An Evolutionary Approach to Class Disjointness Axiom Discovery

Thu Huong Nguyen, Andrea G. B. Tettamanzi

An Evolutionary Approach to Class Disjointness Axiom Discovery

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

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

ABSTRACT

Axiom learning is an essential task in enhancing the quality of an ontology, a task that sometimes goes under the name of ontology enrichment. To overcome some limitations of recent work and to contribute to the growing library of ontology learning algorithms, we propose an evolutionary approach to automatically discover axioms from the abundant RDF data resource of the Semantic Web. We describe a method applying an instance of an Evolutionary Algorithm, namely Grammatical Evolution, to the acquisition of OWL class disjointness axioms, one important type of OWL axioms which makes it possible to detect logical inconsistencies and infer implicit information from a knowledge base. The proposed method uses an axiom scoring function based on possibility theory and is evaluated against a *Gold Standard*, manually constructed by knowledge engineers. Experimental results show that the given method possesses high accuracy and good coverage.

CCS CONCEPTS

• **Computing methodologies** → **Ontology engineering**; **Machine learning algorithms**; **Instance-based learning**; **Evolutionary algorithms**;

KEYWORDS

Grammatical Evolution, Ontology Learning, OWL Axioms

ACM Reference Format:
Thu Huong Nguyen and Andrea G.B. Tettamanzi. 2019. An Evolutionary Approach to Class Disjointness Axiom Discovery. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI '19), October 14–17, 2019, Thessaloniki, Greece.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3350546.3352502

1 INTRODUCTION

An ontology [6, 7, 12] may be viewed as a formal representation of a shared domain knowledge. It can be defined by 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. Ontologies provide a major support to modeling and sharing knowledge among various applications in a specific domain. However, ontology construction is limited by the obstacle known as "knowledge acquisition bottleneck". This can arise from the requirement of involving domain experts and knowledge engineers, which is highly expensive and time-consuming. This problem may be tackled by the set of methods and techniques that go under the name of ontology learning [8, 12, 13, 19]. These methods, by adopting learning algorithms from several existing knowledge and information sources, can help alleviate the overall cost of ontology construction by reducing or eliminating altogether the need of domain experts. Ontology learning may be viewed as a special case of knowledge discovery from data (KDD) or data mining, where the data are in a special format (assertions) and knowledge takes the form of axioms of an ontology. At the starting point of ontology learning, two fundamental aspects have to be defined: the types of input data sources from which the system exploits ontological knowledge and the ontological elements which need to be learned.

The growth of the Web of data, also called the Semantic Web, where the Linked Open Data (LOD) is a prominent representative opens up exciting opportunities for learning new knowledge in the context of an open world. The LOD can be considered as a huge real-world knowledge base for learning ontologies. Specifically, ontology learning on the Semantic Web involves handling the enormous and diverse amount of data in the Web and thus enhancing existing approaches for knowledge acquisition instead of only focusing on mostly small and uniform data collections.

The elements of the ontology to be learned, usually axioms formalized in the form of logical assertions, can then be used to enhance and constrain the information contained in the ontology and to check its correctness or deduce new information [1, 19]. As a consequence, learning axioms is a crucial task in ontology learning to discover implicit axioms from existing ontologies or instance data, i.e. prior knowledge or knowledge base (KB). In particular, class disjointness axioms are useful for checking the logical consistency and detecting undesired usage patterns or incorrect assertions. As for the definition of disjointness [21], two classes are disjoint if they do not possess any common individual according to their intended interpretation, i.e., the intersection of these classes is empty in a particular KB. A simple example can demonstrate the potential advantages obtained by the addition of this kind of axioms to an ontology. A knowledge base defining terms of classes like *Mother, Man* and asserting that individual *Tyler* is both a *Mother* and a *Man* would be

WI '19, October 14–17, 2019, Thessaloniki, Greece

logically consistent, without any errors being recognized by a reasoner. However, if a constraint of disjointness between classes *Mother* and *Man* is added, the reasoner will be able to reveal an error in the modeling of such a knowledge base. As a consequence, logical inconsistencies of facts can be detected and excluded—thus enhancing the quality of ontologies. As a matter of fact, very few *DisjointClasses* axioms are currently found in existing ontologies. For example, in the DBpedia ontology, the query `SELECT ?x ?y { ?x owl:disjointWith ?y }` executed on May 11, 2019 returned only 25 solutions, whereas the realistic number of class disjointness axioms generated from hundreds of classes in DBpedia (738 classes in DBpedia version 2015-04, 760 classes in DBpedia version 2016-04)[1] where `?subject a owl:Class.` is expected to be much larger, in the order of the thousands or tens of thousands. Hence, learning implicit knowledge in terms of axioms from a LOD repository in the context of the Semantic Web has been the object of research in several different approaches.

Among the early approaches to ontology development, we summarize top-down and bottom-up techniques. In the case of axiom learning, i.e., learning class disjointness axioms, recent methods [11, 22] apply top-down or *intensional* approaches to learning disjointness which relies on schema-level information, i.e., logical and lexical descriptions of the classes. The contributions based on bottom-up or *extensional* approaches [1, 21], on the other hand, require the instances in the dataset to induce instance-driven patterns to suggest axioms, e.g., disjointness class axioms.

Along the lines of extensional (i.e., instance-based) methods, we propose an evolutionary approach, based on grammatical evolution, for mining implicit axioms from RDF datasets. The goal is to derive potential class disjointness axioms of more complex types, 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. Also, an evaluation method based on possibility theory is adopted to assess the certainty level of induced axioms.

The rest of the paper is organized as follows. Section 2 describes briefly some related works. In Section 3, some foundations are introduced. Section 4 sketches a GE approach for learning OWL classes disjointness axioms. An axiom evaluation method based on possibility theory is also presented in this section. Section 5 provides experimental evaluation and comparison. Section 6 concludes the paper with the final remarks and the directions for future research.

