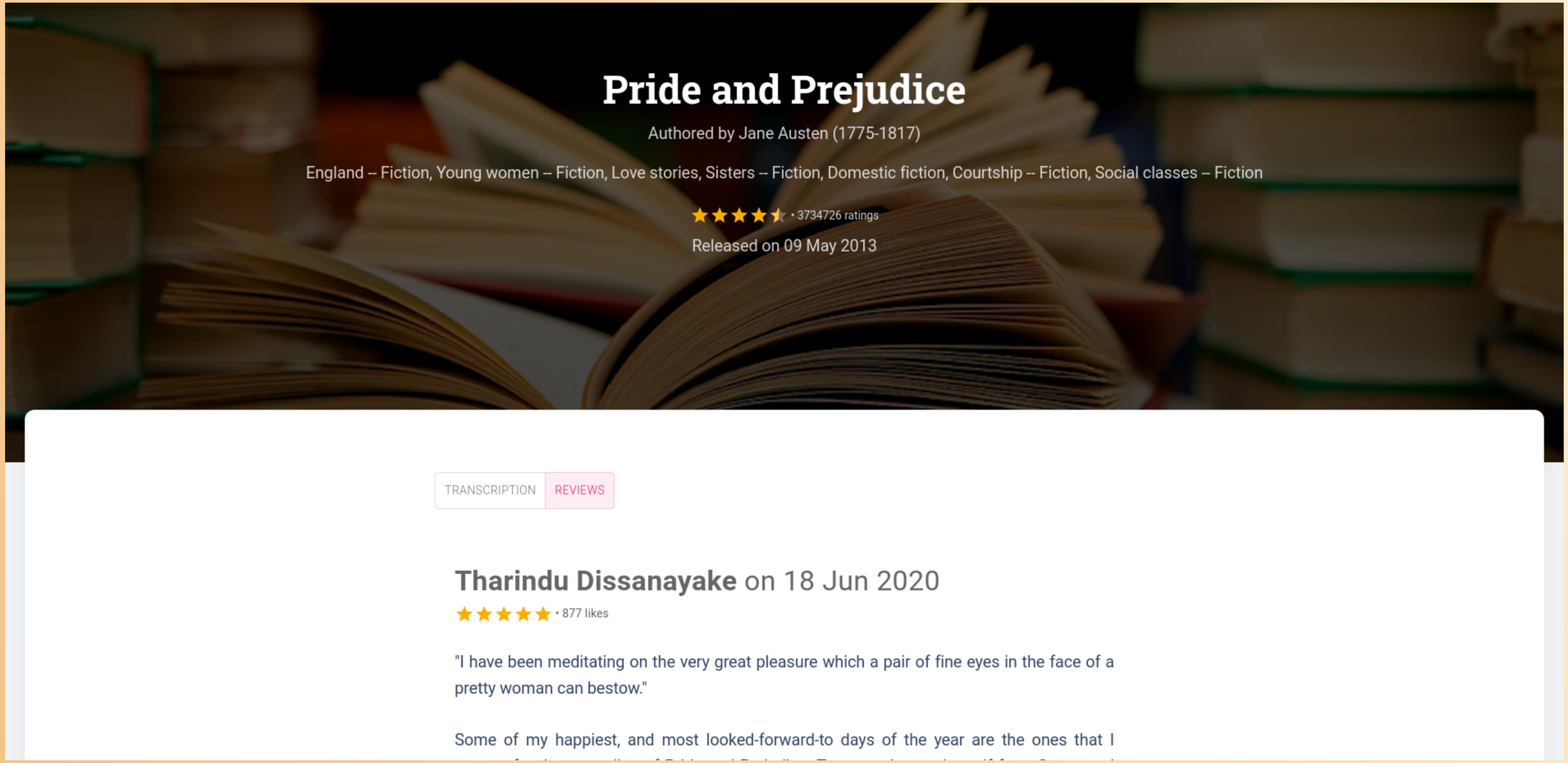


Fig. 10: Book details with reviews

REST API

