Using Grammar-Based Genetic Programming for Mining Subsumption Axioms Involving Complex Class Expressions

Evolutionary Axiom Discovery from Knowledge Graphs

Rémi FELIN and Andrea G. B. TETTAMANZI

Université Côte d'Azur, I3S-INRIA Laboratory
WIMMICS Team

2021/12/16

- An area of research focused on **Ontology Enrichment** of the Semantic Web.
- RDFMiner : an evolutionnary approach for OWL 2 axiom extraction ...
- ... based on association of **grammatical evolution** and **possibility theory**.

- An area of research focused on **Ontology Enrichment** of the Semantic Web.
- RDFMiner : an evolutionnary approach for OWL 2 axiom extraction ...
- ... based on association of **grammatical evolution** and **possibility theory**.

- An area of research focused on **Ontology Enrichment** of the Semantic Web.
- RDFMiner : an evolutionnary approach for OWL 2 axiom extraction ...
- ... based on association of **grammatical evolution** and **possibility theory**.

- A set of W3C Standards to express **linked data** on the Web :
  - The **RDF** '*Ressource Description Format*' standard to express data as **triple** :

    `< Subject Predicate Object >`,

  - The **OWL** '*Web Ontology Language*' standard to express **ontology** as :

    $$\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle,$$

  - The **SPARQL** '*SPARQL Protocol and RDF Query Language*' to query information directly on the web.

- A set of W3C Standards to express **linked data** on the Web :
  - The **RDF** '*Ressource Description Format*' standard to express data as **triple** :

    `< Subject Predicate Object >,`

  - The **OWL** '*Web Ontology Language*' standard to express **ontology** as :

    $$\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle,$$

  - The **SPARQL** '*SPARQL Protocol and RDF Query Language*' to query information directly on the web.

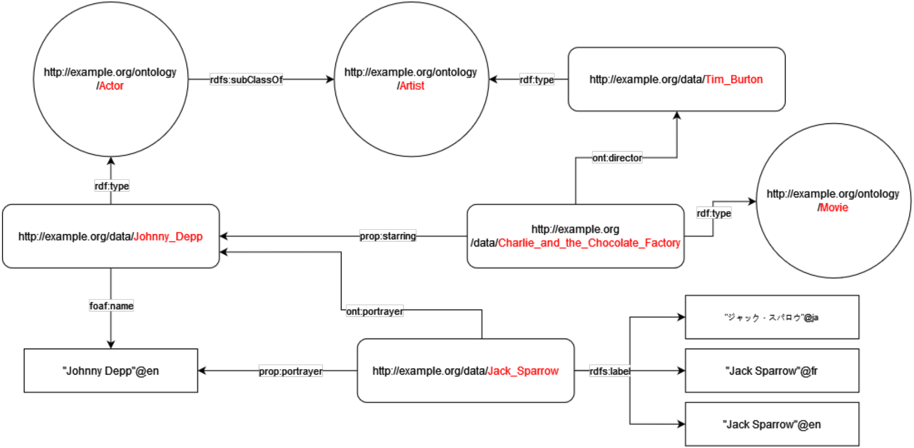
- A set of W3C Standards to express **linked data** on the Web :
  - The **RDF** '*Ressource Description Format*' standard to express data as **triple** :

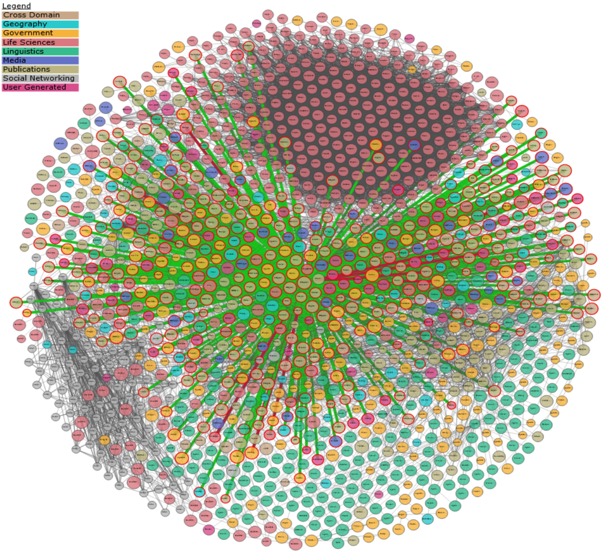
    `< Subject Predicate Object >`,
  - The **OWL** '*Web Ontology Language*' standard to express **ontology** as :

    $$\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle,$$
  - The **SPARQL** '*SPARQL Protocol and RDF Query Language*' to query information directly on the web.

