Grammatical Evolution to Mine OWL Disjointness Axioms Involving Complex Concept Expressions

Thu Huong Nguyen, Andrea G. B. Tettamanzi

Grammatical Evolution to Mine OWL Disjointness Axioms Involving Complex Concept Expressions

Thu Huong Nguyen
Université Côte d'Azur, Inria, CNRS, I3S
Nice, France
thu-huong.nguyen@univ-cotedazur.fr

Andrea G. B. Tettamanzi
Université Côte d'Azur, Inria, CNRS, I3S
Nice, France
andrea.tettamanzi@univ-cotedazur.fr

***Abstract*—Discovering disjointness axioms is a very important task in ontology learning and knowledge base enrichment. To help overcome the knowledge-acquisition bottleneck, we propose a grammar-based genetic programming method for mining OWL class disjointness axioms from the Web of data. The effectiveness of the method is evaluated by sampling a large RDF dataset for training and testing the discovered axioms on the full dataset. First, we applied Grammatical Evolution to discover axioms based on a random sample of DBpedia, a large open knowledge graph consisting of billions of elementary assertions (RDF triples). Then, the discovered axioms are tested for accuracy on the whole DBpedia. We carried out experiments with different parameter settings and analyze output results as well as suggest extensions.**

***Index Terms*—Ontology Learning, OWL Axiom, Disjointness Axiom, Grammatical Evolution**

I. Introduction

There is a significant increase in research interest about detecting disjointness between concepts in knowledge bases. In terms of an *ontology* [1], [2], viewed as a formal representation of a shared domain of knowledge, the incompatibility between pairs of concepts may be defined in the form of particular types of axioms, namely class disjointness axioms. An ontology can be defined as a quadruple $\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle$, where $\mathcal{C}$ is the set of concepts represented in the form of classes; $\mathcal{R}$ is the set of relations, i.e., properties or predicates between classes; $\mathcal{I}$ is the set of all assertions, i.e. instances, in which two or more concepts are related to each other; $\mathcal{A}$ is the set of axioms. Like other types of axioms, the class disjointness axioms are formalized in the form of logical assertions and play an essential role in enhancing and constraining the information contained in the ontology, thus allowing to check its correctness or derive new information. For example, if there is a constraint of disjointness between the two concepts *Animal* and *FloweringPlant*, a reasoner will be able to reveal an error in the modeling of a knowledge base whenever the class *Animal* is associated to a resource related to the class *FloweringPlant*. As a consequence, logical inconsistencies of facts can be detected and excluded—thus improving the quality of ontologies, called *ontology enrichment*.

On the other hand, the manual acquisition of axioms is exceedingly time-consuming and expensive because of the requirement of involving domain specialists and knowledge engineers. Therefore, instead of applying top-down approaches where axioms will be generated based on schema-level information built by domain experts, bottom-up approaches, should be adopted, whereby learning methods rely on instances from several existing knowledge and information resources to suggest axioms. These methods can go under the name *axiom learning* and can be considered as one task of *ontology learning*, which can help alleviate the overall cost of extracting axioms in general. *Ontology learning* comprises the sets of methods and machine learning techniques referring to the automatic discovery and creation of ontological knowledge from scratch or the enrichment or adaptation of an existing ontology in a semi-automatic fashion using several sources [3]–[5].

One important point of the learning process is defining the type of input data sources. The use of dynamic data sources where the facts will be updated or changed in time is preferable, if one wants to achieve scalability and evolution, instead of only focusing on mostly small and uniform data collections. Such dynamic information can be extracted from various data resources on the Web, which constitute an open world of information. Indeed, the Web of data, also called the Semantic Web (SW, detailed in Section II), whose data model is the Resoure Data Framework (RDF) and whose the Linked Open Data (LOD) is a prominent implementation, has become a giant real-world data resource for learning axioms. The advantages of LOD with respect to learning described in [6] is that it is publicly available, highly structured, relational, and large compared with other resources.

As a consequence of the general lack of class disjointness axioms in existing ontologies, learning implicit knowledge in terms of axioms from a LOD repository in the context of the Semantic Web has been the object of research using several different methods. Prominent research towards the automatic creation of class disjointness axioms from RDF facts include supervised classifiers in the *LeDA* system [7], statistical schema induction via associative rule mining in the *GoldMiner* system [8], learning general class descriptions (including disjointness) from training data based in the DL-Learner framework, as pointed out in [9]. To these, we can add recent contributions relevant to class disjointness discovery. For instance, Reynaud *et al.* [10] use *Redescription Mining (RM)* to learn class equivalence and disjointness axioms with the *ReReMi* algorithm. *RM* is about extracting a category definition in terms of a description shared by all the instances

of a given class, i.e. equivalence axioms, and finding incompatible categories which do not share any instance, i.e. class disjointness axioms. Their method, based on *Formal Concept Analysis (FCA)*, a mathematical framework mainly used for classification and knowledge discovery, aims at searching for data subsets with multiple descriptions, like different views of the same objects. While category redescriptions, i.e., equivalence axioms, refer to complex types, defined with the help of relational operators like $A \equiv \exists r.C$ or $A \equiv B \sqcap \exists r.C$, in the case of incompatible categories, the redescriptions are only based on the set of attributes with the predicates of dct:subject, i.e. axioms involving atomic classes only.