2 RELATED WORK

The most prominent related work relevant to learning disjointness axioms consists of the contributions by Johanna Völker and her collaborators [5, 21, 22]. In early work, Völker developed supervised classifiers from LOD incorporated in the *LeDA* tool [22]. However, the learning algorithms need a set of labeled data for training that may demand expensive work by domain experts. In contrast to *LeDA*, statistical schema induction via association rule mining [21] was given in the tool *GoldMiner*, where association rules are representations of implicit patterns extracted from large amount of data and no training data is required. Association rules are compiled based on statistical analysis of a transaction table, which is built from the results of SPARQL queries. That research only focused on generating axioms involving *atomic* classes, i.e., classes that do not consist of logical expressions, but only of a single class identifier.

Another relevant research is the one by Lorenz Bühmann and Jens Lehmann, whose proposed methodology is implemented in the DL-Learner system [11] for learning general class descriptions (including disjointness) from training data. Their work relies on the capabilities of a reasoning component, but suffers from scalability problems for the application to large datasets like LOD. In [1], they tried to overcome these obstacles by obtaining predefined data queries, i.e., SPARQL queries to detect specific axioms hidden within relevant data in datasets for the purpose of ontology enrichment. Bühmann and Lehmann also developed methods for generating more complex axiom types [2] by using frequent terminological axiom patterns from several data repositories. One important limitation of their method is the time-consuming and computationally expensive process of learning frequent axioms patterns and converting them into SPARQL queries before generating actual axioms from instance data. Also, the most frequent patterns refer to inclusion and equivalence axioms like $A \equiv B \sqcap \exists r.C$ or $A \sqsubseteq B \sqcap \exists r.C$.

Here, we extend and enhance our recent proposal [14] of an approach to generate class disjointness axioms from an existing RDF repository using Grammatical Evolution (GE). On the one hand, the enhancement concerns the fitness function used to score axioms, where we now include an improved measure of generality and we removed the necessity measure, which, as we will explain below, does not carry any useful information when dealing with this type of axioms. On the other hand, we greatly extend the experimental validation, by systematically exploring a variety of parameter settings. Along the lines of extensional (i.e., instance-based) methods, we propose a number of improvements as well as more extensive experiments for mining implicit axioms from RDF datasets in an evolutionary approach, based on grammatical evolution to investigate its potential and suggest future extensions. The goal is to derive potential class disjointness axioms of more complex types, i.e., defined with the help of relational operators of intersection and union.

3 FOUNDATIONS

3.1 The Semantic Web and Related Concepts

The Semantic Web[2] (SW) which is an extension of the World Wide Web (WWW) can be considered as the movement

[1] These figures can be obtained by executing the querry " `SELECT (COUNT( DISTINCT ?subject) as ?numberClasses`" on `http://dbpedia.org/sparql`

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

from the Web of documents to the Web of data. In this, semantic information containing the machine-processable information called ***metadata***—a fundamental component of the SW—is embedded within Web content. Among the metadata, IRIs (International Resource Identifiers) are used to identify abstract or physical resources.

The **Resource Description Framework (RDF)**[3] [17] is mainly a data model of SW for describing machine processable semantics of data. RDF uses as statements triples of the form <**Subject**, **Predicate**, **Object**>. Each part of the triples can be described in the form of IRIs and in the shorter representation associated with the *prefix aliases*[4]. The query language for RDF is ***SPARQL***.[5]

Linked Open Data[6] (**LOD**) is an association of Linked Data [7] 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**,[8] which comprises a rather rich collection of facts extracted from the Wikipedia.

OWL[9] (**Web Ontology Language**) is one of the data modeling languages for describing RDF data. OWL is much more expressive not only to describe classes and properties but also to use in data modeling and reasoning. OWL contains the constructors of complex class descriptions such as *owl:UnionOf*, *owl:IntersectionOf*, *owl:Complementary* and express relations between class descriptions by means of class axioms such as *owl:disjointWith*.

3.2 Grammatical Evolution

GE is a grammar-based form of GP [9, 10]. It is a relatively new evolutionary computation technique pioneered by Michael O'Neill and his collaborators [15, 18]. By the tradition of evolutionary algorithm (EA), a population of individuals, i.e., potential solutions, is maintained by the GE algorithm and iteratively refined to find the best solution. At each iteration, known as a *generation*, the fitness of individuals in the population are evaluated. According to these levels of fitness, the process of selecting individuals called *parents selection* is performed to create a list of better qualified parents as input for generating a new set of candidate solutions, i.e. offspring, in the next generation. The offspring of each generation is bred by applying *genetic operators* on the selected parents.

As in [4] and unlike GP, GE has a clear distinction in representation between the search space and the solution space. Programs known as *phenotypic solutions* or *phenotypes* are formed from the variable length binary strings, i.e., *genotypic individuals* or *genotypes* through the transformation called *mapping process*. Among them, the variable length binary string genomes, i.e., *chromosomes*, are used in the form of consecutive groups of 8 bits called *codon* representing an integer value. Meanwhile, the genotype-phenotype mapping process requires a set of production rules expressed in the form of formal grammar, namely a *Backus-Naur form (BNF) grammar* which specifies the syntax of desired programs. *BNF grammar* is a context-free grammar consisting of terminals and non-terminals. A grammar can be represented in the form of the 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. In the mapping process, codons are used consecutively to choose production rules from P in the BNF grammar according to the function:

$$\text{Rule option} = \text{codon } \textbf{modulo} \begin{array}{c}\text{Number of rules}\\ \text{for the current}\\ \text{non-terminal}\end{array} \tag{1}$$

Codons can be reused two or more times (called *wrapping* technique) [4, 15]. In the advantageous cases, programs are generated before the end of the genome is reached; otherwise, a wrapping operator is applied and the reading of codons will continue from the beginning of the chromosome, until the maximum allowed number of wrapping events is reached. An unsuccessful mapping will happen if the threshold on the number of wrapping events is reached but the individual is still not completely mapped; in this case, the individual is assigned the lowest possible fitness.