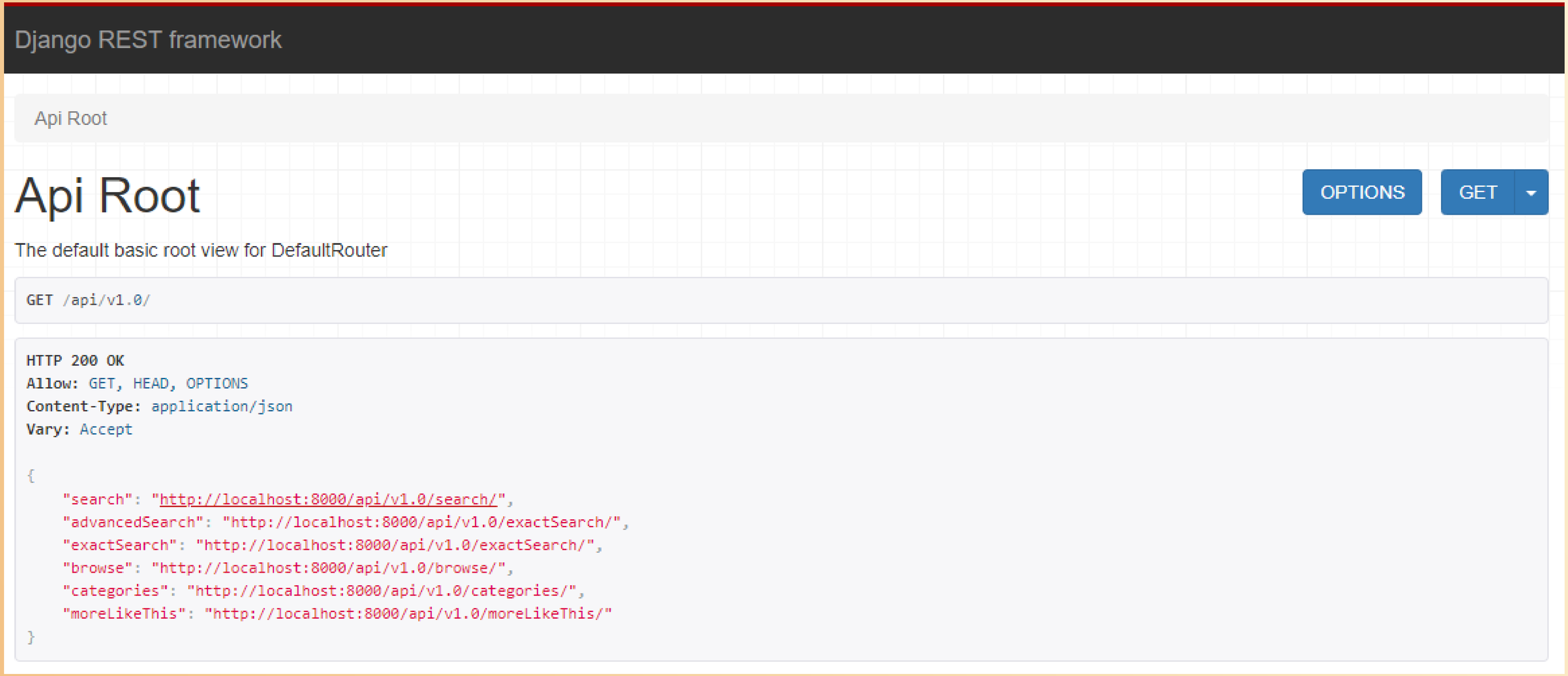
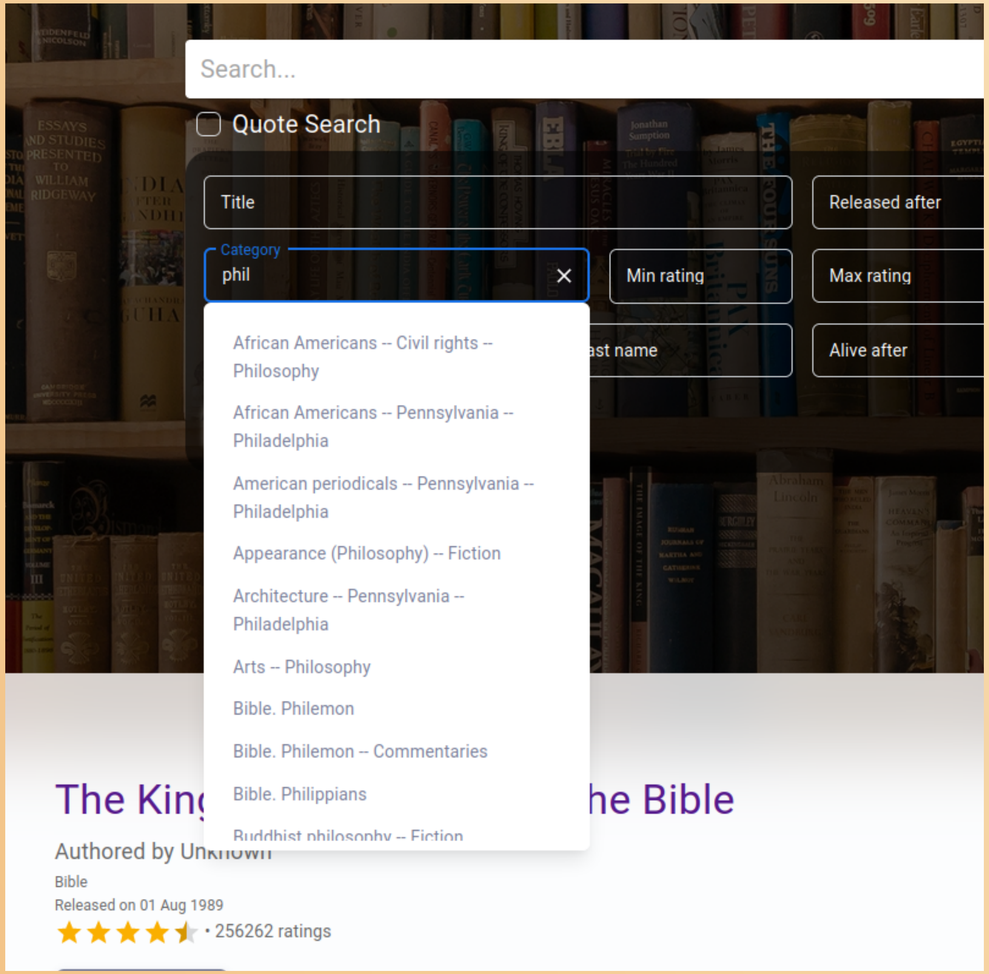


