IPTC EXTRA project
D1.4 EXTRA Rule Language Design

Version 1.0 - 10 March 2017

This deliverable specifies the design of the EXTRA rule language and provides several examples and use cases.

Overview

Given that the purpose and nature of EXTRA rules are very similar to that of queries, a decision has been made for the EXTRA rule language to be based on an established query language. An initial set of query languages that had been identified by the requirement analysis as potential candidates for covering a large part of the expected features (in terms of operators, listed in Table 3) include the XQuery Full Text[1], the Elastic Search Query DSL[2], and the proprietary language used by SAS Teragram suite[3]. In addition to these three options, the Contextual Query Language[4] was also considered as an attractive option, while other options, such as SQL, were not considered for selection due to their highly technical orientation and mismatch with the end user application.

Overall, there are several criteria used for selecting the query language that will serve as basis for expressing the EXTRA rules. These include the following: a) expressive power, b) simplicity for end users, c) quality and wealth of documentation, d) supporting software tools and libraries, e) alignment with EXTRA technical approach and framework (as described in D1.1).

In addition, a requirement for the selected language has been to be based on an openly available and free-to-use specification. As a result, the Teragram language was excluded from the list of candidates due to its proprietary nature.

Table 1 summarizes the main advantages and disadvantages of selecting one of the three considered languages as basis for expressing EXTRA rules. In terms of expressive power, all three candidates offer built-in support for many of the required operators, with XQuery FT having a slight advantage in terms of number of built-in supported operators. On the other hand, CQL is extensible and can be thus appropriately tailored to support the operators that are not inherently supported. In terms of simplicity for end users, CQL appears to be easier to read, even for users without prior experience. In contrast, ES DSL is probably the most difficult language for end users among the three options. In terms of documentation, XQuery FT and ES DSL are in better position compared to CQL, given the presence of very active developer/user

[1] https://www.w3.org/TR/xpath-full-text-10/
[2] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html
[3] https://www.sas.com/en_us/software/teragram.html
[4] http://www.loc.gov/standards/sru/cql/index.html

communities. Also, all three languages are supported by software libraries that could be used for parsing and translating queries.

In conclusion, our preliminary analysis of the existing options has led us to decide on using CQL as the initial basis of the development for EXTRA. Appropriate extensions will be needed in order to support the operators and requirements of EXTRA.

Table 1: Summary of characteristics of candidate query languages.

	XQuery FT	ES DSL	CQL
Expressive power	+ many of the operators are supported.	+ many of the operators are supported. - important operators as minimum_occurence as missing	+ extensible, hence its expressiveness can be enriched by users
Simplicity for users	+ some users are familiar with the language.	- not many users are familiar with the ES DSL. - not particularly easy for novice users. - in many cases queries are much more verbose, compared to the other option	+ easy to read for end users, even for those who are non-experts
Documentation	+ W3C standard means official documentation meets high standards + >3K questions tagged “xquery” on stackoverflow	+ high-quality official documentation + >23K questions tagged “elasticsearch” on stackov.	- other than the official documentation under www.loc.gov, only scarce documentation available
Supporting software	+ rich ecosystem of tools and libraries, e.g. Saxon, Zorba, BaseX, eXist	+ built-in support for ES + rich ecosystem of tools and libraries	+ support for translation to ES DSL.
Alignment with EXTRA	- an additional translation layer would be necessary in order to be supported	+ ES will serve as the indexing layer of EXTRA, so no translation would be needed for already supported operators.	- an additional translation layer would be necessary in order to be supported

In order to validate the EXTRA rule language in terms of design and in terms of implementation, an appropriate set of test rules will need to be created. To ensure that the validation and testing is thorough and sufficient, we envision that this set of rules should have the following properties:

- *Coverage of operators:* Each operator of the ones listed in Table 3 should be used by at least one of the test rules.
- *Representative size:* The set of rules should contain rules of varying size, measured by number of terms per rule. This would enable the testing both in terms of ease of reading by end of users and in terms of performance (response time).
- *Complexity:* The set of rules should be representative of the complexity of rules used in real use cases. At least some of the rules should be of very high complexity, e.g. containing wildcards, “expensive” operators, etc.

- *Language/corpus:* Rules should target both EXTRA corpora (English, German).
- *Nested rules:* The test set should contain a few nested rules at minimum.

XQuery FT

XQuery is a query and functional programming language that queries and transforms collections of structured and unstructured data, usually in the form of XML, text and with vendor-specific extensions for other data formats (JSON, binary, etc.). The language is developed by the XML Query working group of the W3C. XQuery and XPath Full Text 1.0 is an extension of the original XQuery language to enable full-text search, as well as structured searches, against XML documents. Its specification is maintained by W3C on https://www.w3.org/TR/xpath-full-text-10/

ElasticSearch Query DSL

Elasticsearch provides a full Query DSL based on JSON to define queries. The query DSL is a flexible, expressive search language that Elasticsearch uses to expose most of the power of Lucene through a simple JSON interface. The ES queries consist of query clauses, which can be either leaf clauses, i.e. clauses used to compare a field (or fields) to a query string, or compound clauses, i.e. clauses used to combine other query clauses.

