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

Thu Huong Nguyen, Andrea G. B. Tettamanzi

Using Grammar-based Genetic Programming for Mining Disjointness Axioms Involving Complex Class Expressions

Thu Huong Nguyen ✉[0000−0003−3744−0467] and Andrea G.B. Tettamanzi[0000−0002−8877−4654]

Université Côte d'Azur, CNRS, Inria, I3S, France
{thu-huong.nguyen,andrea.tettamanzi}@univ-cotedazur.fr

Abstract. In the context of the Semantic Web, learning implicit knowledge in terms of axioms from Linked Open Data has been the object of much current research. In this paper, we propose a method based on grammar-based genetic programming to automatically discover disjointness axioms between concepts from the Web of Data. A training-testing model is also implemented to overcome the lack of benchmarks and comparable research. The acquisition of axioms is performed on a small sample of DBpedia with the help of a Grammatical Evolution algorithm. The accuracy evaluation of mined axioms is carried out on the whole DBpedia. Experimental results show that the proposed method gives high accuracy in mining class disjointness axioms involving complex expressions.

Keywords: Ontology Learning · OWL Axiom · Disjointness Axiom · Genetic Programming · Grammatical Evolution.

1 Motivation

The growth of the Semantic Web (SW) and of its most prominent implementation, the Linked Open Data (LOD), has made a huge number of interconnected RDF (Resource Definition Framework) triples freely available for sharing and reuse. LOD have thus become a giant real-world data resource that can be exploited for mining implicit knowledge, i.e., for *Knowledge Discovery from Data (KDD)*. Such wealth of data can be organized and made accessible by *ontologies* [1, 2], formal representations of shared domains of knowledge, which play an essential role in data and knowledge integration. Through a shared schema, ontologies support automatic reasoning such as query answering or classification over different data sources. In the structure of ontologies, the definition about the incompatibility between pairs of concepts, in the form of class disjointness axioms, is important to ensure the quality of ontologies. Specifically, like other types of axioms, class disjointness axioms allow to check the correctness of a knowledge base or to derive new information, a task that is sometimes called *knowledge enrichment*. For instance, a reasoner will be able to deduce an error,

i.e., a logical inconsistency of facts in the ontology, whenever the class *Fish* is associated to a resource related to the class *Planet*, if there is a constraint of disjointness between the two concepts *Fish* and *Planet*.

However, the manual acquisition of axioms, a central task in ontology construction, is exceedingly expensive and time-consuming and mainly depends on the availability of expert resources, i.e., domain specialists and knowledge engineers. We focus on a subtask of *ontology learning*, which goes under the name of *axiom learning*, and specifically on the learning of class disjointness axioms. Axiom learning is essentially bottom up. While top-down approaches require schema-level information built by domain experts to suggest axioms, bottom-up approaches use learning algorithms and rely on instances from several existing knowledge and information resources to mine axioms. Axiom learning algorithms can help alleviate the overall cost of extracting axioms and of building ontologies in general.

In terms of input data sources for the learning process, supporting dynamic data sources, where the facts are 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 of LOD, which constitute an open world of information. Indeed, the advantages of LOD with respect to learning, as argued in [3], is that it is publicly available, highly structured, relational, and large compared to 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 classification, like in the *LeDA* system [4], statistical schema induction via associative rule mining, like in the *GoldMiner* system [5], and learning general class descriptions (including disjointness) from training data, like in the DL-Learner framework [6]. Furthermore, recent research has proposed using unsupervised statistical approaches like Formal Concept Analysis (FCA) [7] or Terminological Cluster Trees (TCT) [8], to discover disjointness axioms.