Fig. 11: REST API

Category Suggestions



- Suggestions based on what the user has typed
- Solr's faceting system to find categories
- Non-stored string field *subjects_facet*

Fig. 12: Filling categories with suggestions

Quote Search

- Applies proximity search of 5 words
- Uses Solr highlighting feature (*hl* flag)

Fig. 13: Book card with quote highlighting and relevant information

Spelling Correction

- Robustness to mistypes
- TextBlob python library (processes textual data)
- User may have intended to type like that → option to search without TextBlob processing

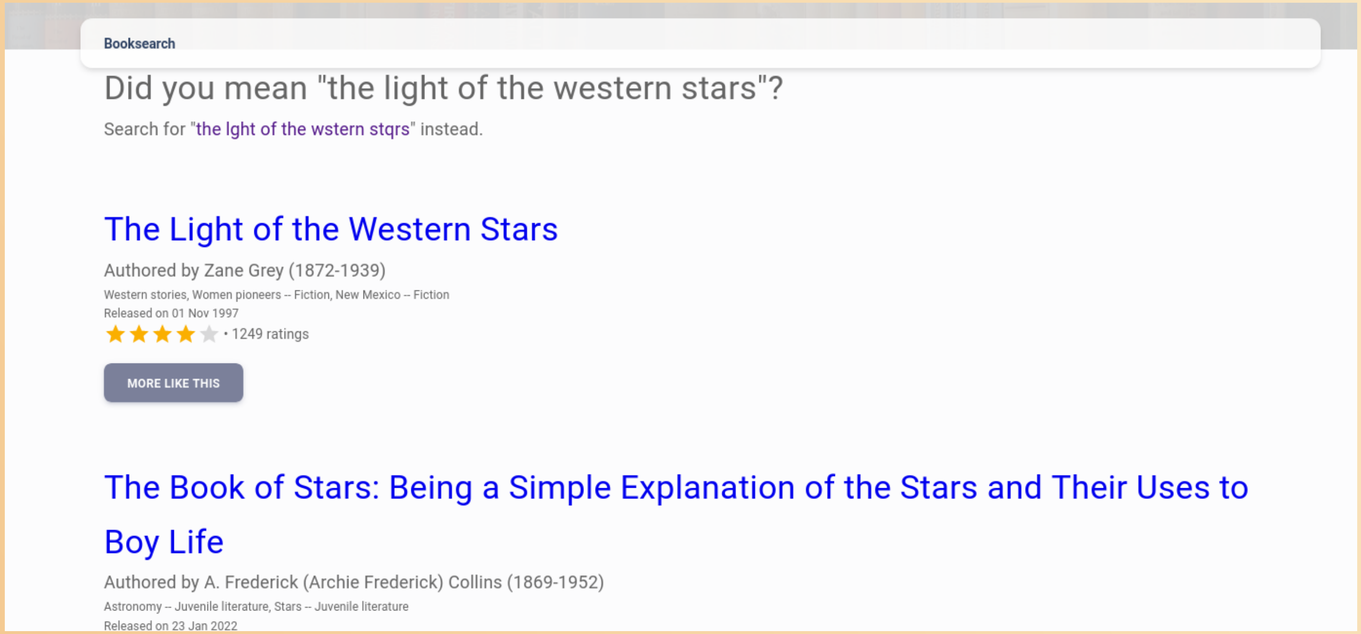


Fig. 14: Search results of corrected text

Related Book Recomendation

- Click on “More like this” to search for related books
- Uses “More like this” feature of Solr
- Loaded “More like this” component to make */mlt* route available
- Target field: text