Another procedure for extracting disjointness axioms [11] requires a *Terminological Cluster Tree (TCT)* to search for a set of pairwise disjoint clusters. A decision tree is built and each node in it corresponds to a concept with a logical formula. The tree is traversed to create concept descriptions collecting the concepts installed in the leaf-nodes. Then, by exploring the paths from the root to the leaves, intensional definitions of disjoint concepts are derived. Two concept descriptions are disjoint if they lie on different leaf nodes. An important limitation of the method is the time-consuming and computationally expensive process of growing a *TCT*. A small change in the data can lead to a large change in the structure of the tree. Also, like other intensional methods, that work relies on the services of a reasoning component, but suffers from scalability problems for the application to large datasets, like the ones found on the LOD, caused by the excessive growth of the decision tree. In [9], [12], we applied a heuristic method by using *Grammatical Evolution (GE)* to generate class disjointness axioms from an RDF repository. Extracted axioms include both atomic and complex axioms, i.e., defined with the help of relational operators of intersection and union; in other words, axioms like $\mathsf{Dis}(C_1, C_2)$, where C_1 and C_2 are complex class expressions including $\sqcap$ and $\sqcup$ operators. The use of a grammar allows great flexibility: only the grammar needs to be changed to mine different data repositories for different types of axioms. However, the dependence on SPARQL endpoints (i.e., query engines) for testing mined axioms against facts, i.e. instances, in large RDF repositories limits the performance of the method. In addition, evaluating the effectiveness of the method requires the participation of experts in specific domains, i.e. the use of a *Gold Standard*, which is proportional to the number of concepts. Hence, the extracted axioms are limited to the classes relevant to a small scope of topics, namely the *Work* topic of DBpedia.[1] Also, complex axioms are defined with the help of relational operators of intersection and union, which can also be mechanically derived from the known atomic axioms.

Along the lines of extensional (i.e. instance-based) methods and expanding on the GE method proposed in [9], we propose a new approach to overcome its limitations as well as to enhance the diversity of discovered types of axioms. Specifically, a set of axioms with more diverse topics is generated from a small sample of an RDF dataset which is randomly extracted from the full RDF repository, more specifically, DBpedia. Also, the type of class disjointness axioms is extended to include the existential quantification $\exists r.C$ and value restriction operators $\forall r.C$, where r is a property and C a class, which cannot be mechanically derived from a given set of atomic axioms. The grammar is updated to suit these changes. A set of candidate axioms is improved in the evolutionary process through the use of evolutionary operators of crossover and mutation. Finally, the final population of generated axioms is evaluated on the full RDF dataset, specifically the whole DBpedia, which can be considered as the objective benchmark eliminating the need of domain experts to evaluate the ability of generating axioms on a wider variety of topics. The evaluation of generated axioms in each generation of the evolutionary process is thus performed on a reasonably sized data sample, which alleviates the computational cost of query execution and enhances the performance of the method. Following [9], we apply a method based on possibility theory to score candidate axioms. It is important to mention that, to the best of our knowledge, no other method has been proposed so-far in the literature to mine the Web of data for class disjointness axioms involving complex class expressions with existential quantifications and value restrictions in addition to conjunctions.

The rest of the paper is organized as follows: some background notions are provided in Section II. The method to discover class disjointness axioms with a GE approach is presented in Section III. Section IV-C describes the experimental settings. Results are presented and analyzed in Section V. Conclusions and directions for future research are given in Section VI.

II. Preliminaries

The Semantic Web[2] (SW) is an extension of the World Wide Web (WWW) and it can be considered as the movement from the Web of documents to the Web of data. In fact, a huge amount of data on the Web is maintained in human-readable form only. The aim of the SW is to provide information in a structured form that machines too can understand. Machine-readable information combined with automated reasoning mechanisms can improve the capability of finding, retrieving, and exploiting much information not explicitly stated. For instance, only a few results of the thousands of matches typically returned by search engines carry truly relevant content. Some contents are hidden within the identified pages as well as classification and generalization of identifiers are irrelevant to the searching context. To solve this problem, semantic information containing machine-processable information called ***metadata***—a fundamental component of the SW—is embedded within Web content. Among the metadata, ***URIs*** (***Uniform Resource Identifiers***), defined in the *RFC3986* standard[3] are used to identify

[1] https://wiki.dbpedia.org/

[2] https://www.w3.org/standards/semanticweb/

[3] http://www.ietf.org/rfc/rfc3986.txt

abstract or physical resources. A URI consisting of a string of characters can be identified as a locator (***URL—Uniform Resource Locator***), a name (***URN—Uniform Resource Name***) or both. A URI that provides a means of locating the resource by describing its primary access mechanism is referred to as a URL. Meanwhile, a URI used as a URN refers to providing a globally unique name for a resource. Also, according to *RFC3987*,[4] an upgraded version of URIs are ***IRIs (International Resource Identifiers)***, which extend the *ASCII* characters of the URI version to a wide range of characters from the *Universal Character Set (Unicode)*, including many special characters in different languages. Compared with the traditional WWW, we can summarize some differences in the structure of the SW. Specifically, the SW uses a common syntax for understandable machine statements, i.e., RDF statements, and common vocabularies for easy distribution and reuse. Also, the SW relies on a logical representation of metadata with decidable logical languages to make it possible for *reasoners* to deduce implicit information. The **Resource Description Framework (RDF)**[5] [13] is mainly a data model of the SW for describing machine-processable semantics of data. RDF uses as statements triples of the form $\langle subject\ predicate\ object \rangle$. E.g., the content of the sentence "*The 1997 film Titanic was directed by James Cameron*" can be expressed in machine-accessible form as an RDF statement as follow: the subject is Film_Titanic_1997; the predicate is hasDirector; the object is James_Cameron. The statement can be described in the triple of IRIs and in the shorter representation associated with the *prefix aliases*,[6] PREFIX dbr: http://dbpedia.org/resource/ PREFIX dbo: http://dbpedia.org/ontology/ ⟨dbr:Titanic_(1997_film) dbo:director dbr:James _Cameron⟩.