Along the lines of extensional (i.e., instance-based, bottom-up) methods and expanding on the Grammatical Evolution(GE) method proposed in [9, 10], 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 a full RDF repository, more specifically, DBpedia. Also, the type of mined class disjointness axioms is extended to include the existential quantification ($\exists r.C$) and value restriction ($\forall r.C$) constructors, where r is a property and C a class, which cannot be mechanically derived from a given set of atomic axioms. Indeed, it is not because one knows that, for instance, the two classes Person and City are disjoint, that one can conclude that, say, Person and $\forall$birthPlace.City (i.e., who was born in a city) are disjoint too; indeed we all

know they are not! Conversely, knowing that Person and Writer are *not* disjoint does not allow us to conclude that Person and ∀author.Writer are not disjoint either. We propose a specific axiom grammar that generates the axioms of the type we target. A set of candidate axioms is improved thanks to an evolutionary process through the use of the 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 an objective benchmark, thus eliminating the need of domain experts to assess the quality of the generated axioms on a wide 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 basics in GE are provided in Section 2. The method to discover class disjointness axioms with a Grammar-based Genetic Programming approach is presented in Section 3. Section 4 describes the experimental settings. Results are presented and discussed in Section 5. A brief survey of current related work in the field of learning class disjointness axioms is provided in Section 6. Finally, conclusions and directions for future research are given in Section 7.

2 Basic Concepts of Grammar-Based Genetic Programming

Genetic Programming (GP) [11, 12] is an evolutionary approach that extends genetic algorithms (GA) to allow the exploration of the space of computer programs. Inspired by biological evolution and its fundamental mechanisms, these programs are "bred" using iterative improvement of an initially random population of programs. That is an *evolutionary process*. At each iteration, known as a *generation*, improvements are made possible by stochastic variation, i.e., by a set of *genetic operators*, usually *crossover* and *mutation* and probabilistic selection according to pre-specified criteria for judging the quality of an individual (solution). According to the levels of fitness, the process of selecting individuals, called *fitness-based selection*, is performed to create a list of better qualified individuals as input for generating a new set of candidate solutions in the next generation. The new solutions of each generation are bred by applying genetic operators on the selected old individuals. Then, *replacement* is the last step and decides which individuals stay in a population and which are replaced on a par, with selection influencing convergence.

A grammar-based form of GP, namely *Grammatical Evolution* (GE) [13], differs from traditional GP in that it distinguishes the search space from the so-

lution space, through the use of a grammar-mediated representation. Programs, viewed as *phenotypic solutions* or *phenotypes*, are decoded from variable-length binary strings, i.e., *genotypic individuals* or *genotypes*, through a transformation called *mapping process*. According to it, the variable-length binary string genomes, or *chromosomes*, are split into consecutive groups of bits, called *codons*, representing an integer value, used to select, at each step, one of a set of production rules from a formal grammar, typically in *Backus-Naur form (BNF)*, which specifies the syntax of the desired programs. A *BNF grammar* is a context-free grammar consisting of terminals and non-terminals and being represented in the form of a four-tuple $\{N, T, P, S\}$, where N is the sets of non-terminals, which can be extended into one or more terminals; T is the set of terminals which are items in the language; P is the set of the production rules that map N to T; S is the start symbol and a member of N. When there are a number of productions that can be used to rewrite one specific non-terminal, they are separated by the '|' symbol.
In the mapping process, codons are used consecutively to choose production rules from P in the BNF grammar according to the function:

$$production = codon \ \mathbf{modulo} \begin{bmatrix} \text{Number of productions} \\ \text{for the current non-} \\ \text{terminal} \end{bmatrix} \tag{1}$$

3 A Grammar-Based GP for Disjointness Axioms Discovery

We consider axiom discovery as an evolutionary process and reuse the GE method of [9, 10] to mine class disjointness axioms with a few modifications. As said, we focus on axioms involving class expressions with existential quantification and value restriction. Also, a "training-testing" model is applied. Specifically, the learning process is performed with the input data source derived from a training RDF dataset, a random sample of DBpedia, whereas the evaluation of discovered axioms is based on a testing one, namely the full DBpedia. In terms of GE, axioms are "programs" or "phenotypes", obeying a given BNF grammar. A population of candidate genotypic axioms, encoded as variable-length integer strings, i.e., numerical chromosomes, is randomly generated. Then, a mapping process based on the BNF grammar is performed to translate these chromosomes to phenotypic axioms. The set of axioms is maintained and iteratively refined to discover axioms that satisfy two key quality measures: generality and credibility. The quality of generated axioms can be enhanced gradually during the evolutionary process by applying variation operators, i.e., crossover and mutation, on phenotypic axioms. In this section, we first introduce the BNF grammar construction and a specific example illustrating the decoding phase to form well-formed class disjointness axioms. A possibilistic framework for the evaluation of the discovered axioms is then presented in detail.