- A set of W3C Standards to express **linked data** on the Web :
  - The **RDF** '*Ressource Description Format*' standard to express data as **triple** :

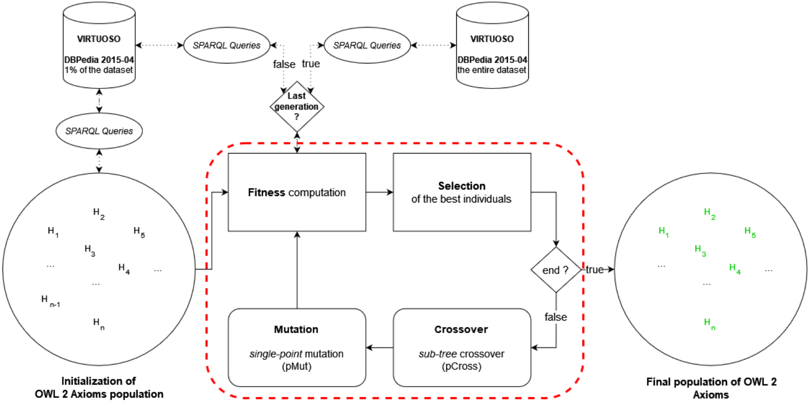
    `< Subject Predicate Object >,`
  - The **OWL** '*Web Ontology Language*' standard to express **ontology** as :
    $$\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle,$$
  - The **SPARQL** '*SPARQL Protocol and RDF Query Language*' to query information directly on the web.


http://example.org/ontology
/Actor
rdfs:subClassOf
http://example.org/ontology
/Artist
rdf:type
http://example.org/data/Tim_Burton
ont:director
http://example.org/ontology
/Movie
rdf:type
rdf:type
http://example.org/data/Johnny_Depp
prop:starring
http://example.org
/data/Charlie_and_the_Chocolate_Factory
foaf:name
ont:portrayer
"ジャック・スパロウ"@ja
"Johnny Depp"@en
prop:portrayer
http://example.org/data/Jack_Sparrow
rdfs:label
"Jack Sparrow"@fr
"Jack Sparrow"@en



The Linked Open Data Cloud from lod-cloud.net

- Here is our evolutionary approach, based on a population of OWL 2 subsumption axioms. The operators deployed aim to **maximise** the fitness of individuals, which attests to the credibility of an axiom.



An available candidate axiom, formed with ressources of the training dataset, is builded according to the **BNF Grammar** specified in a file. It provides the **template** of axioms during the experiments, such as:

Static rules of BNF Grammar

A set of rules which used to build the **structure** of axiom. This generalization is limited by the *final content* of an axiom, which gives meaning.

Dynamic rules of BNF Grammar

A set of rules, called **primitives**, which used to specify the **final content** of axiom and complete the static part. A set of *SPARQL Query* is used to define the values for a given rules.

An available candidate axiom, formed with ressources of the training dataset, is builded according to the **BNF Grammar** specified in a file. It provides the **template** of axioms during the experiments, such as:

Static rules of BNF Grammar

A set of rules which used to build the **structure** of axiom. This generalization is limited by the *final content* of an axiom, which gives meaning.

Dynamic rules of BNF Grammar

A set of rules, called **primitives**, which used to specify the **final content** of axiom and complete the static part. A set of *SPARQL Query* is used to define the values for a given rules.

An available candidate axiom, formed with ressources of the training dataset, is builded according to the **BNF Grammar** specified in a file. It provides the **template** of axioms during the experiments, such as:

Static rules of BNF Grammar

A set of rules which used to build the **structure** of axiom. This generalization is limited by the *final content* of an axiom, which gives meaning.

Dynamic rules of BNF Grammar

A set of rules, called **primitives**, which used to specify the **final content** of axiom and complete the static part. A set of *SPARQL Query* is used to define the values for a given rules.