The query language for RDF is ***SPARQL***,[7] which can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via some middleware.

Linked Data (LD)[8] is a method to create a Web of Data, i.e., SW, by linking datasets to one another on the Web; in this case, we talk about RDF datasets. Linked Data comprises a set of principles for sharing machine-readable interlinked data on the Web as follows: HTTP URIs (or IRIs) are used to name things, so that these things can be looked up and linked to other things; useful information is provided in standard format such as RDF on look up.

Linked Open Data[9] (**LOD**) is an association of LD and Open Data where data can be linked while being freely available for sharing and reuse. One of the prominent representatives of the LOD is **DBpedia**,[10] which comprises a rather rich collection of facts extracted from Wikipedia. DBpedia covers a broad variety of topics, which makes it a fascinating object of study for a knowledge extraction method. DBpedia owes to the collaborative nature of Wikipedia the characteristic of being incomplete and ridden with inconsistencies and errors. Also, the facts in DBpedia are dynamic, because they can change in time. DBpedia has become a giant repository of RDF triples and, therefore, it looks like a perfect testing ground for the automatic extraction of new knowledge. **OWL**[11] (**Web Ontology Language**) and **RDFS**[12] **(RDF schema)** are data modeling languages for describing RDF data. Nevertheless, OWL is much more expressive when it comes to the description of classes and properties. OWL not only comprises all the vocabulary from RDFS such as rdfs:subPropertyOf, rdfs:domain, rdfs:range but also provides further sophisticated terms to use in data modeling and reasoning. For example, OWL contains the constructors of complex class descriptions such as owl:UnionOf ($\sqcup$), owl:IntersectionOf ($\sqcap$), owl:ComplementOf ($\neg$) and can express relations between class descriptions by means of class axioms such as rdfs:SubClassOf ($\sqsubseteq$), owl:equivalentClass ($\equiv$), and owl:disjointWith. We are interested not only in extracting new knowledge from an existing knowledge base expressed in RDF, but also in being able to inject such extracted knowledge into an ontology in order to be able to exploit it to infer new logical consequences. While the former objective calls for a target language, used to express the extracted knowledge, which is as expressive as possible, lest we throttle our method, the latter objective requires using at most a decidable fragment of first-order logic and, possibly, a language which makes inference problems tractable. OWL strikes a good compromise between these two objectives. In addition, OWL is standardized and promotes interoperability with different applications. Furthermore, depending on the applications, it is possible to select an appropriate *profile* (corresponding to a different language fragment) exhibiting the desired trade-off between expressiveness and computational complexity.

III. Grammatical Evolution for Axiom Learning

We apply GE [14] to detect disjointness between class expressions, i.e. class disjointness axioms, from a training RDF dataset. Class disjointness axioms here are phenotypes which are mapped from integer strings, i.e. genotypes, based on a given BNF grammar. An evolutionary process is performed on the population of candidate axioms to extract *credible* and *general* axioms. Then, the discovered axioms are evaluated on a test dataset. In this section, we first briefly present the grammar structure used for generating OWL class disjointness axioms, then we describe the main ingredients of the evolutionary process. After that, the possibilistic evaluation of the generated candidates is introduced.

[4] http://www.ietf.org/rfc/rfc3987.txt
[5] https://www.w3.org/RDF/
[6] https://docs.microsoft.com/en-us/windows/desktop/winrm/uri-prefixes
[7] https://www.w3.org/TR/rdf-sparql-query/
[8] http://linkeddata.org/
[9] https://lod-cloud.net/
[10] https://wiki.dbpedia.org/
[11] https://www.w3.org/TR/owl-ref/
[12] https://www.w3.org/TR/rdf-schema/

A. BNF Grammar Construction

The grammar for generating well-formed OWL class disjointness axioms[13] is designed based on the functional-style grammar in the extended BNF notation used by the W3C.[14] In the functional-style syntax of OWL,[15] class disjointness axioms have the form DisjointClasses$(C_1, C_2, ..., C_n)$ where C_1, C_2,...,C_n are class expressions which can be atomic classes, i.e. single class identifiers or complex classes involving relational operators and possibly including more than one single class identifier. Like in [9], [12] and without loss of generality, we only consider the case of binary axioms such as DisjointClasses(C_1, C_2) where C_1 and C_2 can be atomic or complex classes like DisjointClasses(Building, ObjectSomeValuesFrom(hasWings, Animals)). In addition, the structure of the BNF grammar here refers to mining well-formed axioms expressing the facts contained in a given RDF triple store. Hence, only resources that actually occur in the RDF dataset should be generated. We follow the approach proposed by [9], [12] to organize the structure of a BNF grammar which ensures that changes in the contents of RDF repositories will not require the grammar to be rewritten. Accordingly, the BNF grammar is split into a static and a dynamic part.