Contextual Query Language

CQL, the Contextual Query Language[5], is a formal language for representing queries to information retrieval systems such as web indexes, bibliographic catalogs and museum collection information. The design objective is that queries be human readable and writable, and that the language be intuitive while maintaining the expressiveness of more complex languages. Traditionally, query languages have fallen into two camps: Powerful, expressive languages, not easily readable nor writable by non-experts (e.g. SQL, PQF, and XQuery);or simple and intuitive languages not powerful enough to express complex concepts (e.g. CCL and google). CQL tries to combine simplicity and intuitiveness of expression for simple, everyday queries, with the richness of more expressive languages to accommodate complex concepts when necessary.
Contextual Query Language:
There are some interesting software projects that could be usable by EXTRA, for instance a Java library for translating CQL to ES queries, and a Perl library that serves a similar purpose. In addition, CQL-Java is a CQL compiler written in Java, maintained by Index Data
More CQL documentation is available here and here.

[5] http://www.loc.gov/standards/sru/cql/spec.html

JSONiq

JSONiq[6] is a language, expressed in XQuery, for querying structured data (in JSON). The JSONiq syntax enables to dynamically construct objects and arrays using a syntax close to JSON. Nesting of expressions inside these constructors are allowed. JSONiq also provides a syntax for updating JSON objects and array. The language is expressive and highly optimizable for querying and updating NoSQL stores. Yet, it is designed for use by developers.

Example Rules

Table 2 presents a comparison of the three candidate languages through an example of a simple rule called “Anemia” that intends to classify documents that relate to the well-known blood disorder. The example rule makes use of the AND and OR operators, wildcards and a minimum occurrence constraint.

Table 2: An example rule called “anemia” expressed in the three language candidates.

XQuery FT:

```
(
    (
      {("anemia.*", "anaemia.*")} any word occurs at least 4 times using wildcards
    )
    ftor
    (
      (
        {("anemia.*", "anaemia.*")} any word occurs at least 3 times using wildcards
      )
      ftand
      (
        {("patient", "anemic", "treatment","AOP","haemoglobin","hemoglobin","red blood cell","red blood-cell",
"thalassaemias","sickle-cell","sickle
cell","RBC","hypoxia","symptom","hematocrit","Microcytic","Normocytic","Macrocytic","erythropoietin","anaemic","a
plastic","hemolytic","disorder","disease","myelodysplastic","illness")} any word occurs at least 4 times using
stemming
      )
    )
)
```

[6] http://www.jsoniq.org/

ES DSL[7]:

```
{
  "bool": {
    "should": [
      {
        "simple_query_string": {
          "default_operator": "or",
          "fields": [
            "_all"
          ],
          "query": "anemia* anaemia*"
        }
      },
      {
        "must": [
          {
            "simple_query_string": {
              "default_operator": "or",
              "fields": [
                "_all"
              ],
              "query": "anemia* anaemia*"
            }
          },
          {
            "match": {
              "_all": {
                "analyzer": "english_analyzer",
                "operator": "or",
                "query": "patient anemic treatment AOP haemoglobin hemoglobin red blood cell red blood-cell thalassaemias sickle-cell sickle cell RBC hypoxia symptom hematocrit Microcytic Normocytic Macrocytic erythropoietin anaemic aplastic hemolytic disorder disease myelodysplastic illness"
              }
            }
          }
        ]
      }
    ]
  }
}
```

CQL:

```
(
  (cql.allIndexes any/extra.count>4 "anemia* anaemia*")
  or
  (
    ((cql.allIndexes any/extra.count>3 "anemia* anaemia*"))
    and
    (cql.allIndexes any/stem/extra.count>4 "patient anemic treatment AOP haemoglobin hemoglobin red blood cell red blood-cell thalassaemias sickle-cell sickle cell RBC hypoxia symptom hematocrit Microcytic Normocytic Macrocytic erythropoietin anaemic aplastic hemolytic disorder disease myelodysplastic illness")
```

[7] As ES Query Language does not support retrieval of documents based on minimum occurrences of terms or statements, this query is slightly different from the queries expressed in XQuery FT and CQL. Namely, the requirement to have a statement occur more than N times is expressed in ES in a relaxed way by demanding only for its existence. The actual validation i.e. if the statement occurs more than N times, can be implemented as a post-processing step in the procedure of retrieval.

```
  )
)
```

Table 3: List of rule operators based on the IPTC EXTRA requirements. Support for unshaded rules is mandatory. Rules shaded in green are optional but high-priority, and rules shaded in yellow are optional and of secondary priority.