- Static rules of SubClassOf axiom :

```
Axiom := ClassAxiom
ClassAxiom := SubClassOf
SubClassOf := 'SubClassOf' '('
    subClassExpression ' ' superClassExpression
')'
subClassExpression := Class
superClassExpression := Class
```

- Primitive Class :
  - Sparql query: `SELECT ?class WHERE { ?instance rdf:type ?class .}`
  - Write the results such as :

```
Class := dbo:Actor | dbo:Work | dbo:Artist | ...
```

- Possible result : `SubClassOf(dbo:Actor dbo:Artist)`

- Static rules of `SubClassOf` axiom :

```
Axiom := ClassAxiom
ClassAxiom := SubClassOf
SubClassOf := 'SubClassOf' '('
    subClassExpression ' ' superClassExpression
')'
subClassExpression := Class
superClassExpression := Class
```

- Primitive `Class` :
  - Sparql query: `SELECT ?class WHERE { ?instance rdf:type ?class .}`
  - Write the results such as :

    `Class := dbo:Actor | dbo:Work | dbo:Artist | ...`

- Possible result : `SubClassOf(dbo:Actor dbo:Artist)`

- Static rules of `SubClassOf` axiom :

```
Axiom := ClassAxiom
ClassAxiom := SubClassOf
SubClassOf := 'SubClassOf' '('
    subClassExpression ' ' superClassExpression
')'
subClassExpression := Class
superClassExpression := Class
```

- Primitive `Class` :
  - Sparql query: `SELECT ?class WHERE { ?instance rdf:type ?class .}`
  - Write the results such as :

```
Class := dbo:Actor | dbo:Work | dbo:Artist | ...
```

- Possible result : `SubClassOf(dbo:Actor dbo:Artist)`

- Static rules of `SubClassOf` axiom :

```
Axiom := ClassAxiom
ClassAxiom := SubClassOf
SubClassOf := 'SubClassOf' '('
    subClassExpression ' ' superClassExpression
')'
subClassExpression := Class
superClassExpression := Class
```

- Primitive `Class` :
  - Sparql query: `SELECT ?class WHERE { ?instance rdf:type ?class .}`
  - Write the results such as :

```
Class := dbo:Actor | dbo:Work | dbo:Artist | ...
```

- Possible result : `SubClassOf(dbo:Actor dbo:Artist)`

Definition of axiom codons

In order to choose *randomly* the *production* and apply operations of evolutionnary algorithms using *codons*, we used the following mapping process:

$$production = codon \mod n$$

where *production* is a value of a given rule, *codon* an integer and n the number of productions for the current non-terminal.

- Simple example applied on the previous BNF Grammar :

```
(A) Axiom := (0) ClassAxiom
(B) ClassAxiom := (0) SubClassOf
(C) SubClassOf := (0) 'SubClassOf' '('
        subClassExpression ' ' superClassExpression
    ')'
(D) subClassExpression := (0) Class
 superClassExpression := (0) Class
(E) Class := (0) dbo:Actor | (1) dbo:Work | (2) dbo:Artist
```

- We don't focus on the first rules: only 1 possibility.
- let's consider a set of integer as codons for `(E)`:
  - **codon=16**: $16 \mod 3 = 1$, thus 16 correspond to `dbo:Work`
  - **codon=92**: $92 \mod 3 = 2$, thus 92 correspond to `dbo:Artist`
  - ...
- A given axiom correspond to **a set of codons**

Possibility theory

Possibility theory is a mathematical theory of epistemic uncertainty which uses the events, variables, ... denoted ω of a universe of discourse Ω ($\omega \in \Omega$) where each ω has a degree of possibility such that $\pi : \Omega \rightarrow [0, 1]$. The theory includes a measure of possibility denoted by Π and a measure of necessity denoted by N such that:

$$\Pi(A) = \max_{\omega \in A} \pi(\omega),$$
$$N(A) = 1 - \Pi(\bar{A}) = \min_{\omega \in \bar{A}}\{1 - \pi(\omega)\},$$

where $A \in \Omega$ or $A = \{\omega : \omega \models \phi\}$.