3.1 BNF Grammar Construction

The functional-style grammar[1] used by the W3C is applied to design the grammar for generating well-formed OWL class disjointness axioms. Like in [9, 10] and without loss of generality, we only focus on the case of binary axioms of the form DisjointClasses(C_1, C_2), where C_1 and C_2 may be atomic or complex classes involving relational operators and possibly including more than one single class identifier, like DisjointClasses(VideoGame, ObjectAllValuesFrom(hasStadium, Sport)). The structure of the BNF grammar here aims at mining well-formed axioms expressing the facts, i.e., instances, 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, 10] 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. The grammar is split into a static and a dynamic part. The static part defines the syntax of the types of axioms to be extracted. The content of this part is loaded from a hand-crafted text file. Unlike [9, 10], 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, 10]. 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 the following SPARQL queries to extract atomic classes and properties (represented by their IRI) from the RDF dataset.

```
SELECT ?class WHERE { ?instance rdf:type ?class.}
SELECT ?property WHERE { ?subject ?property ?object.
                         FILTER (isIRI(?object))}
```

Let us consider an example representing a small sample of an RDF dataset:

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

```
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:Amblycera       rdf:type            dbo:Animal.
dbr:Salweenia       rdf:type            dbo:Plant.
Dbr:Himalayas       rdf:type            dbo:NaturalPlace.
dbr:Amadeus         rdf:type            dbo:Work.
dbr:Cat_Napping     dbprop:director     dbr:William_Hanna.
dbr:With_Abandon    dbprop:artist       dbr:Chasing_Furies.
dbr:Idris_Muhammad  dbprop:occupation   dbr:Drummer.
dbr:Genes_Reunited  dbo:industry        dbr:Genealogy.
```

The productions for Class and ObjectPropertyOf would thus be:

```
(r9) Class :=  dbo:Animal       (0)     (r10) ObjectPropertyOf :=  dbprop:director      (0)
            |  dbo:Plant        (1)                                |  dbprop:artist        (1)
            |  dbo:NaturalPlace (2)                                |  dbprop:occuptation   (2)
            |  dbo:Work         (3)                                |  dbo:industry         (3)
```

3.2 Translation to Class Disjointness Axioms

We illustrate the decoding of an integer chromosome into an OWL class disjointness axiom in functional-style syntax through a specific example. Let the chromosome be 352,265,529,927,419. There is only one production for the non-terminals `Axiom`, `ClassAxiom`, `DisjointClasses`, `ObjectIntersectionOf`, `ObjectSomeValuesFrom` and `ObjectAllValuesFrom` as it can be seen from Rules 1–3, and 6–8. In these cases, we skip using any codons for mapping and concentrate on reading codons for non-terminals having more than one production, like in Rules 4,5,9 and 10 . We begin by decoding the first codon, i.e. `352`, by Eq. 1. The result, i.e `352 modulo 4 = 0`, is used to determine which production is chosen to replace the leftmost non-terminal (`ClassExpression1`) from its relevant rule (Rule 4). In this case, the leftmost `ClassExpression1` will be replaced by the value of `Class`. Next, the next codon will determine the production rule for the leftmost `Class` and `dbo:Plant` is selected by the value from 265 mod 4 =1. The mapping goes on like this until eventually there is no non-terminal left in the expression. Not all codons were required and extra codons have been simply ignored in this case.

3.3 Evaluation Framework

We follow the evaluation framework based on possibility theory, presented in [10] (see [14] for the theoretical background) 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, but we refer the interested reader to [10, 14] for an in-depth treatment.

Possibility theory [15] is a mathematical theory of epistemic uncertainty. Given a finite universe of discourse Ω, whose elements $\omega \in \Omega$ may be regarded as events, values of a variable, possible worlds, or states of affairs, a possibility distribution is a mapping $\pi : \Omega \to [0, 1]$, which assigns to each ω a degree

of possibility ranging from 0 (impossible, excluded) to 1 (completely possible, normal). A possibility distribution π for which there exists a completely possible state of affairs ($\exists\omega \in \Omega : \pi(\omega) = 1$) is said to be *normalized*.