In the static part, the production rules define the types of axioms that need to be extracted and their syntax. The content of this part is loaded from a hand-crafted text file. Unlike [9], [12], we specify it to mine only disjointness axioms involving at least one complex axiom, containing a relational operator of existential quantification $\exists$ or value restriction $\forall$, i.e., of the form $\exists r.C$ or $\forall r.C$, where r is a property and C is an atomic class. The remaining class expression can be an atomic class or a complex class expression involving an operator out of $\sqcap$, $\exists$ or $\forall$. The static part of the grammar is thus structured as follows:

```
(r1) Axiom := ClassAxiom
(r2) ClassAxiom := DisjointClasses
(r3) DisjointClasses := 'DisjointClasses''('ClassExpression1' 'ClassExpression2')'
(r4) ClassExpression1 :=  Class                (0)
                        | ObjectSomeValuesFrom    (1)
                        | ObjectAllValuesFrom      (2)
                        | ObjectIntersection       (3)
(r5) ClassExpression2 := | ObjectSomeValuesFrom (0)
                         | ObjectAllValuesFrom  (1)
(r6) ObjectIntersectionOf := 'ObjectIntersectionOf''('Class' 'Class')'
(r7) ObjectSomeValuesFrom := 'ObjectSomeValuesFrom''('ObjectPropertyOf' 'Class')'
(r8) ObjectAllValuesFrom := 'ObjectAllValuesFrom''('ObjectPropertyOf' ' Class')
```

The dynamic part contains production rules for the low-level non-terminals, called *primitives* in [9], [12]. These production rules are automatically filled at run-time by querying the SPARQL endpoint of the RDF data source at hand. The data source here is a training RDF dataset and the primitives are Class and ObjectPropertyOf.

The production rules for these two primitives are filled by SPARQL queries

```
SELECT ?class WHERE { ?instance rdf:type ?class.}
```

to extract atomic classes (represented by their IRI) and

```
SELECT ?property WHERE { ?subject ?property ?object.
                         FILTER (isIRI(?object))}
```

to extract properties (represented by their IRI) from the RDF dataset. The following is an example representing a small excerpt of an RDF dataset:

```
PREFIX dbr: http://dbpedia.org/resource/
PREFIX dbo: http://dbpedia.org/ontology/
PREFIX dbprop: http://dbpedia.org/property/
PREFIX rdf: http://www.w3.org/1999/02/22\-rdf-syntax-ns\#

dbr:Cavacoa                  rdf:type         dbo:Plant.
dbr:Mussaenda_erythrophylla  rdf:type         dbo:FloweringPlant.
dbr:The_Times                rdf:type         dbo:WrittenWork.
dbr:Chai_Ling                dbprop:spouse    dbr:Feng_Congde.
dbr:Revolution_Radio_Tour    dbprop:artist    dbr:Green_Day
```

and options for the Class and ObjectPropertyOf non-terminals are represented as follows:

```
(r.9) Class := dbo:Plant             (0)
             | dbo:FloweringPlant    (1)
             | dbo:WrittenWork       (2)
(r.10) ObjectPropertyOf := dbprop:spouse            (0)
                         | dbprop:artist            (1)
```

B. Evolutionary Process

Like in [9], our approach to axiom learning relies on a quite standard implementation of GE, whose pseudo-code is shown in Algorithm 1. In particular, we have adopted the reference implementation in the GEVA framework [15]. In this section, we focus on the specific adaptations of the standard algorithm to the problem at hand.

Algorithm 1 - GE for discovering axioms from an RDF datasets

Input: *T*: RDF Triples data; *Gr*: BNF grammar; *popSize*: the size of the population; *initlenChrom*: the initialized length of chromosome ; *maxWrap*: the maximum number of wrapping; *pElite*: elitism propotion *pselectSize*: parent selection propotion; *pCross*: the probability of crossover; *pMut*: the probability of mutation;

Output: *Pop*: a set of axioms discovered based on Gr

1: Initialize a list of chromosomes *L* of length *initlenChrom*.
   Each codon value in chromosome are integer.
2: Create a population *P* of size *popSize* mapped from list of chromosomes *L* on grammar *Gr*
3: Compute the fitness values for all axioms in *Pop*.
4: Initialize current generation number (*currentGeneration* = 0)
5: **while**(*currentGeneration* ¡ *maxGenerations)* **do**
6: Sort *Pop* by descending fitness values
7: Create a list of elite axioms *listElites* with the propotion *pElite* of the number of the fittest axioms in *Pop*
8: Add all axioms of *listElites* to a new population *newPop*
9: Select the remaining part of population after elitism selection *Lr* ← *Pop**listElites*
10: Eliminate the duplicates in *Lr* *Lr* ← Distinct (*Lr*)
11: Create a a list of axioms *listCrossover* used for crossover operation with the propotion *pselectSize* of the number of the fittest individuals in *Lr*
11: Shuffle(*listCrossover*)
12: **for** (i=0,1....*listCrossover.length-2*) **do**
10: *parent1* ← *listCrossover*[i]
13: *parent2* ← *listCrossover*[i+1]
14: *child1, child2* ← *CROSSOVER*(*parent1,parent2*) with the probability *pCross*
15: **for each** *offspring* {*child1,child2*} **do** *MUTATION*(*offspring*)
16: Compute fitness values for *child1, child2*
17: Select *w1, w2* - winners of competition between parents and offsprings *w1,w2* ← *CROWDING*((*parent1, parent2, child1, child2)*
18: Add *w*1, w2 to new population *newPop*
19: *Pop= newPop*
20: Increase the number of generation *curGeneration* by 1
21: return *Pop*

[13]https://www.w3.org/TR/owl2-syntax/#Disjoint_Classes

[14]https://www.w3.org/TR/owl2-syntax/

[15]https://www.w3.org/TR/owl2-syntax/#Functional-Style_Syntax

*1) **Initialization**:* The initial population is seeded with *popSize* random chromosomes of *initlenChrom* codons uniformly distributed over $\{0, \ldots, maxValCodon - 1\}$.

*2) **Genotype-to-Phenotype Mapping**:* The standard genotype-to-phenotype mapping is used, with at most *maxWrap* wrapping events. In case of an unsuccessful mapping (because after the maximum allowed number of wrapping events the individual is not yet completely mapped), the individual is assigned a fitness of zero, i.e., the lowest possible fitness.