Although GE uses the standard genetic operators of crossover and mutation in the Evolutionary Algorithms (EA)[3, 16], GE applies these operators on variable-length binary strings, i.e., genotypes, instead of the actual programs, i.e., phenotypes. In this, a standard single point crossover is employed whereby two crossover points are chosen randomly, one on each individual and the two sets of codons following these points exchanged, i.e. the right part of each individual are swapped. Also, a selected individual goes through with single point mutation, i.e. a codon selected at random, and this codon is replaced with a new randomly generated codon.

3.3 Possibility Theory

Possibility theory [23] is a mathematical theory of epistemic uncertainty. Given a finite universe of discourse Ω, whose elements $\omega \in \Omega$ may be regarded as outcomes of an experiment, 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

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

are usually defined as follows:

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

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

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.

4 METHOD FOR LEARNING CLASS DISJOINTNESS AXIOMS

We apply the GE approach introduced in Sec. 3.2 to learn class disjointness axioms from instance data acquired from an RDF repository, e.g., DBpedia. The workflow of such GE approach is shown in Fig. 1. In the context of axiom discovery, the definition of "programs" or "phenotypic solutions" in GE are class disjointness axioms whose syntax is described by a BNF grammar. Therefore, the first important task is to design the grammar used for generating well-formed class disjointness axioms. Afterwards, in the beginning of the evolutionary process, we randomly generate a population of axioms in genotypic representation, i.e., candidate genotypic axioms, which are encoded as variable-length integer strings, i.e., numerical chromosomes. According to the built grammar and the principles of the mapping process, we are able to decode these chromosomes into class disjointness axioms in the form of logical expressions, i.e., candidate phenotypic axioms. The set of candidate axioms is maintained by the GE algorithm and iteratively refined to find axioms that are both general and credible (two key quality measures for discovered knowledge). The quality of the generated axioms can be improved gradually during the evolutionary process by applying genetic operators (crossover and mutation) on genotypic axioms. In this section, we first briefly describe the process of grammar construction and give a specific example of the decoding phase to well-formed class disjointness axioms. After that, possibilistic evaluation of generated axioms is presented in detailed.

4.1 Structure of the BNF Grammar

We use the functional-style grammar in the extended BNF notation used by the W3C to design the grammar for generating OWL class disjointness axioms[10]. The production rules are adapted from the complete normative grammar of OWL[11]. The noteworthy thing is that the use of a BNF grammar here does not focus on defining what a well-formed axiom may be, but on generating well-formed axioms which may express the facts contained in a given RDF triple store. Hence, only resources of the language that actually occur in the RDF dataset should be generated. The BNF grammar is organized in two main parts (namely static and dynamic) as follows:

- The static part contains production rules defining the structure of the axioms, loaded from a hand-crafted text file. Different grammars will generate different kinds of axioms.
- The dynamic part contains production rules for the low-level non-terminals, which we can call the *primitives*. These production rules are automatically built at runtime by querying the SPARQL endpoint of the RDF repository at hand.

This approach to organizing the structure of a BNF grammar ensures that changes in the contents of RDF repositories will not require the grammar to be rewritten.

In the functional-style syntax of OWL[12], class disjointness axioms have the form DisjointClasses$(C_1, C_2, ..., C_n)$. Without loss of generality, we focus on generating binary axioms such as DisjointClasses(C_1, C_2), where C_1 and C_2 can be atomic expressions like in DisjointClasses(dbo : Film, dbo : WrittenWork) or complex expressions involving relational operators like intersection and union, like in DisjointClasses(dbo : Film, ObjectIntersectionOf (dbo : Book, ObjectUnionOf (dbo : Comics, dbo : MusicalWork)))[13].

The pattern of the grammar structured for generating class disjointness axioms is built as follows:

% Static part

```
(r1) Axiom := ClassAxiom
(r2) ClassAxiom := DisjointClasses
(r3) DisjointClasses := 'DisjointClasses' '(' ClassExpression ' 'ClassExpression ')'
(r4) ClassExpression :=   Class                 (0)
                        | ObjectUnionOf         (1)
                        | ObjectIntersectionOf  (2)
(r5) ObjectUnionOf := 'ObjectUnionOf' '(' ClassExpression ' ' ClassExpression ')'
(r6) ObjectIntersectionOf := 'ObjectIntersectionOf' '(' ClassExpression ' ' Class-
                             Expression ')'
```

% Dynamic part - Primitives

```
(r.7) Class := % production rules are constructed by using SPARQL queries
```

The production rule of the primitive `Class` will be filled by using a SPARQL query of the form

```
SELECT ?class WHERE { ?instance a ?class. }
```

to extract classes (represented by their IRI) from the RDF dataset.