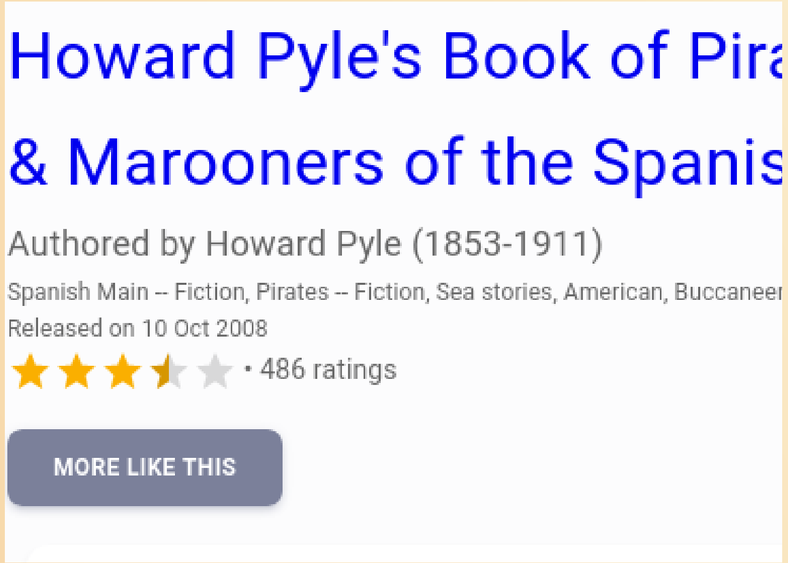


Fig. 15: More like this button

Evaluation

Information Need 1
Books explaining **home economics**

Information Need 2
Recent romantic novels released preferably around 2018

Information Need 3
Books containing the **quote "waste not, want not**", even if the user doesn't quite know the quote (inputting "want not, waste not")

Information Need 4
Books containing the **quote "waste not, want not**", even if the user doesn't quite know the quote and has **typos** (inputting "want nto, waste not")

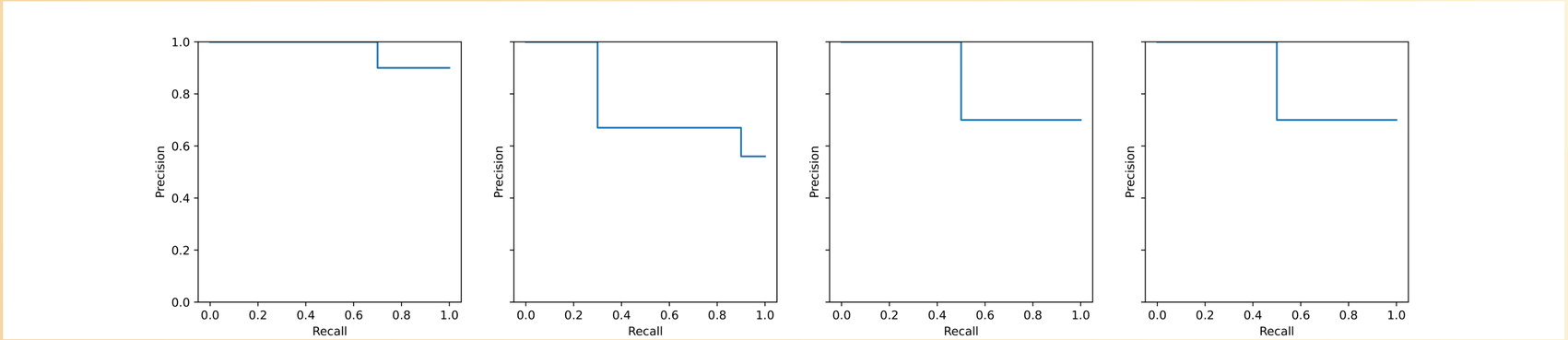


Fig. 16: P-R Curve

Table 1: Information Need 1 Metrics

Rank	IN1	IN2	IN3	IN4
1	R	R	R	R
2	R	I	R	R
3	R	R	R	R
4	R	I	I	I
5	R	R	I	I
6	R	R	R	R
7	I	I	I	I
8	R	I	R	R
9	R	R	R	R
10	R	I	R	R
P@10	0.9	0.8	0.7	0.7
AP	0.952103	0.606032	0.757976	0.757976

Mean average precision:
0.768522

Conclusions and Future Work

Conclusions:

> Information retrieval system covered **all** search scenarios
> Good schema definition and query optimization tools are essential for good performance

Future work:

> Named Entity-based queries
> OpenNLP on query - allowing for natural language queries
> Explore suggestion while writing query