*3) **Parent selection**:* To prevent the loss of the best axioms due to the application of the variation operators, a small proportion *pElite* of elite individuals is first selected and directly copied into the next generation (line 7–8 of Algorithm 1). A *candidate list* for parent selection is established by removing the duplicates from the remaining part of the population to promote diversity. The parent selection mechanism is then carried out using *truncation selection*. In particular, the top proportion *pselectSize* of the distinct individuals in the *candidate list* is replicated until the size *popSize* of population is reached. The list of selected parents is shuffled and the individuals are paired in order from the beginning to the end of the list to undergo recombination.

*4) **Variant operators**:* Unlike in [9], where single-point crossover is applied to genotypes, we use the sub-tree crossover operators at the phenotypic level, with probability *pCross*. The standard mutation operators are also applied with probability *pMut*.

*5) **Survival selection**:* Like in [9], we use the *Deterministic Crowding* method of Mahfoud [16] to improve the diversity of the population. However, there is an innovation in measuring the difference between two individuals, the *DISTANCE* function in Algorithm 2. Specifically, each offspring competes with its most similar peers, based on a phenotypic comparison instead of a genotypic one as in [9], to be selected for inclusion in the population of the next generation. The phenotypic distance between individuals is computed as their *Levenshtein distance (Edit distance)*, with the expectation of obtaining more accurate results.

Algorithm 2 - Crowding(parent1, parent2, offspring1, offspring2)

Input: ***parent1, parent2, child 1, child 2***: a crowd of individual axioms;
Output: *A*: *ListWinners*- a list containing two winners of individual axioms

1: *d1* ← *DISTANCE(parent1,child1) +DISTANCE (parent2,child2)*
d2 ← *DISTANCE(parent1, child2) + DISTANCE(parent2, child1)*
in which *DISTANCE(parent, child)* - the number of distinct codons between parent and child.
2: **if**(*d1* > *d2*)
ListWinners[0]← *COMPARE(parent1,child1)*
ListWinners[1]← *COMPARE(parent2,child2)*
else
ListWinners[0]← *COMPARE(parent1,child2)*
ListWinners[1]← *COMPARE(parent2,child1)*
in which *COMPARE(parent, child)* - defines which individual in *(parent,child)* has higher fitness value.
3: return *ListWinners*

*6) **Determining the Fitness Value**:* We follow the evaluation framework, based on possibility theory, presented in [12], which was enhanced from [9] to determine the fitness value of generated axioms in each generation, i.e. the credibility and generality of axioms. To make the paper self-contained, we recall the most important aspects of the approach.

The incompleteness and noise of some missing and erroneous facts (instances) in the RDF datasets as a result of the heterogeneous and collaborative characters of the Web of data justify adopting an axiom scoring heuristic based on possibility theory [17], which is well-suited to incomplete knowledge.

A candidate axiom ϕ is viewed as a hypothesis that has to be tested against the evidence contained in an RDF dataset. Its *content* ϕ is defined as a finite set of logical consequences

$$\text{content}(\phi) = \{\psi : \phi \models \psi\}, \tag{1}$$

obtained through the instantiation of ϕ to the vocabulary of the RDF repository; every $\psi \in \text{content}(\phi)$ may be readily tested by means of a SPARQL `ASK` query. The *support* of axiom ϕ, u_ϕ is defined as the cardinality of $\text{content}(\phi)$. The support, together with the number of confirmations u_ϕ^+ (i.e., the number of ψ for which the test is successful) and the number of counterexamples u_ϕ^- (i.e., the number of ψ for which the test is unsuccessful), are used to compute a degree of *possibility* $\Pi(\phi)$ for axiom ϕ, defined, for $u(\phi) > 0$, as

$$\Pi(\phi) = 1 - \sqrt{1 - \left(\frac{u_\phi - u_\phi^-}{u_\phi}\right)^2}.$$

Alongside $\Pi(\phi)$, the dual degree of *necessity* $N(\phi)$ could normally be defined. However, for reasons explained in [12], the necessity degree of a formula would not give any useful information for scoring class disjointness axioms against real-world RDF datasets. Possibility alone is a reliable measure of the *credibility* of a class disjointness axiom.

In terms of the generality scoring, an axiom is the more general the more facts are in the extension of its components. In [9], the generality of an axiom is defined as the cardinality of the sets of the facts in the RDF repository reflecting the support of each axioms, i.e. u_ϕ. However, in case one of the components of an axiom is not supported by any fact, its generality will be zero. Hence, the generality of an axiom should be measured by *the minimum* of the cardinalities of the extensions of the two class expressions involved, i.e. $g_\phi = min||[C]||, |||[D]|||$ where C, D are class expressions. For the above reasons, instead of the fitness function in [9],

$$f(\phi) = u_\phi \cdot \frac{\Pi(\phi) + N(\phi)}{2}, \tag{2}$$

we resorted to the following improved definition, proposed in [12]:

$$f(\phi) = g_\phi \cdot \Pi(\phi). \tag{3}$$

IV. Experimental Setup

In our experimental protocol, two phases are distinguished: (1) mining class disjointness axioms with the GE framework introduced in Section III from a training RDF dataset, i.e., a random sample of DBpedia 2015-04, and (2) testing the resulting axioms against the test dataset, i.e., the entire DBpedia 2015-04, which can be considered as an objective benchmark

to evaluate the effectiveness of the method. Before diving into those details, we first describe how we further prepare the training dataset.