An example representing a small excerpt of an RDF triple repository is the following:

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

dbr:Quiet_City_(film)     rdf:type      dbo:Film.
dbr:Cantata               rdf:type      dbo:MusicalWork.
dbr:The_Times             rdf:type      dbo:WrittenWork.
dbr:The_Hobbit            rdf:type      dbo:Book.
dbr:Fright_Night_(comics) rdf:type      dbo:Comic
```

and options for the *Class* non-terminal are represented as follows:

```
(r.7) Class := dbo:Film            (0)
           |  dbo:MusicalWork      (1)
           |  dbo:WrittenWork      (2)
           |  dbo:Book             (3)
           |  dbo:Comic            (4)
```

[10] https://www.w3.org/TR/owl2-syntax/#Disjoint_Classes
[11] https://www.w3.org/TR/owl2-syntax/
[12] https://www.w3.org/TR/owl2-syntax/#Functional-Style_Syntax
[13] PREFIX dbo: http://dbpedia.org/ontology/

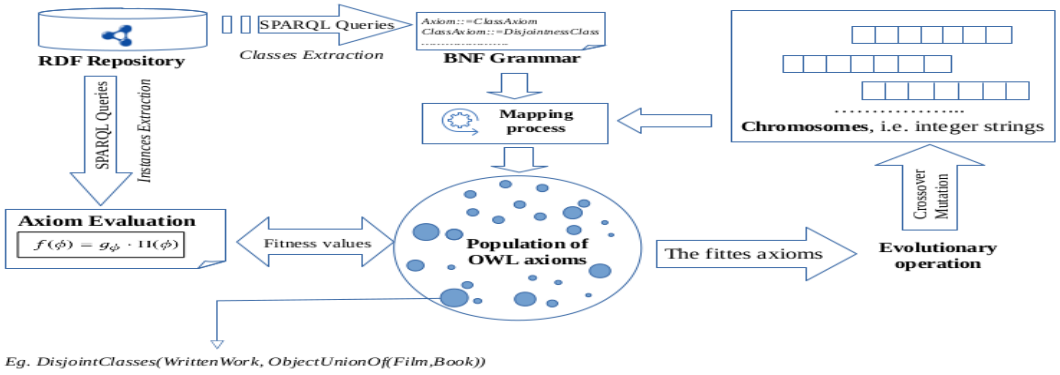


Figure 1: Workflow of class disjointness axioms learning from an RDF repository using GE.

4.2 Mapping Process

In this section, we illustrate the transformation of an integer chromosome into an OWL class disjointness axiom in functional-style syntax through an example. Let the chromosome be (253, 213, 397, 387, 268, 342, 321, 408, 182, 132) and let the BNF grammar and the RDF store be as in Sec. 4.1. We apply Eq. 1 in Sec. 3.2 to choose production rules from the grammar. Table 1 illustrates the steps of the mapping to a class disjointness axiom expression relevant to the considered example.

There is only one production for non-terminals `Axiom`, `ClassAxiom`, `DisjointClasses`, `ObjectIntersectionOf`, and `ObjectUnionOf` as it can be seen from Rules 1–3, 5, and 6. 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 and 7. We begin by decoding the first codon, i.e. `253`, by Eq. 1. The result, i.e `253 modulo 3 = 1`, is used to determine which production is chosen to replace the leftmost non-terminal (`ClassExpression`) from its relevant rule (Rule 4). In this case, the leftmost `ClassExpression` will be replaced by the value of `ObjectUnionOf`. 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.

4.3 Axiom Evaluation

Some facts (instances) in the RDF repository may be missing or erroneous as a result of the heterogeneous and collaborative character of the LOD. This incompleteness and noise determines a sort of *epistemic* uncertainty in the evaluation of the quality of a candidate axiom. In order to properly capture this type of uncertainty, typical of an open world, which contrasts with the *ontic* uncertainty typical of random processes, we follow [14] in adopting an axiom scoring heuristics based on possibility theory (cf. Section 3.3), which is well-suited to incomplete knowledge. This appears to be a justified choice for assessing knowledge extracted from an RDF repository. We now summarize this scoring heuristics which was applied in [14] and we point out how our definition of the fitness function differs from [14].

In the case of possibilistic axiom scoring, the basic principle for establishing the possibility of a formula ϕ should be that the absence of counterexamples to ϕ in the RDF repository means $\Pi([\phi]) = 1$, i.e., that ϕ is completely possible. Let ϕ be an axiom that we wish to evaluate (i.e., a theory). The *content* of an axiom ϕ that we wish to evaluate 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 formula $\psi \in \text{content}(\phi)$ may be readily tested by means of a SPARQL `ASK` query. Let us define $u_\phi = \|\text{content}(\phi)\|$ as the *support* of ϕ. Let then u_ϕ^+ be the number of confirmations (basic statements ψ that are satisfied by the RDF repository) and u_ϕ^- the number of counterexamples (basic statements ψ that are falsified by the RDF repository).

The possibility measure $\Pi(\phi)$ and the necessity measure $N(\phi)$ of an axiom have been defined as follows in [20]: for $u_\phi > 0$,

$$\Pi(\phi) = 1 - \sqrt{1 - \left(\frac{u_\phi - u_\phi^-}{u_\phi}\right)^2}; \tag{5}$$

$$N(\phi) = \sqrt{1 - \left(\frac{u_\phi - u_\phi^+}{u_\phi}\right)^2}, \quad \text{if } \Pi(\phi) = 1, \text{ 0 otherwise} \tag{6}$$

The cardinality of the sets of the facts in the RDF repository reflects the support of each axiom. An axiom is all the more necessary as it is explicitly supported by facts, i.e., confirmations, and not contradicted by any fact, i.e., counterexamples, while it is the more possible as it is not contradicted by any fact. In principle, the fitness of axiom ϕ should be directly proportional to its necessity $N(\phi)$ and to its possibility $\Pi(\phi)$.