#	Operator	Definition
1	AND	Takes two or more statements. Category matches only if all the statements are true.
2	OR	Takes two or more statements. Category matches if at least one statement is true.
3	NOT	Used in combination with AND. Category matches if the statement does not appear in combination with statement under the AND.
4	MINIMUM	Combined with a number (e.g., MINIMUM_2). Takes one or more statements. Category matches if a minimum of x statements from the list appear in the text.
5	DISTANCE	Combined with a number. Takes two or more statements. Category matches if statements are within x number of words from each other.
6	MINIMUM OCCURRENCE	Combined with a number. Takes one or more statements. Category matches if statement appears x amount of times in the text.
7	ORDER	Takes two or more statements. Category matches if statements appear in the text in the same order that they appear in the rule.
8	SENTENCE	Takes two or more statements. Category matches if statements appear within the same sentence.
9	NOT WITHIN DISTANCE	Combined with a number. Takes two or more statements. Category matches if the statements are not within x amount of words from each other.
10	PARAGRAPH	Takes two or more statements. Category matches if all the statements occur in the same paragraph.
11	NOT IN PHRASE	Takes two statements. Category matches if the first statement occurs outside of the second statement.
12	NOT IN SENTENCE	Takes two or more statements. Category matches if all the statements do appear in the same document, but not in the same sentence.
13	NOT IN PARAGRAPH	Takes two or more statements. Category matches if all the statements do appear in the same document, but not in the same paragraph.
14	ORDER AND DISTANCE	Combined with a number. Takes two or more statements. Category matches if both statements occur in the same order in which they are written in the rule and if both are within x amount of words to each other.
15	MAXIMUM OCCURRENCE	Combined with a number. Takes at least one statement. Category matches if the statement appears no more than x amount of times in the text.
16	FROM START	Combined with a number. Takes one or more statements. Category matches if the statement appears within x amount of words from the start of the text.

17	FROM END	Combined with a number. Takes one or more statements. Category matches if the statement appears within x amount of words from the end of the text.
18	MAXIMUM SENTENCES	Combined with a number. Takes one or more statements. Category matches if the statements appear with the first x sentences.
19	MAXIMUM PARAGRAPHS	Combined with a number. Takes one or more statements. Category matches if statement appears within the first x paragraphs.
20	PARAGRAPH POSITION	Combined with a number. Takes one or more statements. Category matches if statement appears within the x-th paragraph.

Table 4: EXTRA operators expressed in the three candidate languages.

#	Operator	XQuery FT	ES DSL	CQL
1	AND	ftand	{"bool":{"must"}}	and
2	OR	ftor	{"bool":{"should"}}	or
3	NOT	ftnot	{"bool":{"must_not" }}	not
4	MINIMUM	**N/A**	**N/A**	any/extra.countunique>N[8]
5	DISTANCE	distance (exactly|at least|at most|from X to Y)	span_near with slop param	prox/relation/distance/unit/ordering[9]
6	MINIMUM OCCURRENCE	{} any word occurs at least N times	**N/A**	any/extra.count>N[10]
7	ORDER	ordered	span query with in_order param set to "true"	ordered
8	SENTENCE	same sentence	nested AND queries (sentences need to be indexed as sub-fields in ES)	prox///sentence
9	NOT WITHIN DISTANCE	($a ftand $b) ftand ftnot ($a ftand $b distance exactly N words)	span_not with dist param set	($a and $b) not ($a prox/=/N/word[11]
10	PARAGRAPH	same paragraph	nested AND queries (paragraphs need to be indexed as sub-fields in ES)	prox///paragraph
11	NOT IN PHRASE	not in	span_not	$a not ($a prox $b)
12	NOT IN SENTENCE	different sentence	nested AND queries (sentences need to be indexed as sub-fields in ES)	$a not ($a prox///sentence $b)

[8] Requires a custom relation modifier (in the 'extra' namespace); N=number of words
[9] E.g. prox/<=/1/word/unordered
[10] Requires a custom relation modifier (in the 'extra' namespace); N=number of words
[11] N=number of words

13	NOT IN PARAGRAPH	different paragraph	nested AND queries (paragraphs need to be indexed as sub-fields in ES)	$a not ($a prox///paragraph $b)
14	ORDER AND DISTANCE	distance (exactly|at least|at most|from X to Y) ordered	span_near with in_order param set to "true"	prox/relation/distance/unit/ ordered[12]
15	MAXIMUM OCCURRENCE	{} any word occurs at most N times	**N/A**	any/extra.count<N
16	FROM START	**N/A**	**N/A**	any/extra.fromstart>N
17	FROM END	**N/A**	**N/A**	any/extra.fromend>N
18	MAXIMUM SENTENCES	**N/A**	**N/A**	**TBD**
19	MAXIMUM PARAGRAPHS	**N/A**	**N/A**	**TBD**
20	PARAGRAPH POSITION	**N/A**	**N/A**	**TBD**

[12] foo prox/>/4/word/ordered bar