A. Training Dataset Preparation

We randomly collect 1% of the RDF triples from DBpedia 2015-04 in English version (which contains 665,532,306 RDF triples) to create the *Training Dataset (TD)*.[16] Specifically, a small linked dataset is generated where RDF triples are interlinked with each other and the number of RDF triples accounts for 1% of the triples of DBpedia corresponding to each type of resource, i.e. subjects and objects. A demonstration of this mechanism to extract the sample training dataset is illustrated in Fig. 1. Let r be an initial resource for the extraction process, e.g., http://dbpedia.org/ontology/Plant; 1% of the RDF triples having r as subject, of the form $\langle r\ p\ r' \rangle$, and 1% of the triples having r as object, of the form $\langle r''\ p'\ r \rangle$, will be randomly extracted from DBpedia. Then, the same will be done for every resource r' and r'' mentioned in the extracted triples, until the size of the training dataset reaches 1% of the size of DBpedia. If the number of triples to be extracted for a resource is less than 1 (following the 1% proportion), we round it to 1 triple.

We applied the proposed mechanism to extract a training dataset containing 6,739,240 connected RDF triples with a variety of topics from DBpedia.

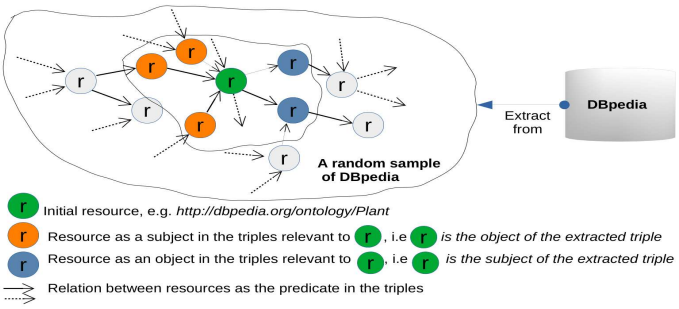


Fig. 1. An illustration of the Training Dataset sampling procedure

B. Parameters Settings for GE runs

We use the BNF grammar introduced in Section III-A. Given how the grammar was constructed, the mapping of any chromosome of length ≥ 6 will always be successful. Hence, we can set $maxWrap = 0$.

In order to investigate the ability of the method to discover class disjointness axioms for different parameter settings, we ran our algorithm in 20 different runs for each of 4 distinct population sizes, namely 1,000; 2,000; 5,000 and 10,000 individuals, respectively. In addition, to make fair comparisons possible, a set of milestones of total effort k (defined as the total number of fitness evaluations) corresponding to each population size are also recorded for each run, namely 100,000; 200,000; 300,000 and 400,000, respectively. The maximum numbers of generations *maxGenerations* (used as the stopping criterion of the algorithm) are automatically determined based on the values of the total effort k so that $popSize \cdot maxGenerations = k$. The parameters are summarized in Table I.

[16]Available for download at http://bit.ly/2Kl36wB.

TABLE I
PARAMETER VALUES FOR GE.

Parameter	Value
Total effort k	100,000; 200,000; 300,000; 400,000
initLenChrom	6
pCross	80%
pMut	1%
popSize	1000; 2000; 5000; 10000

C. Performance Evaluation

We measure the performance of the method using the entire DBpedia 2015-04 as a test set, measuring possibility and generality for every distinct axiom discovered by our algorithm. To avoid overloading DBpedia's SPARQL endpoint, we set up a local mirror using the Virtuoso Universal Server.[17]

V. Result Analysis

Running our method 20 times with the parameters shown in Table I yielded the number of distinct axioms involving complex expressions listed in Fig. 2. Together with the mandatory class expression containing the $\forall$ or $\exists$ operators, most extracted disjointness axioms contain an atomic class expression. This may be due to the fact that the support of atomic classes is usually larger than the support of a complex class expression. Table II contains some examples of discovered axioms. Full results are available online.[18] The axioms are presented in short form using prefixes dbo: http://dbpedia.org/ontology and dbprop: http://dbpedia.org/property.

We also have statistically compared the number of distinct valid axioms, i.e., axioms ϕ such that $\Pi(\phi) > 0$ and $g_\phi > 0$, discovered using different settings of *popSize* and total effort k. Overall, we can see a trend whereby the number of discovered axioms increases steadily during the early stage of evolution, i.e. for low values of k, before gradually decaying at the end of the process. This trend is most clearly visible when $popSize = 2,000$ and $popSize = 5,000$. This phenomenon may be due to the prevalence of *exploration* in the early phases of the evolutionary process, as opposed to *exploitation*, when the population, despite our efforts to preserve diversity, begins to converge towards few axioms with particularly high fitness. Depending on the population size, this may happen before reaching the first milestone of total effort $k = 100,000$ (as it is the case for $popSize = 1000$) or in the generations following the last milestone, as one could expect to observe for $popsize = 10,000$, if the evolutionary process were allowed to continue. For measuring the accuracy of our results, given that the discovered axioms come with an estimated degree of possibility, which is essentially a fuzzy degree of membership, we propose to use a fuzzy extension of the usual definition of