A possibility distribution π induces a *possibility measure* and its dual *necessity measure*, denoted by Π and N respectively. Both measures apply to a set $A \subseteq \Omega$ (or to a formula ϕ, by way of the set of its models, $A = \{\omega : \omega \models \phi\}$), and are usually defined as follows:

$$\Pi(A) = \max_{\omega \in A} \pi(\omega); \tag{2}$$

$$N(A) = 1 - \Pi(\bar{A}) = \min_{\omega \in \bar{A}} \{1 - \pi(\omega)\}. \tag{3}$$

In other words, the possibility measure of A corresponds to the greatest of the possibilities associated to its elements; conversely, the necessity measure of A is equivalent to the impossibility of its complement $\bar{A}$. A generalization of the above definition can be obtained by replacing the min and the max operators with any dual pair of triangular norm and co-norm.

Given incomplete knowledge like RDF datasets, where there exist some missing and erroneous facts (instances) as a result of the heterogeneous and collaborative character of the LOD, adopting an axiom scoring heuristic based on possibility theory is a well-suited approach. Accordingly, a candidate axiom ϕ is viewed as a hypothesis that has to be tested against the evidence contained in an RDF dataset. Its *content* is defined as a finite set of logical consequences

$$\text{content}(\phi) = \{\psi : \phi \models \psi\}, \tag{4}$$

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 content(ϕ). 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 [10], 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, all the more so because (and this is a very important point), in view of the open world assumption, for two classes that do not share any instance, disjointness *can only be hypothetical* (i.e., fully possible, if not contradicted by facts, but *never* necessary).

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 axiom, i.e., u_ϕ. However, in case one of the components of an axiom is not supported by any fact, its generality should 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],

$$f(\phi) = u_\phi \cdot \frac{\mathit{\Pi}(\phi) + N(\phi)}{2}, \tag{5}$$

we resorted to the following improved definition, proposed in [10]:

$$f(\phi) = g_\phi \cdot \mathit{\Pi}(\phi). \tag{6}$$

The fitness value of a class disjointness axiom $\mathsf{DisjointClasses}(C, D)$ (or $\mathsf{Dis}(C, D)$ in Description Logic notation) is measured by defining the numbers of counterexamples and the support. These values are counted by executing the corresponding SPARQL queries based on *graph patterns*, via an accessible SPARQL endpoint. Each SPARQL graph pattern here is a mapping $Q(E, \texttt{?x}, \texttt{?y})$ translated from the coressponding OWL expression in axiom where E is an OWL expression, `x` and `y` are variables such that the query `SELECT DISTINCT ?x ?y WHERE` $\{Q(E, \texttt{?x}, \texttt{?y})\}$ returns all individuals that are instances of E.

The definition of $Q(E, \texttt{?x}, \texttt{?y})$ is based on different kinds of OWL expressions.

- E is an atomic expression.
  - For an atomic class A,

$$Q(A, \texttt{?x}, \texttt{?y}) = \texttt{?x} \quad \texttt{rdf:type} \quad A. \tag{7}$$

    where A is a valid IRI.
  - For a simple relation R,

$$Q(R, \texttt{?x}, \texttt{?y}) = \texttt{?x} \quad R \quad \texttt{?y}. \tag{8}$$

    where R is a valid IRI.
- E is a complex expression. We only focus on the case of complex class expressions involving relational operators, i.e., intersection, existential quantification and value restriction and skip complex relation expressions, i.e., we only allow simple relations in the expressions. In this case, Q can be inductively extended to complex expressions:
  - if $E = C_1 \sqcap \ldots \sqcap C_n$ is an intersection of classes,

$$Q(E, \texttt{?x}, \texttt{?y}) = Q(C_1, \texttt{?x}, \texttt{?y}) \ldots Q(C_n, \texttt{?x}, \texttt{?y}). \tag{9}$$

  - if E is an existential quantification of a class expression C,

$$Q(\exists R.C, \texttt{?x}, \texttt{?y}) = Q(R, \texttt{?x}, \texttt{?z1}) \quad Q(C, \texttt{?z1}, \texttt{?z2}) \tag{10}$$

    where R is a simple relation.