In addition, what we are looking for is not only credible axioms, but also general ones. As suggested in [14], the generality of an axiom may be defined as the cardinality of its support, u_ϕ. In other words, an axiom is more general as the extension of its components contains more facts. In case one of the components of an axiom is not supported by any fact, its generality will be zero.

Table 1: An illustration of mapping process to an expression of class disjointness axiom

Codon	Rule	Option	Mapped Expression
-	(r.1)	-	ClassAxiom
-	(r.2)	-	DisjointClasses
-	(r.3)	-	'DisjointClasses''('ClassExpression' ' ClassExpression')'
253	(r.4)	(1)	'DisjointClasses' (ObjectUnionOf' 'ClassExpression')'
-	(r.5)	-	'DisjointClasses''(''ObjectUnionOf''('ClassExpression' 'ClassExpression')'ClassExpression')'
213	(r.4)	(0)	'DisjointClasses''(''ObjectUnionOf''('Class' 'ClassExpression')' 'ClassExpression')'
397	(r.7)	(2)	'DisjointClasses'"('ObjectUnionOf''('dbo:WrittenWork' 'ClassExpression')' 'ClassExpression')'
387	(r.4)	(0)	'DisjointClasses''(''ObjectUnionOf''('dbo:WrittenWork' 'Class')'ClassExpression')'
268	(r.7)	(3)	'DisjointClasses''(''ObjectUnionOf''('dbo:WrittenWork' 'dbo:Book')'ClassExpression')'
342	(r.4)	(0)	'DisjointClasses''(''ObjectUnionOf''('dbo:WrittenWork' 'dbo:Book')'Class')'
321	(r.7)	(1)	'DisjointClasses''(''ObjectUnionOf''('dbo:WrittenWork' 'dbo:Book')'dbo:MusicalWork')'

A definition of the fitness function that satisfies all the above requirements is

$$f(\phi) = u_\phi \cdot \frac{\Pi(\phi) + N(\phi)}{2}, \tag{7}$$

which was adopted by [14].

However, it should be noticed that negation is not supported by the syntax of RDF. Negated assertions can of course be expressed using the vocabulary of OWL, but then the services of an OWL reasoner would have to be used to infer the negation of an assertion thus expressed; however, that would be way more expensive than using SPARQL to query the dataset and also of little use, since very few or no negated assertions at all do occur in real-world RDF datasets. As a result, an RDF dataset will naturally provide counterexamples for the disjointness axioms (e.g., an individual that is asserted to belong in the two supposedly disjoint classes). On the other hand, confirmations, which should take the form of negated assertions, like "such individual, which belongs to either of the supposedly disjoint classes, *does not* belong in the other", will be completely missing. The simple solution of taking the absence of a counterexample as a confirmation (i.e., letting $u_\phi^+ = u_\phi - u_\phi^-$) would betray the open-world hypothesis that underlies the SW and is, therefore, not satisfactory. However, this problem can be overcome by actually scoring axioms based on counterexamples only, which is, after all, much in agreement with the falsificationist approach that underlies the current practice in Science (to corroborate a hypothesis, one should not look for easy confirmations, but should rather try hard to find counterexamples). Since the number of confirmations u_ϕ^+ only appears in the definition of $N(\phi)$, we can safely drop $N(\phi)$ from Equation 7.

A second refinement of the definition of fitness stems from the observation that, for a disjointness axiom of the form $\mathsf{Dis}(C, D)$, a better measure of its generality would be given by the minimum of the cardinalities of the extensions of the two classes involved, C and D, in the RDF dataset, whereas $u_{\mathsf{Dis}(C,D)}$ is the cardinality of the extension of $C \sqcup D$. Let us denote by $[C]$ the extension of class C in the RDF dataset at hand: this is the set of instances of C returned by a SPARQL query of the form

$$\texttt{SELECT DISTINCT ?x WHERE \{ ?x a } C \texttt{ . \}}$$

Then we define the generality of axiom $\mathsf{Dis}(C, D)$ as

$$g_{\mathsf{Dis}(C,D)} = \min\{\|[C]\|, \|[D]\|\} \tag{8}$$

and use it instead of u_ϕ in Equation 7. This yields the following improved definition of the fitness function,

$$f(\phi) = g_\phi \cdot \Pi(\phi), \tag{9}$$

which is the one used in our method.