[17]https://virtuoso.openlinksw.com/
[18]http://bit.ly/2pisZWB

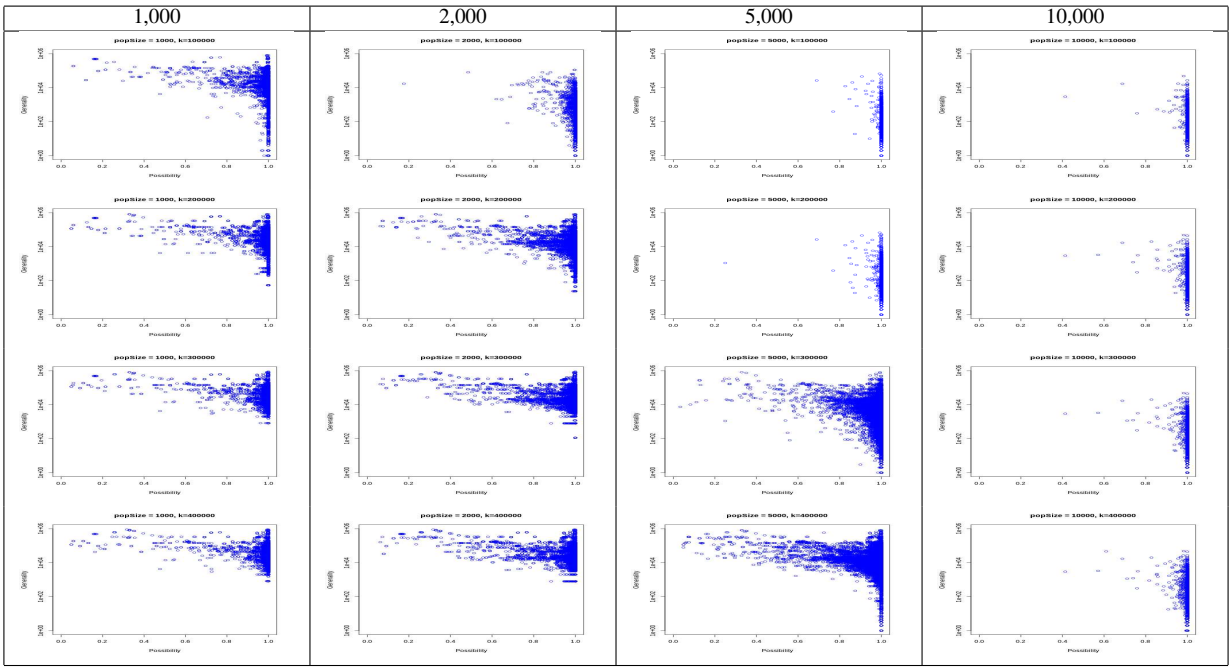
TABLE II
EXAMPLES OF ACQUIRED AXIOMS

1. DisjointClasses(dbo:Factory ObjectSomeValuesFrom(dbpprop:artist dbo:Organisation) (possibility= 1.0 ; generality= 385)
2. DisjointClasses(dbo:Airline ObjectSomeValuesFrom(dbpprop:party dbo:Organisation) (possibility= 1.0; generality= 3687)
3. DisjointClasses(dbo:MountainPass ObjectAllValuesFrom(dbprop:surface dbo:Place)) (possibility = 1.0; generality= 329)
4. DisjointClasses(ObjectIntersectionOf(dbo:ArtistDiscography dbo:MusicalWork) ObjectSomeValuesFrom(dbprop:debutagainst dbo:SportsTeam)) (possibility = 1.0; generality= 1362)

TABLE III
AVERAGE PRECISION PER RUN ($\pm$***std***)

popSize / k	1,000	2,000	5,000	10,000
100,000	0.981 ± 0.019	0.999 ± 0.002	0.998 ± 0.002	0.998 ± 0.003
200,000	0.973 ± 0.024	0.979 ± 0.011	0.998 ± 0.001	0.998 ± 0.002
300,000	0.972 ± 0.024	0.973 ± 0.014	0.993 ± 0.007	0.998 ± 0.001
400,000	0.972 ± 0.024	0.969 ± 0.018	0.980 ± 0.008	0.998 ± 0.001

TABLE IV
POSSIBILITY AND GENERALITY DISTRIBUTION OF THE DISCOVERED AXIOMS FOR DIFFERENT POPULATION SIZES (COLUMNS) AND TOTAL EFFORTS $k = 100,000, \ldots, 400,000$ (ROWS).



precision, based on the most widely used definition of fuzzy set cardinality, introduced in [18] as follows: given a fuzzy set F defined on a countable universe set Δ,

$$\|F\| = \sum_{x \in \Delta} F(x), \tag{4}$$

In our case, we may view $\Pi(\phi)$ as the degree of membership of axiom ϕ in the (fuzzy) set of the "positive" axioms. The value of precision can thus be computed against the test dataset, i.e. DBpedia 2015-04 according to the formula

$$\text{precision} = \frac{\|\text{true positives}\|}{\|\text{discovered axioms}\|} = \frac{\sum_\phi \Pi_{DBpedia}(\phi)}{\sum_\phi \Pi_{TD}(\phi)}, \tag{5}$$

where Π_{TD} and $\Pi_{DBpedia}$ are the possibility measures computed on the training dataset and DBpedia, respectively.

The results in Table III confirm the high accuracy of our axiom discovery method with a precision ranging from 0.969 to 0.998 for all the different considered sizes of population and different numbers of generations (reflected through the values of total effort). We also see that the accuracy remains stable

across different values of total effort k in the case of large populations like $popSize = 10,000$, whilst there is an opposite trend in the case of smaller populations, where the values decrease slightly as the total effort k increases. This surprising behavior suggests that the method tends to *overfit* individuals in the population after a high number of generations (reflected by the values of total effort). This overfitting may be the only way to achieve higher fitness values (as computed against the training set), whereas the evaluated axioms actually turn out to be incorrect when evaluated against the test dataset, i.e the full DBpedia. We can witness this phenomenon more clearly from the plots illustrating the distribution of axioms in terms of possibility and generality shown in Table IV. Even though most discovered axioms are highly possible ($\Pi(\phi)$ close to 1), the number of highly general axioms possessing a lower possibility increases slightly as total effort k increases. This suggests that the evolutionary process should be stopped early before axioms begin overfitting the training dataset. Indeed, with the same value of total effort, the larger populations, which correspond to a lower number of generations, as it is the case for $popSize = 10,000$, allow the method to discover axioms that correctly generalize to the full DBpedia and the evidence of the precision values in Table III seems to confirm this hypothesis.