- if E is a value restriction of a class expression C,

$$
\begin{aligned}
Q(\forall R.C, \texttt{?x}, \texttt{?y}) = \{\ & Q(R, \texttt{?x}, \texttt{?z0}) \\
& \texttt{FILTER NOT EXISTS } \{ \\
& \quad Q(R, \texttt{?x}, \texttt{?z1}) \\
& \quad \texttt{FILTER NOT EXISTS } \{ \\
& \quad\quad Q(C, \texttt{?z1}, \texttt{?z2}) \\
& \} \ \} \ \} \ .
\end{aligned}
\tag{11}
$$

  where R is a simple relation.

The support $u_{\mathsf{Dis}(C,D)}$ can thus be computed with the following SPARQL query:

$$
\begin{aligned}
&\texttt{SELECT (count (DISTINCT ?x AS ?u)WHERE } \{Q(C, \texttt{?x}, \texttt{?y}) \\
&\quad \texttt{UNION } \ Q(D, \texttt{?x}, \texttt{?y})\}
\end{aligned}
\tag{12}
$$

To compute the generality $g_{\mathsf{Dis}(C,D)} = \min(u_C, u_D)$, u_C and u_D are required, which are returned by the following SPARQL queries:

$$
\texttt{SELECT (count (DISTINCT ?x) AS ?u\_C)WHERE } \{Q(C, \texttt{?x}, \texttt{?y})\} \tag{13}
$$

$$
\texttt{SELECT (count (DISTINCT ?x) AS ?u\_D)WHERE } \{Q(D, \texttt{?x}, \texttt{?y})\} \tag{14}
$$

Finally, we must figure out the number of counterexamples $u^{-}_{\mathsf{Dis}(C,D)}$. Counterexamples are individuals i such that $i \in [Q(\texttt{C}, \texttt{?x}, \texttt{?y})]$ and $i \in [Q(\texttt{D}, \texttt{?x}, \texttt{?y})]$; this may be translated into a SPARQL query to compute $u^{-}_{\mathsf{Dis}(C,D)}$:

$$
\begin{aligned}
&\texttt{SELECT (count (DISTINCT ?x) AS ?counters)} \\
&\texttt{WHERE } \{Q(\texttt{C}, \texttt{?x}, \texttt{?y}) \ \ Q(\texttt{D}, \texttt{?x}, \texttt{?y})\}
\end{aligned}
\tag{15}
$$

4 Experimental Setup

The experiments are divided into two phases: (1) mining class disjointness axioms with the GE framework introduced in Section 3 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.

4.1 Training Dataset Preparation

We randomly collect 1% of the RDF triples from DBpedia 2015-04 (English version), which contains 665,532,306 RDF triples, to create the *Training Dataset (TD)*.[2] 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

[2] Available for download at `http://bit.ly/2OtFqHp`.

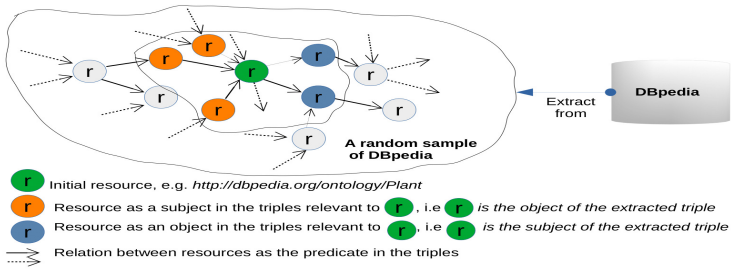

Fig. 1: An illustration of the Training Dataset sampling procedure

objects. An illustration of this mechanism to extract the sample training dataset is provided 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 their subject, of the form $\langle r\ p\ r' \rangle$, and 1% of the triples having r as their 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 (according to 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.