In order to measure the fitness of class disjointness axioms $\mathsf{Dis}(C, D)$, with C and D class expressions, the numbers of the confirmations, the counterexamples and the support are counted by executing the corresponding SPARQL queries via an accessible SPARQL endpoint. Such SPARQL queries are based on *graph patterns* that are a direct translation of the expression of the OWL2 axiom considered. A graph pattern here is a mapping $Q(\texttt{E}, \texttt{?x})$ from OWL2 expressions to SPARQL graph patterns where `E` is an OWL 2 class expression, `x` is a variable such that the query `SELECT DISTINCT ?x WHERE` $\{Q(\texttt{E}, \texttt{?x})\}$ returns all individuals that are instances of `E`. We define $Q(\texttt{E}, \texttt{?x})$ when

- `E` is an atomic class expression of an entity `A`

$$Q(\texttt{A}, \texttt{?x}) = \texttt{?x} \quad a \quad \texttt{A}. \tag{10}$$

  where `A` is a valid IRI.
- `E` is an complex class expression involving relational operators, i.e., intersection and union. In this case, `Q` can be inductively extended to complex expressions:
  – if `E` is an intersection of classes $C_i : C_1 \sqcap ... \sqcap C_n$,

$$Q(C_1 \sqcap ... \sqcap C_n, \texttt{?x}) = Q(C_1, \texttt{?x})...Q(C_n, \texttt{?x}) \tag{11}$$

  – if `E` is a union of classes $C_i : C_1 \sqcup ... \sqcup C_n$,

$$Q(C_1 \sqcup \ldots \sqcup C_n, \texttt{?x}, \texttt{?y}) = \{Q(C_1, \texttt{?x})\} \texttt{ UNION } \cdots \texttt{ UNION } \{Q(C_n, \texttt{?x})\}. \tag{12}$$

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

$$\begin{array}{l}\texttt{SELECT(count (DISTINCT ?x)} \;\; AS \;\; \texttt{?u)} \\ \quad\quad \texttt{WHERE } \{Q(\texttt{C}, \texttt{?x}) \;\; \texttt{UNION} \;\; Q(\texttt{D}, \texttt{?x})\}\end{array} \tag{13}$$

The generality $g_{\mathsf{Dis}(C,D)}$ can be obtained by computing the min value in the supports of axiom components g_C, g_D with the following SPARQL queries.

$$\begin{array}{l}\texttt{SELECT(count (DISTINCT ?x)} \;\; AS \;\; \texttt{?u\_C)WHERE } \{Q(\texttt{C}, \texttt{?x})\} \\ \texttt{SELECT(count (DISTINCT ?x)} \;\; AS \;\; \texttt{?u\_D)WHERE } \{Q(\texttt{D}, \texttt{?x})\}\end{array} \tag{14}$$

4.4 Gold Standard Construction

In order to evaluate the effectiveness of our method in discovering disjointness class axioms, we use a manually created benchmark of class disjointness axioms, which we call the *Gold Standard*.

The process of creating the *Gold Standard* was carried out by knowledge engineers and consisted of two phases illustrated in Fig. 2. In the first phase, the disjointness of the top-most classes to their siblings was assessed manually. Therefrom, two sibling classes being disjoint will automatically imply the disjointness of their corresponding pair of subclasses. This process was repeated in the same way on the next level of concepts. The second phase consisted in manually annotating the disjointness for the not yet assessed pairs of classes which did not belong to the cases given in the previous phase. The result of the completion of the *Gold Standard* is a matrix representing the disjointness evaluation between pairs of distinct atomic class expressions. We first constructed the (62×62) matrix[14] of class disjointness axioms relevant to the *Work* topic of DBpedia 2015-04. This matrix contains (0 and 1) values representing the disjointness evaluation between 3,844 pairs of classes relevant to the topic, with 1,891 pairs of distinct asymmetric classes.

In compliance with the *Gold Standard* thus constructed, we can measure the quality of class disjointness axioms involving both atomic and more complex types, i.e. involving the intersection and union operators. Algorithm 1 describes in detail how a complex axiom is assessed using the *Gold Standard*.

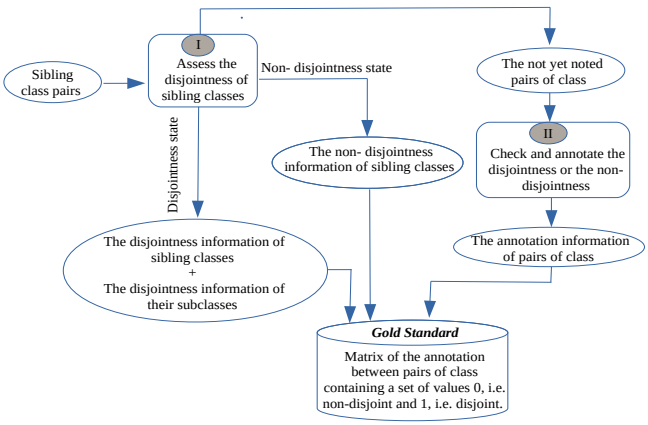


Figure 2: The process of *Gold Standard* creation

5 EXPERIMENTS & RESULTS

5.1 Experimental Protocol

We applied the proposed approach, introduced in Section 4, to mining class disjointness axioms from DBpedia, a large real-world RDF dataset.

The axioms involving both atomic and complex classes relevant to topic *Work* are systematically generated and evaluated on DBpedia version 2015-04 in English as the reference RDF fact repository. Of 62 classes about the *Work* topic in DBpedia 2015-04, 53 classes with 5,195,019 instances are relevant to our experiments. All data used in this experiment are represented by RDF triples, as in the examples of Section 4.1.

We use the BNF grammar of disjointness axioms shown in Section 4.1. However, to increase the success of the mapping to complex axiom expressions, we double the appearance probability of non-terminal `ClassExpression`. Rule (r4) in the grammar is thus modified to