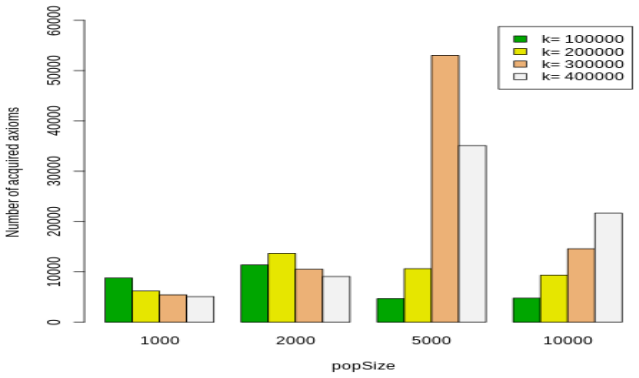


Fig. 2. Number of axioms discovered over 20 runs.

VI. Conclusion and Future Work

In this work, we presented an extension of the use of GE to discover disjointness axioms involving complex class expressions. The study aims at mining axioms containing the relational operators of existential quantification $\exists$ and value restriction $\forall$ on a wide variety of topics from DBpedia. A *training-testing* approach is also implemented to solve current limitations of performance and obtain a fair and objective assessment of its accuracy. The experimental results show that the approach is highly accurate and competitive with related approaches.

As future work, we will focus on three directions: (1) approach axiom discovery as a two-objective optimization problem to treat axiom credibility and generality as two independent criteria; (2) mine complex axioms involving additional relational operators like *owl:hasValue*, *owl:oneOf*; (3) take some complexity measurement of class expressions into account for evaluating the fitness of axioms.

Acknowledgments

This research was performed in Wimmics team which is a joint research team of Université Côte d'Azur, Inria, and I3S. Our research motto: AI in bridging social semantics and formal semantics on the Web.

This work has been partially supported by the French government, through the 3IA Côte d'Azur "Investments in the Future" project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.

References

[1] N. Guarino, D. Oberle, and S. Staab, *What Is an Ontology? Handbook on Ontologies*, ser. International Handbooks on Information Systems. Springer, 2009.

[2] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *Int. J. Hum.-Comput. Stud.*, vol. 43, no. 5-6, pp. 907–928, 1995.

[3] A. Maedche and S. Staab, "Ontology learning for the semantic web," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 72–79, March 2001.

[4] J. Lehmann and J. Völker, *Perspectives on Ontology Learning*, ser. Studies on the Semantic Web. IOS Press, 2014, vol. 18.

[5] A. Konys, "Knowledge systematization for ontology learning methods," in *KES*, ser. Procedia Computer Science, vol. 126. Elsevier, 2018, pp. 2194–2207.

[6] M. Zhu, "DC proposal: Ontology learning from noisy linked data," in *International Semantic Web Conference (2)*, ser. Lecture Notes in Computer Science, vol. 7032. Springer, 2011, pp. 373–380.

[7] J. Völker, D. Vrandecic, Y. Sure, and A. Hotho, "Learning disjointness," in *ESWC*, ser. Lecture Notes in Computer Science, vol. 4519. Springer, 2007, pp. 175–189.

[8] J. Völker, D. Fleischhacker, and H. Stuckenschmidt, "Automatic acquisition of class disjointness," *J. Web Sem.*, vol. 35, pp. 124–139, 2015.

[9] T. H. Nguyen and A. G. B. Tettamanzi, "Learning class disjointness axioms using grammatical evolution," in *EuroGP*, ser. Lecture Notes in Computer Science, vol. 11451. Springer, 2019, pp. 278–294.

[10] J. Reynaud, Y. Toussaint, and A. Napoli, "Redescription mining for learning definitions and disjointness axioms in linked open data," in *ICCS*, ser. Lecture Notes in Computer Science, vol. 11530. Springer, 2019, pp. 175–189.

[11] G. Rizzo, C. d'Amato, N. Fanizzi, and F. Esposito, "Terminological cluster trees for disjointness axiom discovery," in *ESWC (1)*, ser. Lecture Notes in Computer Science, vol. 10249, 2017, pp. 184–201.

[12] T. H. Nguyen and A. G. B. Tettamanzi, "An evolutionary approach to class disjointness axiom discovery," in *WI*. ACM, 2019, pp. 68–75.

[13] P. F. Patel-Schneider and D. Fensel, "Layering the semantic web: Problems and directions," in *International Semantic Web Conference*, ser. Lecture Notes in Computer Science, vol. 2342. Springer, 2002, pp. 16–29.

[14] M. O'Neill and C. Ryan, "Grammatical evolution," *Trans. Evol. Comp*, vol. 5, no. 4, pp. 349–358, Aug. 2001. [Online]. Available: http://dx.doi.org/10.1109/4235.942529

[15] M. O'Neill, E. Hemberg, C. Gilligan, E. Bartley, J. Mcdermott, and A. Brabazon, *GEVA - Grammatical Evolution in Java (v1.0)*, UCD's Natural Computing Research & Application group, 01 2008.

[16] S. W. Mahfoud, "Crowding and preselection revisited," in *PPSN*. Elsevier, 1992, pp. 27–36.

[17] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[18] A. De Luca and S. Termini, "A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory," *Information and Control*, vol. 20, pp. 301–312, 1972.