4.2 Parameters

We use the BNF grammar introduced in Section 3.1. Given how the grammar was constructed, the mapping of any chromosome of length ≥ 6 will always be successful. Hence, we can set $maxWrap = 0$. We ran our algorithm in 20 different runs on different parameter settings. 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 1.

4.3 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.[3]

[3] `https://virtuoso.openlinksw.com/`

5 Results & Discussions

We ran the GE method 20 times with the parameters shown in Table 1 on the BNF grammar defined in Section 3.1. Full results are available online.[4]

The number of valid distinct axioms, i.e., axioms ϕ such that $\Pi(\phi) > 0$ and $g_\phi > 0$, discovered is listed in Table 2. For measuring the accuracy of our results,

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

Table 1: Parameter values for GE.

k \ popSize	1000	2000	5000	10000
100000	8806	11389	4684	4788
200000	6204	13670	10632	9335
300000	5436	10541	53021	14590
400000	5085	9080	35102	21670

Table 2: Number of valid distinct axioms discovered over 20 runs.

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 *precision*, based on the most widely used definition of fuzzy set cardinality, introduced in [16] as follows: given a fuzzy set F defined on a countable universe set Δ,

$$\|F\| = \sum_{x \in \Delta} F(x), \tag{16}$$

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

$$\text{precision} = \frac{\|\text{true positives}\|}{\|\text{discovered axioms}\|} = \frac{\sum_\phi \Pi_{\text{DBpedia}}(\phi)}{\sum_\phi \Pi_{\text{TD}}(\phi)}, \tag{17}$$

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

The results in Table 3 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). According the results, we have statistically compared the performance of using different settings of *popSize* and k. The best setting $\{popSize = 5,000; k = 300,000\}$ allows the method to discover 53,021 distinct valid axioms with very high accuracy, i.e., $precision = 0.993 \pm 0.007$. Indeed, the plot in Fig. 2 illustrating the distribution of axioms in terms of possibility and generality shows that most discovered axioms with this setting are highly possible ($\Pi(\phi) > \frac{2}{3}$).

[4] `http://bit.ly/32YEQH1`

In order to obtain a more objective evaluation, we analyze in detail the axioms discovered by the algorithm with this best setting. First, we observe that together with the mandatory class expression containing the $\forall$ or $\exists$ operator, 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. We also analyse axioms containing complex expressions in both their members. These axioms are less general, even though they are completely possible. An example is the case with DisjointClasses(ObjectAllValuesFrom(dbprop:author dbo:Place) ObjectAllValuesFrom(dbprop:placeofBurial dbo:Place))($\Pi(\phi) = 1.0$; $g_\phi = 4$), which states that"*what can only be buried in a place cannot be what can only have a place as its author*".

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

Table 3: Average precision per run ($\pm$***std***)

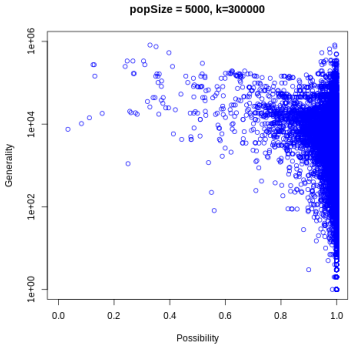

Fig. 2: Possibility and generality distribution of the discovered axioms

We also observe that some discovered axioms have a particularly high generality, as it is the case with DisjointClasses(dbo:Writer ObjectAllValuesFrom(dbo:writer dbo:Agent)) ($\Pi(\phi) = 0.982$; $g_\phi = 79,464$). This can be explained by the existence of classes supported by a huge number of instances (like dbo:Agent or dbo:Writer). From it, we might say that it is quite possible that "*writers are never written by agents*". Another similar case is axiom DisjointClasses(dbo:Journalist ObjectAllValuesFrom(dbo:distributor dbo:Agent)) ($\Pi(\phi) = 0.992$; $g_\phi = 32,533$) whereby in general "*journalists are not distributed by agents*", although it would appear that some journalists are, since $\Pi(\phi) < 1$!