```
(r4) ClassExpression   := Class                (0)
                        | Class                (1)
                        | ObjectUnionOf        (2)
                        | ObjectIntersectionOf (3)
```

Although the main purpose of our research is to focus on exploring complex disjointness axioms involving intersection and union, we also performed experiments to generate axioms involving atomic classes only, for comparison purpose. In that case, Rule (r4) is simplified to only one option `ClassExpression := Class` .

We also increase the length of chromosomes *initlenChrom* (in the case of complex axioms) to 30 instead of 20 in [14].

The algorithm parameters, summarized in Table 2, were empirically determined by performing a systematic exploration of a grid of possible settings. A prototype system of the proposed method was developed in Java, using Apache Jena to interface with SPARQL endpoints and GEVA v.2.0 [15], a Java implementation of GE. To avoid overloading DBpedia's SPARQL endpoint, we set up a local mirror[16] using the Virtuoso Universal Server.

All the experiments have been performed on a HP ZBook 15 G3 Mobile Workstation equipped with an eight-core Intel i7 CPU 6820HQ processor at 2.7GHz clock speed, with 32 GB RAM, 1 TB of disk space under the Ubuntu 16.04 LTS 64-bit operating system.

5.2 Results

In order to measure the effectiveness of the method, we ran it 3 times with the parameters shown in Table 2 on either version of the BNF grammar of axioms, involving atomic and complex expressions: the results are available on line.[17]. We also carried out a comparison of our results with [14] and *GoldMiner* [21]. Precision and recall are computed against a publicly available *Gold Standard*[18] of class disjointness axioms introduced in sec. 4.4.

The results, shown in Table 3, confirm that the accuracy and the coverage of our approach in extracting atomic axioms are higher than the results of [14] and *GoldMiner*. In terms of generating complex axioms, we witness a quite higher accuracy than in [14] and a superiority of our method compared with *GoldMiner*. In the recall comparison for the case of atomic axioms, we can also observe that the coverage of the set of generated atomic axiom in each run is much higher than the result in [14]. The recall value in *GoldMiner* is constant, namely 0.38, because that algorithm is deterministic,

[14] https://bitbucket.org/RDFMiner/disjointnessclassaxiomge/src/master/GoldStandard.tsv

[15] http://ncra.ucd.ie/Site/GEVA.html

[16] https://joernhees.de/blog/2015/11/23/setting-up-a-linked-data-mirror-from-rdf-dumps-dbpedia-2015-04-freebase-wikidata-linkedgeodata-with-virtuoso-7-2-1-and-docker-optional/

[17] https://bitbucket.org/RDFMiner/classdisjointnessaxioms/src/master/Results/

[18] https://bitbucket.org/RDFMiner/classdisjointnessaxioms/src/master/Results/GoldStanda

Algorithm1 - CheckDisjointClasses(classexpr1(n), classexpr2(m))

Input: ***classexpr1(n), classexpr2(m)***: class expressions being arguments in axiom,
n,m: the numbers of the classes contained in the class expressions;
G: matrix of Gold Standard
Output: R: Results of disjointness - returns to a non- negative integer value
if the return value is greater 0, ***classexpr1(n)*** and ***classexpr2(m)*** are disjoint
if the return value equals 0, ***classexpr1(n)*** and ***classexpr2(m)*** are non-disjoint

1: **if** both $classexpr_1(n)$ and $classexpr_(_2(m)$ are atomic expressions
$classexpr_1(1) \leftarrow classexpr_1(n)$
$classexpr_2(1) \leftarrow classexpr_2(m)$
$R \leftarrow$ ***CheckDisjointAtomicClasses***$(classexpr_1[1],classexpr_2[1])$
CheckDisjointAtomicClasses$(classexpr1,classexpr2)$ scans in the matrix G and returns 0,
i.e. non-disjoint or 1, i.e disjoint.
else
if $classexpr_1(n)$ is a complex expression containing union operator *"ObjectUnionOf"*
$R \leftarrow$ ***CheckDisjointClasses***$(classexpr_1[1],classexpr_2(m))$ + $CheckDisjointClasses(classexpr_1(n-1),classexpr_2(m))$
if $classexpr_1(n)$ is a complex expression containing union operator *"ObjectIntersectionOf"*
$R \leftarrow$ ***CheckDisjointClasses***$(classexpr_1[1],classexpr_2(m))$ * $CheckDisjointClasses(classexpr_1(n-1),classexpr_2(m))$
if $classexpr_2(m)$ is a complex expression containing union operator *"ObjectUnionOf"*
$R \leftarrow$ ***CheckDisjointClasses***$(classexpr_1(n),classexpr_2[1])$ + $CheckDisjointClasses(classexpr_1(n),classexpr_2(m-1))$
if $classexpr_2(m)$ is a complex expression containing union operator *"ObjectIntersectionOf"*
$R \leftarrow$ ***CheckDisjointClasses***$(classexpr_1(n),classexpr_2[1])$ * $CheckDisjointClasses(classexpr_1(n),classexpr_2(m-1))$
2: return R

Table 2: Parameter values for the GE

Parameter	Atomic Axioms	Complex Axioms
initLenChrom	5	30
maxWrap	2	2
pCross	80%	80%
pMut	1%	1%
popSize	2,000	2,000
nGenerations	25	5

while ours is stochastic. Therefore, the comparison in this case is unnecessary. Also, the overall recall value gets much higher, namely 0.323 over 3 runs, and would easily overtake the results of GoldMiner and [14] simply by executing more runs. We do not present the recall for complex axioms, because it is not clear how the cardinality of the set of *all* true complex axioms should be computed; under the most general assumptions, this set is infinite, although enumerable.

Fig. 3 plots the generality of discovered axioms against their possibility degree. Most discovered axioms are highly possible ($\Pi(\phi)$ close to 1) and most are supported by a large number of facts (instances) both in the atomic and complex case. In terms of generality, some discovered axioms have a particularly high generality, i.e. true axioms, such as DisjointClasses(dbo:Article dbo:Image) ($g_\phi = 2,220,106$) or DisjointClasses(dbo:Image ObjectUnionOf(ObjectUnionOf(dbo:Album dbo:TelevisionShow)