Possibility theory applied on subsumption axioms evaluation

let us consider v_ϕ^+ the *confirmations* and v_ϕ^- the *exceptions* observed among the elements of v_ϕ, either the **support** for ϕ. We define the **possibility** $\Pi(\phi)$ and **necessity** $N(\phi)$ of an axiom as follows:

$$\Pi(\phi) = 1 - \sqrt{1 - \left(\frac{v_\phi - v_\phi^-}{v_\phi}\right)^2}, N(\phi) = \begin{cases} \sqrt{1 - \left(\frac{v_\phi - v_\phi^+}{v_\phi}\right)^2}, & if\, \Pi(\phi) = 1 \\ 0 & otherwise \end{cases}$$

Fitness definition for subsumption axioms

The fitness f of an axiom ϕ should be proportional to its **necessity**, its **possibility** and its **support**, we proposed this approach:

$$f(\phi) = v_\phi \times \frac{\Pi(\phi) + N(\phi)}{2}.$$

- We proposed the following rules:

```
(r.1) Axiom := ClassAxiom
(r.2) ClassAxiom := SubClassOf
(r.3) SubClassOf := 'SubClassOf' '('
        classExpression ' ' classExpression
      ')'
(r.4) classExpression := ObjectSomeValuesFrom |
                         ObjectAllValuesFrom |
                         ObjectIntersectionOf |
                         Class
(r.5) ObjectIntersectionOf := 'ObjectIntersectionOf' '('
        Class ' ' Class
      ')'
(r.6) ObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '('
        ObjectPropertyOf ' ' Class
      ')'
(r.7) ObjectAllValuesFrom := 'ObjectAllValuesFrom' '('
        ObjectPropertyOf ' ' Class
      ')'
```

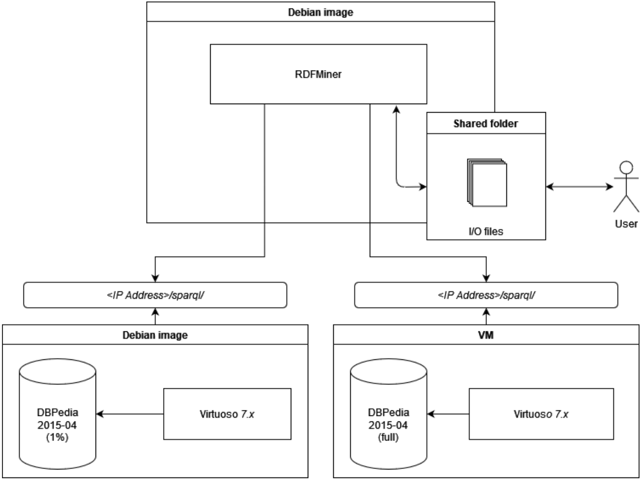
- We proposed **10 executions** of **30 generations** for each population size fixed bellow.
- Population size : **100** ; **200** ; **500**.
- We fixed the length of candidate axiom: **6**.
- probability of crossover : **80%**.
- probability of mutation : **1%**.
- the calculation of exceptions is very time-consuming. Therefore, we limit the calculation time of these requests to **30 seconds**.

- We proposed **10 executions** of **30 generations** for each population size fixed bellow.
- Population size : **100** ; **200** ; **500**.
- We fixed the length of candidate axiom: **6**.
- probability of crossover : **80%**.
- probability of mutation : **1%**.
- the calculation of exceptions is very time-consuming. Therefore, we limit the calculation time of these requests to **30 seconds**.

- We proposed **10 executions** of **30 generations** for each population size fixed bellow.
- Population size : **100** ; **200** ; **500**.
- We fixed the length of candidate axiom: **6**.
- probability of crossover : **80%**.
- probability of mutation : **1%**.
- the calculation of exceptions is very time-consuming. Therefore, we limit the calculation time of these requests to **30 seconds**.

- We proposed **10 executions** of **30 generations** for each population size fixed bellow.
- Population size : **100** ; **200** ; **500**.
- We fixed the length of candidate axiom: **6**.
- probability of crossover : **80%**.
- probability of mutation : **1%**.
- the calculation of exceptions is very time-consuming. Therefore, we limit the calculation time of these requests to **30 seconds**.

- We proposed **10 executions** of **30 generations** for each population size fixed bellow.
- Population size : **100** ; **200** ; **500**.
- We fixed the length of candidate axiom: **6**.
- probability of crossover : **80%**.
- probability of mutation : **1%**.
- the calculation of exceptions is very time-consuming. Therefore, we limit the calculation time of these requests to **30 seconds**.