Finally, we analyze an example of a completely possible and highly general axiom, DisjointClasses(dbo:Stadium ObjectAllValuesFrom(dbo:birthPlace dbo:Place)) ($\Pi(\phi) = 1.0$; $g_\phi = 10,245$), which we can paraphrase as "*stadiums cannot have a place as their birthplace*". Knowing that Stadium and Place are *not* disjoint, this axiom states that Stadium and $\forall$birthPlace.Place are in fact disjoint; in addition, $\forall$.birthPlace.Place, i.e., "(people) whose birthplace is a place" is a class with many instances, hence the high generality of the axiom.

6 Related Work

Some prominent works are introduced in Section 1 and are also analysed in [9, 10]. In this paper, we only focus on recent contributions relevant to class disjointness discovery. For instance, Reynaud *et al.* [7] 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 [8] 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, 10], a heuristic method by using *Grammatical Evolution (GE)* is applied 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.[5] Also, complex axioms are defined with the help of relational

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

operators of intersection and union, which can also be mechanically derived from the known atomic axioms.

7 CONCLUSION

We have proposed an extension of a grammar-based GP method for mining disjointness axioms involving complex class expressions. These expression consist of the relational operators of existential quantification and value restriction. The use of a training-testing model allows to objectively validate the method, while also alleviating the computational bottleneck of SPARQL endpoints. We analyzed some examples of discovered axioms. The experimental results confirm that the proposed method is capable of discovering highly accurate and general axioms.

In the future, we will focus on mining disjointness axioms involving further types of complex classes, by bringing into the picture other relational operators such as `owl:hasValue` and `owl:OneOf`. We might also forbid the occurrence of atomic classes at the root of class expressions. We also plan on refining the evaluation of candidate axioms with the inclusion of some measurement of their complexity in the fitness function.

Acknowledgments

This work has been 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. Guarino, N., Oberle, D., Staab, S.: What Is an Ontology? Handbook on Ontologies. International Handbooks on Information Systems, Springer (2009)
2. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum.-Comput. Stud. 43(5-6), 907–928 (1995)
3. Zhu, M.: DC proposal: Ontology learning from noisy linked data. In: International Semantic Web Conference (2). Lecture Notes in Computer Science, vol. 7032, pp. 373–380. Springer (2011)
4. Völker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning disjointness. In: ESWC. Lecture Notes in Computer Science, vol. 4519, pp. 175–189. Springer (2007)
5. Völker, J., Fleischhacker, D., Stuckenschmidt, H.: Automatic acquisition of class disjointness. J. Web Sem. 35, 124–139 (2015)
6. Lehmann, J.: Dl-learner: Learning concepts in description logics. Journal of Machine Learning Research 10, 2639–2642 (2009)
7. Reynaud, J., Toussaint, Y., Napoli, A.: Redescription mining for learning definitions and disjointness axioms in linked open data. In: ICCS. Lecture Notes in Computer Science, vol. 11530, pp. 175–189. Springer (2019)
8. Rizzo, G., d'Amato, C., Fanizzi, N., Esposito, F.: Terminological cluster trees for disjointness axiom discovery. In: ESWC (1). Lecture Notes in Computer Science, vol. 10249, pp. 184–201 (2017)

9. Nguyen, T.H., Tettamanzi, A.G.B.: Learning class disjointness axioms using grammatical evolution. In: EuroGP. Lecture Notes in Computer Science, vol. 11451, pp. 278–294. Springer (2019)
10. Nguyen, T.H., Tettamanzi, A.G.B.: An evolutionary approach to class disjointness axiom discovery. In: WI. pp. 68–75. ACM (2019)
11. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
12. Vanneschi, L., Poli, R.: Genetic Programming — Introduction, Applications, Theory and Open Issues. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
13. O'Neill, M., Ryan, C.: Grammatical evolution. Trans. Evol. Comp 5(4), 349–358 (Aug 2001), http://dx.doi.org/10.1109/4235.942529
14. Tettamanzi, A.G.B., Faron-Zucker, C., Gandon, F.: Possibilistic testing of OWL axioms against RDF data. Int. J. Approx. Reasoning 91, 114–130 (2017)
15. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1, 3–28 (1978)
16. De Luca, A., Termini, S.: A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory. Information and Control 20, 301–312 (1972)