dbo:Website)) ($g_\phi = 190,783$). This can be explained by the existence of classes supported by a huge number of instances (like dbo:Article or dbo:Image) in the content of the generated axioms.

6 CONCLUSION

We have proposed an improved fitness function for a method using GE to learn OWL class disjointness axioms from RDF datasets. We have thoroughly tested the proposed method on a subset of DBpedia through extensive experiments with different parameter settings, to investigate the effectiveness of the method.

The results support the effectiveness of the proposed method to generate atomic axioms with high accuracy and coverage and suggest further improvements in the case of complex axioms. The results also shed more light on the use of GE to learn axioms. Based on their analysis, some promising research directions clearly emerge:

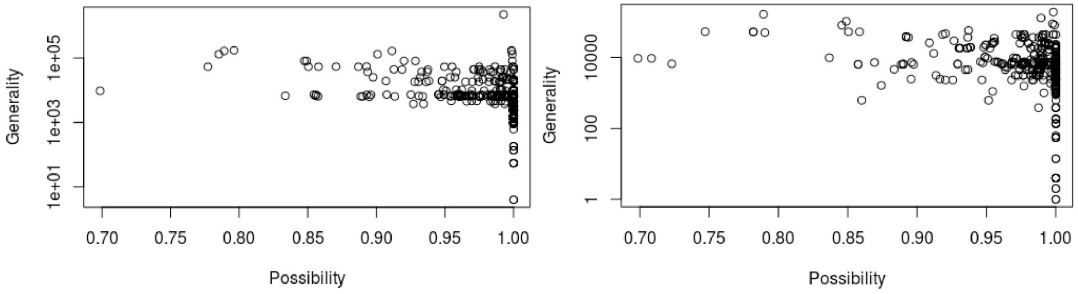
(1) tune the fitness function to better suit complex axioms;
(2) implement the method on parallel hardware to carry out a more ambitious experimental validation using bigger datasets.

REFERENCES

[1] Lorenz Bühmann and Jens Lehmann. 2012. Universal OWL Axiom Enrichment for Large Knowledge Bases. In *EKAW (Lecture Notes in Computer Science)*, Vol. 7603. Springer, 57–71.
[2] Lorenz Bühmann and Jens Lehmann. 2013. Pattern Based Knowledge Base Enrichment. In *International Semantic Web Conference (1) (Lecture Notes in Computer Science)*, Vol. 8218. Springer, 33–48.
[3] Tom Castle and Colin G. Johnson. 2010. Positional Effect of Crossover and Mutation in Grammatical Evolution. In *EuroGP (Lecture Notes in Computer Science)*, Vol. 6021. Springer, 26–37.
[4] Ian Dempsey, Michael O'Neill, and Anthony Brabazon. 2009. *Foundations in Grammatical Evolution for Dynamic Environments - Chapter 2 Grammatical Evolution.* Studies in Computational Intelligence, Vol. 194. Springer.
[5] Daniel Fleischhacker and Johanna Völker. 2011. Inductive Learning of Disjointness Axioms. In *OTM Conferences (2) (Lecture Notes in Computer Science)*, Vol. 7045. Springer, 680–697.
[6] Thomas R. Gruber. 1995. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.* 43, 5-6 (1995), 907–928.
[7] Nicola Guarino, Daniel Oberle, and Steffen Staab. 2009. *What Is an Ontology? Handbook on Ontologies.* Springer. 1–17 pages.
[8] Maryam Hazman, Samhaa R. El-Beltagy, and Ahmed Rafea. 2011. Article: A Survey of Ontology Learning Approaches. *International Journal of Computer Applications* 22, 8 (May 2011), 36–43.
[9] John R. Koza. 1993. *Genetic programming - on the programming of computers by means of natural selection.* MIT Press.
[10] William B. Langdon, Riccardo Poli, Nicholas Freitag McPhee, and John R. Koza. 2008. Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications. In *Computational Intelligence: A Compendium (Studies in Computational Intelligence)*, Vol. 115. Springer, 927–1028.
[11] Jens Lehmann. 2009. DL-Learner: Learning Concepts in Description Logics. *Journal of Machine Learning Research* 10 (2009), 2639–2642.

Table 3: Experimental results

	Our approach		Results in [14]		GoldMiner
	Atomic axioms	Complex axioms	Atomic axioms	Complex axioms	Atomic axioms
Precision	0.958 ± 0.011	0.876 ± 0.01	0.95 ± 0.02	0.867 ± 0.03	0.95
Recall (per run)	0.247 ± 0.01	N/A	0.15 ± 0.017	N/A	N/A
Recall (overall)	0.323 (over 3 runs)	N/A	0.69 (over 20 runs)	N/A	0.38



(a) Atomic axioms (b) Complex axioms

Figure 3: Possibility and generality distribution of the discovered axioms.

[12] Jens Lehmann and Johanna Völker. 2014. *Perspectives on Ontology Learning*. Studies on the Semantic Web, Vol. 18. IOS Press.

[13] A. Maedche and S. Staab. 2001. Ontology learning for the Semantic Web. *IEEE Intelligent Systems* 16, 2 (March 2001), 72–79.

[14] Thu Huong Nguyen and Andrea G. B. Tettamanzi. 2019. Learning Class Disjointness Axioms Using Grammatical Evolution. In *EuroGP (Lecture Notes in Computer Science)*, Vol. 11451. Springer, 278–294.

[15] M. O'Neill and C. Ryan. 2001. Grammatical Evolution. *Trans. Evol. Comp* 5, 4 (Aug. 2001), 349–358. https://doi.org/10.1109/4235.942529

[16] Michael O'Neill, Conor Ryan, Maarten Keijzer, and Mike Cattolico. 2003. Crossover in Grammatical Evolution. *Genetic Programming and Evolvable Machines* 4, 1 (2003), 67–93.

[17] Peter F. Patel-Schneider and Dieter Fensel. 2002. Layering the Semantic Web: Problems and Directions. In *International Semantic Web Conference (Lecture Notes in Computer Science)*, Vol. 2342. Springer, 16–29.

[18] Conor Ryan, J. J. Collins, and Michael O'Neill. 1998. Grammatical Evolution: Evolving Programs for an Arbitrary Language. In *EuroGP (Lecture Notes in Computer Science)*, Vol. 1391. Springer, 83–96.

[19] Mehrnoush Shamsfard and Ahmad Abdollahzadeh Barforoush. 2003. The state of the art in ontology learning: a framework for comparison. *Knowledge Eng. Review* 18, 4 (2003), 293–316.

[20] Andrea G. B. Tettamanzi, Catherine Faron-Zucker, and Fabien L. Gandon. 2014. Testing OWL Axioms against RDF Facts: A Possibilistic Approach. In *EKAW (Lecture Notes in Computer Science)*, Vol. 8876. Springer, 519–530.

[21] Johanna Völker, Daniel Fleischhacker, and Heiner Stuckenschmidt. 2015. Automatic acquisition of class disjointness. *J. Web Semant.* 35 (2015), 124–139. https://doi.org/10.1016/j.websem.2015.07.001

[22] Johanna Völker, Denny Vrandecic, York Sure, and Andreas Hotho. 2007. Learning Disjointness. In *ESWC (Lecture Notes in Computer Science)*, Vol. 4519. Springer, 175–189.

[23] L. A. Zadeh. 1978. Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems* 1 (1978), 3–28.