RDFMiner source code: `https://github.com/RemiFELIN/RDFMining`

- We observe that a large part of the axioms found and evaluated from our training dataset with a non-zero possibility and non-zero support, keep a non-zero possibility in our full dataset.

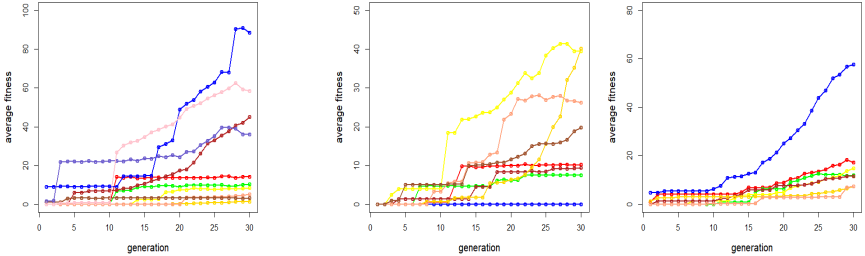
Parameter	Number of axioms	
	with training dataset	with full dataset
100	476	358
200	525	525
500	1911	1888

- the calculation of exceptions on the complete database is expensive and the time allocated is not sufficient for most of the axioms found.

- Some axioms founded have been studied in more detail, in particular by allowing the time needed to calculate exceptions to the request.
- Example:

```
SubClassOf (
   ObjectSomeValuesFrom (
      <http://dbpedia.org/property/narrated>
      <http://dbpedia.org/ontology/Agent>
   )
   <http://dbpedia.org/ontology/Work>
)
```

  - **1052 confirmations** against only **5 counterexamples** on the 1283 individuals concerned, giving a very strong possibility: **0.912** completed in **47 minutes**.



(a) popSize=100 (b) popSize=200 (c) popSize=500

Figure: Evolution of average fitness over 10 executions with the same parameters (those that have been successfully completed)

- This first approach to the discovery of the axiom of subsumption composed of a complex class allowed us to obtain interesting results.
- It highlights several areas of discussion to improve the project.
- the **multi-threading system** for the fitness calculation discussed, has been processed and tested:
  - considerably reduces the execution time. It decreases according to the number of threads available on the machine.
  - Allows us to increase the time allocated to calculating exceptions.
- We plan to extend our search for axioms composed of complex classes to other types of axioms among those provided by OWL 2.

- This first approach to the discovery of the axiom of subsumption composed of a complex class allowed us to obtain interesting results.
- It highlights several areas of discussion to improve the project.
- the **multi-threading system** for the fitness calculation discussed, has been processed and tested:
  - considerably reduces the execution time. It decreases according to the number of threads available on the machine.
  - Allows us to increase the time allocated to calculating exceptions.
- We plan to extend our search for axioms composed of complex classes to other types of axioms among those provided by OWL 2.

- This first approach to the discovery of the axiom of subsumption composed of a complex class allowed us to obtain interesting results.
- It highlights several areas of discussion to improve the project.
- the **multi-threading system** for the fitness calculation discussed, has been processed and tested:
  - considerably reduces the execution time. It decreases according to the number of threads available on the machine.
  - Allows us to increase the time allocated to calculating exceptions.
- We plan to extend our search for axioms composed of complex classes to other types of axioms among those provided by OWL 2.

- This first approach to the discovery of the axiom of subsumption composed of a complex class allowed us to obtain interesting results.
- It highlights several areas of discussion to improve the project.
- the **multi-threading system** for the fitness calculation discussed, has been processed and tested:
  - considerably reduces the execution time. It decreases according to the number of threads available on the machine.
  - Allows us to increase the time allocated to calculating exceptions.
- We plan to extend our search for axioms composed of complex classes to other types of axioms among those provided by OWL 2.

- This first approach to the discovery of the axiom of subsumption composed of a complex class allowed us to obtain interesting results.
- It highlights several areas of discussion to improve the project.
- the **multi-threading system** for the fitness calculation discussed, has been processed and tested:
  - considerably reduces the execution time. It decreases according to the number of threads available on the machine.
  - Allows us to increase the time allocated to calculating exceptions.
- We plan to extend our search for axioms composed of complex classes to other types of axioms among those provided by OWL 2.

Thank you !